 Electronic Notes in Theoretical Computer Science 167 (2007) 425–447 

[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

Jordan Curves with Polynomial Inverse Moduli of Continuity

Ker-I Ko[a](#_bookmark0)*,*[1](#_bookmark1) *,*[2](#_bookmark1) Fuxiang Yu[a](#_bookmark0)*,*[1](#_bookmark1) *,*[3](#_bookmark1)

a *Department of Computer Science State University of New York Stony Brook, NY, U.S.A.*

Abstract

Computational complexity of two-dimensional domains whose boundaries are polynomial-time com- putable Jordan curves with polynomial inverse moduli of continuity is studied. It is shown that the membership problem of such a domain can be solved in *PNP* , i.e., in polynomial time relative to an oracle in *NP* , in contrast to the higher upper bound *PMP* for domains without the property of polynomial inverse modulus of continuity. On the other hand, the lower bound of *UP* for the membership problem still holds for domains with polynomial inverse moduli of continuity. It is also shown that the path problem of such a domain can be solved in *PSPACE*, matching its known lower bound, while no fixed upper bound was known for domains without this property.

# Introduction

In computable analysis, we know that if a real, one-to-one function *f* from [0*,* 1] to [0*,* 1] is computable, then its inverse function *f* −1 is also computable. One can find an approximate value of *f* −1(*y*) by the binary search method. [4](#_bookmark2)

1 This material is based upon work supported by National Science Foundation under grant No. 0430124.

2 Email: [keriko@cs.sunysb.edu](mailto:keriko@cs.sunysb.edu)

3 Email: [fuxiang@cs.sunysb.edu](mailto:fuxiang@cs.sunysb.edu)

4 Although the problem of determining whether two given real numbers are equal is un- decidable, we can use a modified binary search to avoid this issue. At each stage of the binary search, we can select two dyadic rationals *d* and *e* which are near each other, and

1571-0661 © 2007 Elsevier B.V. Open access under [CC BY-NC-ND license.](http://creativecommons.org/licenses/by-nc-nd/3.0/)

doi:10.1016/j.entcs.2006.08.022

However, there is, in general, no fixed complexity bound for *f* −1, even if *f* itself is known to be polynomial-time computable (see Ko [[9](#_bookmark24)]). To be more precise, the complexity of the inverse function *f* −1 depends exactly on its modulus of continuity, or, the inverse modulus of continuity of *f* :

Proposition 1.1 *Let f* : [0*,* 1] → [0*,* 1] *be a one-to-one function computable in time t*(*n*)*. Assume that m* : N → N *is an inverse modulus of continuity of f; that is, m satisﬁes that* |*x* − *y*|≥ 2−*n implies* |*f* (*x*) − *f* (*y*)|≥ 2−*m*(*n*)*. Then, f* −1 *is computable in time O*(*n* · *t*(*m*(*n*)))*.*

This simple fact suggests that, when we study the complexity of roots or the inverse of a given function *f* , we first need to know the inverse modulus of continuity of *f* . This idea also applies to one-to-one functions *f* which define more complicated objects. For instance, Berg et al. [[1](#_bookmark16)] pointed out that, from a constructive point of view, when a one-to-one function *f* : [0*,* 1] → R2 defining a Jordan curve on the two-dimension plane is “given,” the modulus of continuity of *f* −1 needs to be given too.

In this paper, we investigate the effect of the inverse modulus of continuity of a Jordan curve on the computational complexity of functions defined on two-dimensional domains. Chou and Ko [[4](#_bookmark18),[5](#_bookmark20),[6](#_bookmark21)] have studied the complexity issues of two-dimensional domains *S* whose boundaries are polynomial-time computable Jordan curves Γ. Among the problems studied in this investi- gation, results about the following two problems are incomplete. (We write Int(Γ) and Ext(Γ) to denote the interior and the exterior, respectively, of the curve Γ.)

Membership Problem: Given a point z on the two-dimensional plane and an integer *n >* 0, determine whether z locates in Int(Γ), or in Ext(Γ), or is close to the boundary Γ within a distance 2−*n*.

Path Problem: Given two points x and y in Int(Γ) with distance at least 2−*n* from the boundary Γ and an integer *m* ≥ *n*, find a (shortest) path from x to y that lies entirely within Int(Γ), and has a distance at least 2−*m* from the boundary Γ.

It was shown in Chou and Ko [[4](#_bookmark18)] that, for a polynomial-time computable Jordan curve Γ, the membership problem has an upper bound *PMP* (or, *PMidbitP* ) and a lower bound *UP* . The gap between the upper and lower bounds is quite big. For the (shortest) path problem, Chou and Ko [[5](#_bookmark20)] showed an upper bound of *PSP ACE* and a lower bound of #*P* for polynomial-time computable Jordan curves, assuming that such a path exists. It is, however,

test whether *f* (*d*) *< y*, *f* (*d*) *> y*, *f* (*e*) *< y* or *f* (*e*) *> y*. Since *f* is one-to-one, one of these four conditions must hold.

not clear whether this assumption holds for a given Jordan curve, even if it is known to be computable in polynomial time. Indeed, a polynomial-time computable Jordan curve could be very “complex.” For instance, Ko [[10](#_bookmark25)] and Ko and Weihrauch [[11](#_bookmark26)] constructed polynomial-time computable Jordan curves which have Hausdorff dimension two (and hence are fractals) and whose interior measures are not even recursive real numbers.

In this paper, we restrict our attention to two-dimensional domains whose boundaries are polynomial-time computable Jordan curves that also have poly- nomial inverse moduli of continuity. With respect to such curves, we are able to show the following:

1. For the complexity of the membership problem, the upper bound is re- duced to Δ*P* = *PNP* .

2

1. For the path problem, it is shown that there is a polynomial *q* such that, for any two points x and y in Int(Γ) that have distance 2−*n* away from the boundary Γ, there must be a path from x to y that lies in Int(Γ) and has distance 2−*q*(*n*) away from the boundary Γ.

Result (2) above is particularly useful when we consider analytic functions defined on two-dimensional regions. For instance, combining it with the al- gorithm for the shortest-path problem of Chou and Ko [[5](#_bookmark20)], we show that the problem of analytic continuation has an exponential space upper bound.

On the other hand, for the lower bounds, we observe that the Jordan curves constructed for the *UP* lower bound of the membership problem (Chou and Ko [[4](#_bookmark18)]) and the #*P* lower bound for the path problem (Chou and Ko [[5](#_bookmark20)]) both have polynomial inverse moduli of continuity. These results seem to indicate that Jordan curves with polynomial inverse moduli of continuity are a natural class of objects, and by studying such curves, we can get more accurate characterization of the complexity of related problems.

Our basic computational model for real-valued functions and two- dimensional regions is the oracle Turing machine model. For the general theory of computable analysis based on the Turing machine model, see, for instance, Pour-El and Richards [[14](#_bookmark28)] and Weihrauch [[17](#_bookmark29)]. For the theory of computa- tional complexity of real functions based on this computational model, see Ko [[9](#_bookmark24)]. The extension of this theory to include the computational complexity of two-dimensional regions has been presented in Chou and Ko [[4](#_bookmark18)]. Com- putational complexity of problems related to two-dimensional regions has been studied recently in several directions. Chou and Ko [[4](#_bookmark18)], Ko [[10](#_bookmark25)] and Ko and Weihrauch [[11](#_bookmark26)] studied the notion of polynomial-time computable two-dimensional regions. Rettinger and Weihrauch [[16](#_bookmark30)], Braverman [[2](#_bookmark17)], Ret- tinger [[15](#_bookmark31)] and Braverman and Yampolsky [[3](#_bookmark19)] studied the the computational

complexity of Julia sets. Chou and Ko [[5](#_bookmark20),[6](#_bookmark21)] studied the problem of finding paths in a two-dimensional domain. Ko and Yu [[12](#_bookmark27)] studied the problem of computing single-valued analytic branches of logarithm and square-root func- tions on a two-dimensional domain. Yu, Chou and Ko [[18](#_bookmark32)] studied the prob- lem of computing the minimum-area circumscribed rectangles and squares of a Jordan curve. All these works (and many others) used Turing machines and oracle Turing machines as the basic model.

# Definitions

This paper involves notions used in both discrete computation and continuous computation. The basic computational objects in discrete computation are integers and strings in {0*,* 1}∗. The length of a string *w* is denoted *l*(*w*).

The fundamental complexity classes we are interested in are the class *P* of sets accepted by deterministic polynomial-time Turing machines, and the class *FP* of functions (mapping strings to strings) computable by deterministic polynomial-time Turing machines. We will also use in this paper the following complexity classes (see, e.g., Du and Ko [[7](#_bookmark22)]):

#*P* : the class of functions that count the number of accepting paths of non- deterministic polynomial-time machines.

*UP* : the class of sets *A* accepted by an unambiguous nondeterministic polynomial-time Turing machine *M* such that for all *x* ∈ {0*,* 1}∗, *M* (*x*) has at most one accepting computational path.

*PSP ACE*: the class of sets accepted by polynomial-space Turing machines.

The basic computational objects in continuous computation are dyadic rationals D = {*m/*2*n* : *m* ∈ Z*,n* ∈ N}, and we denote D*n* = {*m/*2*n* : *m* ∈ Z}.

We use R to denote the class of real numbers and R2 the class of points on the two-dimensional plane. A point in R2 is denoted by a boldface font, such as z, or a pair of real numbers, such as ⟨*x, y*⟩. For any point z ∈ R2 and any set *S* ⊆ R2, we let dist(z*, S*) be the distance between z and *S*; that is, dist(z*, S*)= inf{|z − z'| : z' ∈ *S*}, where |·| denotes the absolute value. The boundary of a set *S* is written as *∂S*.

We also define a function *δ*(*s, t*), which can be viewed as the distance of two sets {*s* + *m* : *m* ∈ Z} and {*t* + *n* : *n* ∈ Z}. Assume that 0 ≤ *s < t* ≤ 1, we write *δ*(*s, t*)= min{*t* − *s,* 1+ *s* − *t*}. If *δ*(*s, t*) *<* 1*/*2 and *t* − *s* = *δ*(*s, t*), then we let *I*(*s, t*) be the interval [*s, t*]; otherwise, if *δ*(*s, t*) *<* 1*/*2 and *t* − *s > δ*(*s, t*) then let *I*(*s, t*)= [0*, s*] ∪ [*t,* 1].

We say a function *φ* : N → D *binary converges* to (or *represents*) a real number *x*, if (i) for all *n* ≥ 0, *φ*(*n*) ∈ D*n*, and (ii) for all *n* ≥ 0, |*φ*(*n*)−*x*|≤ 2−*n*.

For any *x* ∈ R, there is a unique function *bx* : N → D that binary converges to *x* and satisfies the condition *x* − 2−*n < bx*(*n*) ≤ *x* for all *n* ≥ 0. We call this function *bx* the *standard Cauchy function* for *x*. We say two functions *φx, φy* : N → D binary converge to (or represent) a complex number ⟨*x, y*⟩ if *φx* and *φy* binary converge to two real numbers *x* and *y*, respectively.

To compute a real-valued function *f* : R → R, we use oracle Turing ma- chines as the computational model. An oracle Turing machine *M* is an or- dinary Turing machine equipped with an extra query tape which it may use to make queries to an oracle function. *M* makes queries by writing the query on the query tape, and then reading the answer after the oracle replaces the query strings by an answer string. Each query made by *M* counts only one machine step. We use *Mφ*(*n*) to denote the output of machine *M* with regard to an input *n* and an oracle *φ*. We say an oracle Turing machine *M operates in polynomial time* if there exists a polynomial *p* such that for all inputs *n* and all oracles *φ*, *Mφ*(*n*) halts in time *p*(*n*).

Definition 2.1 (*a*) A function *f* : [0*,* 1] → R is said to be *computable* if there is an oracle Turing machine *M* that, on an oracle function *φ* : N → D that binary converges to a real number *x* and an input *n* ∈ N, outputs a string *d* ∈ D*n* such that |*d* − *f* (*x*)|≤ 2−*n*.

* 1. A function *f* : [0*,* 1] → R is *polynomial-time computable* if it is com- putable by an oracle Turing machine that operates in polynomial time.

The following equivalent definition for polynomial-time computable real functions *f* is useful. We say a function *f* : [0*,* 1] → R has a *polynomial modulus* if there exists a polynomial *p* such that |*x* − *y*| ≤ 2−*p*(*n*) implies

|*f* (*x*) − *f* (*y*)|≤ 2−*n*.

Proposition 2.2 *A function f* : [0*,* 1] → R *is polynomial-time computable if and only if*

* + 1. *f has a polynomial modulus, and*
    2. *There exists a Turing machine M and a polynomial p such that for any integer n and any d* ∈ D*m* ∩ [0*,* 1]*, M* (*d, n*) *outputs, in time p*(*m* + *n*)*, a dyadic rational number e such that* |*e* − *f* (*d*)|≤ 2−*n.*

We say a real function *f* : [0*,* 1] → R is computable in nondeterministic polynomial time if there are two polynomial functions *p, q*, and a deterministic oracle TM *M* such that for any oracle *φ* representing a real number *x* ∈ [0*,* 1] and any integer *n >* 0,

˜

1. *M φ*(*n, w*) halts in *p*(*n*) moves for all binary strings *w* ∈ {0*,* 1}*p*(*n*), and it may or may not output a dyadic rational;

˜

1. For at least one string *w* ∈ {0*,* 1}*p*(*n*), *M*˜*φ*(*n, w*) outputs a dyadic rational *d*;
2. If *M*˜*φ*(*n, w*) outputs a dyadic rational *d*, then |*d* − *f* (*x*)| ≤ 2 .

−*n*

We say a function *f* : [0*,* 1] → R2 represents a Jordan curve if (i) *f* is one-to- one on [0*,* 1) and *f* (0) = *f* (1), and (ii) the image of [0*,* 1] under *f* is the curve Γ. In this paper, we consider Jordan curves that are represented by polynomial- time computable functions *f* . Note that if *f* represents a Jordan curve and *f* is polynomial-time computable, then *f* has a polynomial modulus of continuity; that is, there is a polynomial *p* such that, for all *n >* 0, *δ*(*s, t*) ≤ 2−*p*(*n*) implies *f* (*s, t*) ≤ 2−*n*.

We say a function *m* : N → N is an inverse modulus of continuity of a function *f* : [0*,* 1] → R2 if there is a polynomial function *p* : N → N and an integer *n*0 such that the following holds for all *n > n*0: For any two points *s, t* ∈ [0*,* 1] |*f* (*s*) − *f* (*t*)| *>* 2−*p*(*n*) whenever *δ*(*s, t*) *>* 2−*n*. We say a Jordan curve Γ is polynomial-time computable and has a polynomial inverse modulus of continuity if there is a polynomial-time computable function *f* representing Γ which has a polynomial inverse modulus of continuity.

In general, when we deal with two-dimensional domains, there is a bound- ary issue. Consider, for instance, the membership problem: When a point z is very close to the boundary Γ, it requires more resources to decide whether it is inside or outside of Γ; moreover, it is undecidable whether z is on Γ. In other words, the function *f* : R2 → {−1*,* 0*,* 1} defined by

⎧ −1*,* z ∈ *Ext*(Γ)*.*

⎪

*f* (z)= ⎨ 0*,* z ∈ Γ*.*

⎪⎪⎩ 1*,* z ∈ *Int*(Γ)*.*

is not computable since it is not continuous on the boundary Γ. Similarly, for the path problem, when a path is very close to the boundary Γ, it is relatively harder to decide whether the path is inside Γ or not.

To overcome this problem, we adopt a less restrictive model of computation and allow the oracle Turing machines that solve the problems to make errors near the boundary. More precisely, as stated in Section 1, for an input *n* ∈ N, the membership problem of a point z needs to be solved only if the distance dist(z*,* Γ) is no less than 2−*n*, and the path problem of two points x and y needs to be solved only if dist(x*,* Γ) *>* 2−*n* and dist(y*,* Γ) *>* 2−*n*. Similar treatments can be found in Chou and Ko [[4](#_bookmark18)] and Ko and Yu [[12](#_bookmark27)].

# Inverse moduli and inverse functions

One-way functions are a fundamental concept in discrete complexity the- ory, with important applications in cryptography. Intuitively, a function *φ* : {0*,* 1}∗ → {0*,* 1}∗ is *one-way* if it satisfies the following properties:

* 1. *φ* is one-to-one,
  2. *φ* is polynomial-time computable, and
  3. *φ*−1 is not polynomial-time computable.

A function satisfying the above conditions, however, does not really capture the concept of one-way functions. For instance, for any integer *n >* 0, let *b*(*n*) denote the binary representation of *n*. Then, the function *φ*(0*n*)= *b*(*n*), and *φ*(*w*) = 0*w* if *w* /∈ {0}∗, satisfies the above conditions (i)–(iii). We note, however, that he inverse *φ*−1 of this function is actually very “easy” to compute; it is not polynomial-time computable simply because its output is too long to be written down in polynomially many steps. The usual way to get around this issue is to require that a one-way function *φ* must be *polynomially honest*; that is, *φ* must satisfy an addition condition:

* 1. There is a polynomial *q* such that *q*(*l*(*φ*(*w*))) ≥ *l*(*w*).

With this additional condition, the existence of one-way functions is no longer easy to prove. In fact, it is known that if *P* = *NP* then there are no one-way functions. More precisely, one-way functions with properties (i)–(iv) exist if and only if *P* = *UP* (see, e.g., Ko [[9](#_bookmark24)] and Du and Ko [[7](#_bookmark22)]).

The analogous notion of polynomial honestity for continuous functions is polynomial inverse modulus of continuity. Intuitively, it means that when two numbers *s* and *t* are far from each other, then their images *f* (*s*) and *f* (*t*) cannot be too close to each other. For instance, consider polynomial-time computable one-to-one functions *f* mapping [0*,* 1] to [0*,* 1]. It is easy to see that *f* −1 is polynomial-time computable if and only if *f* has a polynomial inverse modulus of continuity (cf. Proposition [1.1](#_bookmark3)).

For a function *f* defining a Jordan curve on the two-dimensional plane, it is not hard to show that if *f* is polynomial-time computable and has a polynomial inverse modulus of continuity, then the complexity of its inverse function *f* −1 is similar to that of discrete one-way functions.

Theorem 3.1 *Assume that f* : [0*,* 1] → R2 *represents a Jordan curve. Also assume that f has a polynomial inverse modulus of continuity. Then, there is a nondeterministic oracle TM M that computes the inverse function f* −1 *of f.*

Proof. Assume that *p* is a polynomial function that is both a time bound for *f* and an inverse modulus of continuity of *f* . To find an approximate value to *f* −1(z) within error 2−*n*, *M* works as follows:

Guess a dyadic point *d* ∈ D*p*(*p*(*n*)+1). Then, simulate the computation of *f* to find an approximate point w to *f* (*d*) with error ≤ 2−(*p*(*n*)+1). Outputs *d* if |w − z| ≤ 2−(*p*(*n*)+1).

Note that, if *f* (*e*)= z and |*f* (*d*) − z| ≤ 2−*p*(*n*), then |*d* − *e*| ≤ 2−*n*. Thus, the output of *M* is always correct. In addition, for any input z = *f* (*e*), if *M* guesses a dyadic point *d* ∈ D*p*(*p*(*n*)+1) such that |*d* − *e*| ≤ 2−*p*(*p*(*n*)+1)), then we must have |*f* (*d*) − z| ≤ 2−(*p*(*n*)+1), and *M* will accept. That is, for any z in the range of *f* , *M* always has at least one computation path that accepts. It is obvious that *M* always halts in polynomial time.

Corollary 3.2 *In the following,* (*a*)⇒(*b*)⇒(*c*)*:*

* + 1. *P* = *NP.*
    2. *For any polynomial-time computable function f* : [0*,* 1] → R2 *that rep- resents a Jordan curve* Γ *and has a polynomial inverse modulus of continuity, f* −1 *is polynomial-time computable.*
    3. *P* = *UP* ∩ *coUP.*

Proof. The part of (a) ⇒ (b) follows from Theorem [3.1](#_bookmark4).

(b) ⇒ (c): The proof is similar to the proof for the *UP* lower bound for the membership problem (see Theorem 7.3 of Chou and Ko [[4](#_bookmark18)]). We only give a sketch here.

Assume that *P* /= *UP* ∩*coUP* and *A* ⊆ {0*,* 1}∗ is in *UP* ∩*coUP* −*P* . Then, there exist two sets *B, C* ∈ *P* and a polynomial function *p* such that, for any *s* ∈ {0*,* 1}∗ of length *n*,

*s* ∈ *A* ⇐⇒ (∃!*t, l*(*t*)= *p*(*n*)) ⟨*s, t*⟩∈ *B,*

*s* /∈ *A* ⇐⇒ (∃!*t, l*(*t*)= *p*(*n*)) ⟨*s, t*⟩∈ *C,*

where (∃!*t*) means “there exists a unique *t*.”

Now, we define, for each string *s* ∈ {0*,* 1}∗ of length *n >* 0, a dyadic rational *as* as follows: Let *is* be the integer whose *n*-bit binary expansion (with possible extra leading zeros) is equal to *s*, and define *as* = 1 − 2−(*n*−1) +

*is* · 2−2*n*. That is, *a n* = 1 − 2−*n*+1, and *a n*−1 *,... ,a n* evenly divides the

0 0 1 1

interval [1 − 2−(*n*−1)*,* 1 − 2−*n*]. Also let *cs* = *as* + 2−2*n*−1.

Next, for any *s, t* ∈ {0*,* 1}∗ with length *l*(*s*)= *n* and *l*(*t*)= *p*(*n*), we define *bs,t* = *as* + *it* · 2−*p*(*n*)−2*n*. That is, {*bs,t* : *l*(*t*)= *p*(*n*)} evenly divides the interval [*as, as* + 2−2*n*].

Now, we define a function *f* : [0*,* 1] → R that satisfies the property of part (b): *f* is a piecewise linear function with the following breakpoints:

1. *f* (0) = *f* (1) = ⟨0*,* 1⟩, *f* (1*/*2) = ⟨1*,* 1⟩, *f* (3*/*4) = ⟨1*,* 0⟩, *f* (7*/*8) = ⟨0*,* 0⟩;
2. For *s, t* ∈ {0*,* 1}∗ with *l*(*s*) = *n >* 0 and *l*(*t*) = *p*(*n*), if ⟨*s, t*⟩ ∈ *B* ∪ *C*, then *f* has a bump with the breakpoints *f* (*bs,t/*2) = ⟨*bs,t,* 1⟩, *f* ((*bs,t* + 2−*p*(*n*)−2*n*−1)*/*2) = ⟨*cs,* 1+ 2−2*n*⟩, and *f* ((*bs,t* + 2−*p*(*n*)−2*n*)*/*2) =

⟨*bs,t* + 2−*p*(*n*)−2*n,* 1⟩.

Note that a string *s* is either in *A* or in *A*, and so there is a unique *t* of length *p*(*l*(*s*)) such that ⟨*s, t*⟩ ∈ *B* ∪ *C*. That is, the Jordan curve Γ defined by *f* has exactly one bump in [*as/*2*,* (*as* + 2−2*n*)*/*2], and so *f* is one-to-one and ⟨*cs,* 1+ 2−2*n*⟩ is always on Γ. Now, if we can compute *f* −1(⟨*cs,* 1+ 2−2*n*⟩), then we can get *t* such that ⟨*s, t*⟩ ∈ *B* ∪ *C*, and we can use this string *t* to determine whether *s* ∈ *A*. Thus, *A* /∈ *P* implies that *f* −1 is not polynomial- time computable.

We note that the above result is weaker than the analogous result in dis- crete complexity theory: one-way functions exists if and only if *P* /= *UP* . In particular, our condition (a) of *P* = *NP* appears much stronger than the condition *P* = *UP* . This seems to come from the nature of continuous com- putation: In the algorithm for *M* in the proof of Theorem [3.1](#_bookmark4), there may be *many* dyadic numbers *d* satisfying |*f* (*d*) − z| ≤ 2−(*p*(*n*)+1), and we cannot tell which one is the best approximation to *e*. It remains open whether the gap between conditions (a) and (c) of Corollary [3.2](#_bookmark5) can be narrowed.

# Membership problem

We assume, in this section, that *f* : [0*,* 1] → R2 is a polynomial-time com- putable function representing a Jordan curve Γ, and has a polynomial inverse modulus of continuity. That is, we assume that there exists a polynomial function *p* and a constant *n*0 *>* 0 such that (i) *f* is computable in time *p*(*n*), and (ii) for *n* ≥ *n* , *δ*(*s* − *t*) *>* 2−*n* implies |*f* (*s*) − *f* (*t*)| *>* 2−*p*(*n*). Without loss of generality, we assume that *p*(*n*) *>* 2*n* for all *n >* 0.

0

Recall that the membership problem about the curve Γ asks, for an oracle

⟨*φ*1*, φ*2⟩ representing a point z ∈ R2 and an input integer *n >* 0, whether z is in Int(Γ). An algorithm solving the membership problem is required to give the correct answer if dist(z*,* Γ) *>* 2−*n*.

In the following, we are going to present such an algorithm which runs in polynomial time relative to an oracle *A* ∈ *NP* . We first explain the basic idea of the algorithm.

First, we consider the curve Γ as a directed curve which begins at point *f* (0) and ends at *f* (1). In the following, by a *section* of Γ from a point x to a point y, written as Γxy, we mean either (i) the image of the interval [*a, b*] under *f* , if 0 ≤ *a < b* ≤ 1 and x = *f* (*a*), y = *f* (*b*), or (ii) the image of [*a,* 1]∪[0*, b*] under *f* if 0 ≤ *b < a* ≤ 1, and x = *f* (*a*), y = *f* (*b*). We assume that the directed curve Γ goes around its interior points in the counterclockwise direction; that is, for any interior point z0, the winding number of z0 about the directed curve Γ is equal to 1. (The case of Γ going around its interior points clockwise is symmetric.)

Now, we consider a point z with dist(z*,* Γ) *>* 2−*n*, where *n > n* . Let

0

*m* = *p*(*n*) + 1, and *q*(*n*)= *p*(*p*(*p*(*n*)+ 2))+ 5. Let x be a point on Γ satisfying

|x − z| ≤ dist(z*,* Γ) + 2−*q*(*n*)+2, and *C* the circle that centered at x with radius 2−*m*. Assume that Γ does not lie entirely within *C* (otherwise, it is obvious that z ∈ Ext(Γ).) Let x0 and x1 be two points on Γ ∩ *C* such that x lies on the section Γx0 x1 , and the whole section Γx0x1 lies inside the circle *C*. Also let y be the intersection point of the line segment zx and the circle *C*. (See Figure [1](#_bookmark6).)





*C*

**x**0

**x**

**y**

**z**

**y**1

**x**1

Fig. 1. Curve Γ enters circle *C* at x0 and leaves at x1.

We need to define the notion of the *orientation* of three points *P* , *Q*, *R*

around a point *X*. For any three points *P* , *Q*, *X* on the plane such that *Q* does

not lie on the halfline −−→ ∠*PXQ* denote the angle from the halfline −−→

−−→

*XP* , we let *XP*

to the halfline *XQ*, measured in the *counterclockwise* direction. (Thus, ∠*PXQ*

is always between 0 and 2*π*.) We say the orientation of the points *P* , *Q*, *R*

around *X* is in the *counterclockwise* direction if 0 *<* ∠*PXQ <* ∠*PXR <* 2*π*, and it is in the *clockwise* direction if 0 *<* ∠*PXR <* ∠*PXQ <* 2*π*. (The orientation is undefined if ∠*PXQ* or ∠*PXR* is equal to 0, or if ∠*PXQ* =

∠*PXR >* 0.)

Lemma 4.1 *The point* z *lies in the interior of* Γ *if and only if the orientation*

*of the three points* x1*,* y *and* x0 *around the point* x *is in the counterclockwise direction.*

−(*m*+1)

Proof. Let y1 be the point on xz with distance 2 from x. Notice that

|z − x| ≤ dist(z*,* Γ) + 2−*q*(*n*)+2 ≤ dist(z*,* Γ) + 2−(*m*+2). Therefore, the curve Γ does not intersect the line segment y1z. It follows that z ∈ Int(Γ) if and only if y1 ∈ Int(Γ).

We treat *C* as a directed curve going around the point x counterclockwise, and let Λ be the directed curve formed with Γx0x1 plus the arc of *C* from x1

'

to x0 (in the counterclockwise direction). Now we define x to be the point in

Γ ∩ xy that is the closest to y. Then, |x' − x| ≤ 2−*q*(*n*)+2.

*Case* 1. x' = x. This means that the Γ meets xy only at the point x. Since Γ goes around its interior points counterclockwise, points to its “left- hand side” are the interior points of Γ. Thus, y1 ∈ Int(Γ) if and only if y1 lies to the left of Γx0x1 , or equivalently, y1 ∈ Int(Λ). The last condition is equivalent to the condition that y lies on the arc of *C* from x1 to x0, or that the orientation of points x1*,* y*,* x0 around point x is in the counterclockwise direction (cf. Figure [1](#_bookmark6)).

1

*Case* 2. x' /= x. Then, we claim that x' lies on Γ

x0x1

. To see this, we let

x = *f* (*t* ), x = *f* (*t* ), x = *f* (*t*), and x' = *f* (*t*'). Assume, for the sake of

0 0 1 1

contradiction, that x' is not on Γ ; that is, *t*' /∈ *I*(*t ,t* ). Then, we must have

x0x1 0 1

*δ*(*t*'*, t*) ≥ min{*δ*(*t , t*)*, δ*(*t , t*)}. Since |x − x| = |x − x| = 2−*m*, we know that

0 1 0 1

both *δ*(*t , t*) and *δ*(*t , t*) are at least 2−*p*(*m*). Thus, we get *δ*(*t*'*, t*) ≥ 2−*p*(*m*), and

0 1

by the assumption of the inverse modulus *p*, |x' −x| ≥ 2−*p*(*p*(*p*(*n*)+1)) *>* 2−*q*(*n*)+2. It follows that |x' − z| *<* dist(z*,* Γ) and leads to a contradiction. We conclude

that x' must be on the section Γ

x0x1 .

Now, we see that the line segment x'y1 does not meet Γ other than the

point x', which lies on Γ

x0x1

. So, the argument for Case 1 also applies to Case

2, and the lemma is proven.

Based on Lemma [4.1](#_bookmark7), we can devise an algorithm to determine the mem- bership of z. In the following, let *m* = *p*(*n*)+ 1, *q*(*n*) = *p*(*p*(*p*(*n*) + 2)) + 5, and *r*(*n*) = *p*(*q*(*n*)). Without loss of generality, we assume that z ∈ D*q*(*n*)2 (otherwise, we can find a dyadic rational point z' ∈ D*q n* 2 with |z−z'| ≤ 2−*q*(*n*) and solve the membership problem about z' instead).

( )

1. Find a dyadic rational *d* ∈ D*q*(*n*) such that |*d* − dist(z*,* Γ)|≤ 2−*q*(*n*).
2. Find a dyadic rational *t* ∈ D*r*(*n*) ∩ [0*,* 1], and a dyadic point w ∈ D*q*(*n*)2 such that |w − *f* (*t*)| ≤ 2−*q*(*n*) and *d* − 2−*q*(*n*)+1 *<* |w − z| *< d* + 2−*q*(*n*)+1.
3. Find a dyadic rational point v ∈ D*q*(*n*)2 such that |v − y| *<* 2−*q*(*n*), where y

is the point in the line segment z*f* (*t*) that has the distance |y− *f* (*t*)| = 2−*m*.

1. Find two dyadic rationals *t*0*, t*1 ∈ D*r*(*n*) ∩ [0*,* 1], and two dyadic points

0

1

1

w0*,* w1

∈ D*q*(*n*)

2, with |w0

— *f* (*t* )| ≤ 2−*q*(*n*) and |w

— *f* (*t* )| ≤ 2−*q*(*n*),

that satisfy

(a) *δ*(*t ,t* ) ≤ 2−*n*+1 and *t* ∈ *I*(*t ,t* ); i.e., either (*t < t < t < t* + 2−*n*+1) or

0 1 0 1 0 1 0

(*t < t* +1 *< t* + 2−*n*+1 and *t* ∈ [0*,t* ] ∪ [*t ,* 1]);

0 1 0 1 0

(b)

0

1

|w0

— w|− 2−*m*

≤ 2−*q*(*n*)+2, and

|w1

— w|− 2−*m*

≤ 2−*q*(*n*)+2; and

(c) for any *u* ∈ D

*r*(*n*)

∩ *I*(*t ,t* ), |*f* (*u*) − w|≤ 2−*m* + 2−*q*(*n*)+2.

1. Determine the orientation of the three points w1*,* v and w0 around the point

w. Return the answer “z ∈ Int(Γ)” if and only if the orientation is in the counterclockwise direction.

Before we can prove that the above algorithm is correct, we need to verify that it is well-defined. First, for step (4), we need to verify that dyadic ratio- nals *t*0 and *t*1 satisfying conditions (a)—(c) do exist. Let x = *f* (*t*), and *C* the circle centered at x with radius 2−*m*. Then, from

|x − z| ≤ |x − w| + |w − z|≤ *d* + 2−*q*(*n*) + 2−*q*(*n*)+1

≤ dist(z*,* Γ) + 2−*q*(*n*)+2 ≤ dist(z*,* Γ) + 2−(*p*(*n*)+3)*,*

we see that the circle *C* and point x satisfy the condition of Lemma [4.1](#_bookmark7). Now let x0 and x1 be the points in Γ ∩ *C* as defined before Lemma [4.1](#_bookmark7); that is, x lies in Γx0 x1 , and the whole section Γx0x1 lies inside the circle *C*. Also let *f* (*s*0) = x0 and *f* (*s*1) = x1. Then, if we select *t*0 and *t*1 such that

|*t* − *s* | ≤ 2−*r*(*n*) and |*t* − *s* | ≤ 2−*r*(*n*), it can be verified that the conditions

0 0 1 1

* 1. —(c) of step (4) are satisfied.

Next, for Step (5), we need to verify that the orientation of the three points

w1, v, w0 around w is well-defined; that is, the angles ∠w1wv and ∠w1ww0

are not equal to zero and are not equal to each other. From conditions of steps

−*q*(*n*)+2

1. and (4), we know that points w1*,* v*,* w0 all lie within distance 2 of

the circle *C*. In addition, since |x − z| ≤ dist(z*,* Γ) + 2−*q*(*n*)+2, we know that dist(y*,* Γ) ≥ 2−*m* − 2−*q*(*n*)+2. It follows that both |v − w | and |v − w | are

0 1

at least 2−*m* − 2−*q*(*n*)+3 ≥ 2−*q*(*n*)+4, and hence the angle ∠w wv is greater

1

than 2−*q*(*n*), and the difference between ∠w wv and ∠w ww

is greater than

2−*q*(*n*).

1 1 0

Furthermore, we note that

|*f* (*t*0) − x| ≥ |w0 − w|− |*f* (*t*0) − w0|− |w − x|

≥ 2−*m* − 2−*q*(*n*)+2 − 2−*q*(*n*) − 2−*q*(*n*)

≥ 2−*m* − 2−*q*(*n*)+3 ≥ 2−(*p*(*n*)+2)*.*

From this, we get |*t* − *t* | ≥ |*t*

— *t*| ≥ 2−*p*(*p*(*n*)+2) and, by the assumption

0 1 0

about the inverse modulus *p*, |*f* (*t* ) − *f* (*t* )| ≥ 2*p*(*p*(*p*(*n*)+2)) = 2−*q*(*n*)+5, and

0 1

|w − w | ≥ 2−*q*(*n*)+4. It follows that the angle between ∠w ww

is greater

0 1 1 0

than 2−*q*(*n*). This shows that the orientation of the points w , v, w

1

0

around

w is well-defined and, in fact, can be determined in time polynomial in *n*.

Correctness. To prove that the algorithm always outputs the correct answer, we need, by Lemma [4.1](#_bookmark7), to verify that the orientation of points x1*,* y*,* x0 around x must be the same as that of the points w1*,* v*,* w0 around w. We note that, although w0 is intended to be an approximation to the point x0, the conditions (a)—(c) of step (4) do not guarantee this property. (In fact, the points x0 and x1 are, in general, like roots of a polynomial-time computable function, hard to compute from Γ and *C*.) Nevertheless, from the above analysis, we know

that all points x0*,* x1*,* w0 and w1

lie within 2−*q*(*n*)+2 of the circle *C*. In other

−*m* −*q*(*n*)+2

words, let *C*1 be the circle centered at x with radius 2 + 2 , *C* be the

0

circle centered at x with radius 2−*m* − 2−*q*(*n*)+2, and *C*' be the circle centered

at z with radius |z − x|− 2−*q*(*n*)+2. Then, these points all lie between *C* and

1

*C* but outside *C*', because dist(z*,* Γ) ≥ |z − x|− 2−*q*(*n*)+2 (see Figure [2](#_bookmark8)). Now,

0

let Λ0 be the section of Γ between *f* (*t*0) and x0 (i.e., Λ0 is equal to Γx0 *f* (*t*0 ) if *t*0 ∈ *I*(*s*0*, s*1), or equal to Γ*f*(*t*0 )x0 if *t*0 /∈ *I*(*s*0*, s*1)), and Λ1 be the section of Γ between *f* (*t*1) and x1 (i.e., Λ1 is equal to Γ*f*(*t*1 )x1 if *t*1 ∈ *I*(*s*0*, s*1), or equal to Γx1*f* (*t*1 ) if *t*1 /∈ *I*(*s*0*, s*1)). Then we claim that for any points v0 ∈ Λ0, v1 ∈ Λ1,

the distances |v − v |, |v − x| and |v − x| are all greater than 2−*q*(*n*)+5. To

0 1 0 1

see this, assume that v0 = *f* (*u*0) and v1 = *f* (*u*1). Then, from the fact that

|*f* (*t* ) − x| and |*f* (*s* ) − x| are greater than 2−*m* − 2−*q*(*n*)+2 ≥ 2−(*p*(*n*)+2), we

0 0

see that both |*s*0 − *t*| and |*t*0

— *t*| are greater than 2−*p*(*p*(*n*)+2). It follows that

|*u* − *t*| ≥ 2−*p*(*p*(*n*)+2) since *u* ∈ *I*(*s ,t* ). By the assumption of the inverse

0 0 0 0

−*p*(*p*(*p*(*n*)+2)) −*q*(*n*)+5

modulus *p*, we get |v0 − x| ≥ 2 = 2 . Similarly, we have

|v1

|v1

— x| ≥ 2−*q*(*n*)+5. Also, |*u*

— v | ≥ 2−*q*(*n*)+5.

1

0

— *u*1| ≥ |*u*0

— *t*| ≥ 2−*p*(*p*(*n*)+2), and we also have

Now, we show that the orientation of points x1*,* y*,* x0 around x and the orientation of *f* (*t*1)*,* y*,f* (*t*0) around x must be identical. First, we observe

that halflines −x−→x0 and −x−→x1 divide the area between circles *C*0, *C*1 and outside

*C*1

*R*0

**x**0

*C*0

*C*

*C*1

**x**

*R*2

**x**1

*R*1

*f* (*t*0 )

*C* ***’***

*R*0

**x**0

*C*0

*C*

*R*2

**x**

**x**1

*R*1

*f* (*t* 0 )

*C* ***’***

**y y**

(a) (b)

Fig. 2. (a) Λ0 passes near x; (b) Λ0 passes near x1.

*C*' into three parts. We let *R* be the part between −x−→x

0

0

and *C*', *R* be the

part between −x−→x1 and *C* , and *R* the part between the two halflines xx0

1

' −−→

2

and

−x−→x . Now, if the orientation of points x *,* y*,* x

around x and the orientation

1 1 0

of w1*,* v*,* w0 around w are not the same, then one of the following cases must

occur:

*Case* 1. w

∈ *R* . Then, *f* (*t* ) is in *R*

or within distance of 2−*q*(*n*) of the

0 1 0 1

halfline −x−→x1. We note that Λ0 does not intersect Γx0x1

except at x0. Therefore,

in order to move from x0 in region *R*0 to *f* (*t*0) in region *R*1, Λ0 must either

pass through the narrow area between x and *C*' (as shown in Figure [2](#_bookmark8)(a)) or pass through the area between x1 and *C*1 (as shown in Figure [2](#_bookmark8)(b)). In either way, it contradicts the above claim.

*Case* 2. w1 ∈ *R*0. We can show, similar to Case 1, that Λ1 must pass

through the area between x and *C*' or pass through the area between x and

0

*C*1, contradicting the claim.

*Case* 3. w0*,* w1 ∈ *R*2. In this case, both w0 and w1 are in *R*2, but w0 is closer than w1 to x1. Thus, either (i) Λ0 or Λ1 passes through the narrow area

between x and *C*' (i.e., Λ and Λ intersects line segment xy), or (ii) neither

0

1

Λ0 nor Λ1 intersects line segment xy. The case (i) is similar to Cases 1 and

2 (see Figure [3](#_bookmark9)(a)), and contradicts the claim. In the case (ii), since both w0 and w1 are in region *R*2, Λ0 and Λ1 must pass through each other within *R*2 at some points (see Figure [3](#_bookmark9)(b)). Again, this case contradicts the claim.

*f* (*t*1)

*R*0

**x**0

*C*

**x**

*R*2

*f* (*t*0 )

**x** *C*0

1

*C*1

*R*1

*C* ***’***

*f* ( *t* )

1

*R*0

**x**0

*C*0

*C*

*R*2

**x**

*f* (*t* 0 )

**x**1

*R*1

*C*1

*C****’***

**y y**

(a) (b)

Fig. 3. (a) Λ1 passes near x; (b) Λ0 and Λ1 pass near each other.

Thus, we conclude that the orientation of points x1*,* y*,* x0 around x and the orientation of points w1*,* v*,* w0 around w must be identical, and the algorithm always outputs the correct answer.

Complexity. It is shown in Chou and Ko [[4](#_bookmark18)] that the distance function *δ*(z) = dist(z*,* Γ) is computable in polynomial time relative to an oracle in *NP* . Thus, Steps (1) and (2) are computable in polynomial time relative to an oracle in *NP* . Also, it is clear that Step (3) is computable in polynomial time.

For step (4), we note that for any given *t*0 and *t*1, condition (a) can be verified in polynomial time. Furthermore, conditions (b) and (c) can be ver- ified in polynomial time with the help of an oracle in *NP* . In fact, let *M* be the oracle Turing machine that computes the function *f* , and let *Mu*(*k*) denote the dyadic rational in D*k* output by *M* using the standard Cauchy function *bu* as the oracle. Then, define *X* to be the set of all dyadic ra- tionals *s* ∈ D*r*(*n*) such that (i) *δ*(*s, t*) ≤ 2 , and (ii) (∃*u* ∈ D*r*(*n*)) [*u* ∈ *I*(*s, t*)*,* |*Mu*(*q*(*n*)) − w| *>* 2−(*p*(*n*)+1) + 2−*q*(*n*)+2}]. It is clear that *X* ∈ *NP* . We can now use a binary search to find the dyadic rational *t*0 not in *X* that has the least *δ*(*t, t*0). We note that such a point *t*0 must also satisfy the con-

−*n*+1

dition (b) that |*Mt*0 (*q*(*n*)) − w|− 2−(*p*(*n*)+1) ≤ 2−*q*(*n*)+2, and we can assign

w0 := *Mt*0 (*q*(*n*)). Similarly, we can use a binary search to find *t*1 and w1

using a similar oracle set in *NP* .

Finally, for Step (5), we already proved that the orientation of the three

points w1*,* v*,* w0 around w is well defined, with the angles ∠w1wv, ∠vww0

and ∠w ww all greater than 2−*q*(*n*). Thus, the orientation can be determined

0 1

in polynomial time by calculating these angles correct within 2−*q*(*n*)−1 and compare their values.

The above analysis completes the proof of the following theorem:

Theorem 4.2 *Assume that f* : [0*,* 1] → R2 *is a polynomial-time represen- tation of a Jordan curve* Γ *and that it has a polynomial inverse modulus of continuity. Then, the membership problem about* Γ *is solvable in polynomial time relative to an oracle in NP.*

Corollary 4.3 *Assume that P* = *NP. Then, a two-dimensional region, whose boundary is a polynomial-time computable Jordan curve with a poly- nomial inverse modulus of continuity, is P-recognizable.*

We note that, in contrast to Theorem [4.2](#_bookmark10), it is not hard to verify that the curve constructed for the *UP* lower bound for the membership problem, as given in Theorem 7.3 of Chou and Ko [[4](#_bookmark18)], actually has a polynomial inverse modulus. Thus, we have the following stronger results about the membership problem than that of Chou and Ko [[4](#_bookmark18)].

Corollary 4.4 *In the following,* (*a*)⇒(*b*)⇒(*c*)*:*

1. *P* = *NP.*
2. *For any Jordan curve* Γ *that has a polynomial-time representation with a polynomial inverse of modulus, the membership problem of* Γ *is solvable in polynomial time.*
3. *P* = *UP .*

# Path problem

Recall that the path problem asks for a path in the given domain *S* that connects two given points and has a distance 2−*m* from the boundary Γ of

*S*. However, it is in general undecidable whether such a path exists, even if Γ is polynomial-time computable. In this section, we show that if Γ has a polynomial-time representation which also has a polynomial inverse modulus, then a path with distance 2−*q*(*n*) away from Γ, for some polynomial *q*, must exist between two points which have distance 2−*n* away from Γ.

Theorem 5.1 *Assume that* Γ *is a Jordan curve represented by a polynomial- time computable function f which has a polynomial inverse modulus. Then, there exist a polynomial function q and a constant n*0 *satisfying the following conditions: For any integer n > n*0 *and any two points* z0*,* z1 *in* Int(Γ) *with*

dist(z *,* Γ) *>* 2−*n and* dist(z *,* Γ) *>* 2−*n, there is a path from* z *to* z

0 1 0 1

*that lies*

*in* Int(Γ) *with distance at least* 2−*q*(*n*) *from the curve* Γ*.*

Proof. Assume that *p* is a polynomial function that bounds the runtime of function *f* and the inverse modulus of *f* . Let *q*(*n*) = *p*(*p*(*n*)) + 2. As in the proof of Theorem [4.2](#_bookmark10), we regard Γ as a directed curve.

−*n*

First, we note that if |z0 − z | ≤ 2 , then it is obvious that the line

1

−(*n*+1) −*q*(*n*)

segment z0z1 has distance at least 2 ≥ 2 from Γ, and the required

−*n*

condition is satisfied. Therefore, we may assume that |z0 − z | *>* 2 .

1

We first draw a line *L*0 passing point z0 that is perpendicular to the line segment z0z1. Let v0 and w0 be the intersection points of *L*0 and Γ such that v0w0 lies entirely in Γ∪Int(Γ) and contains point z0. Then v0w0 divides Int(Γ) into two parts, with z1 belonging to one part. Move *L*0 toward the part that

−*q*(*n*)+2

does not contain z1 for distance 2 . Let x and y0 be the intersection of

0

this line *L*' and Γ such that x y lies entirely in Γ∪Int(Γ) and dist(z *,* x y )=

0 0 0

0 0 0

2−*q*(*n*)+2. Define lines *L* , *L*' and points x and y around point z

in a similar

1 1 1 1 1

way. We arrange the names of these four points x0*,* y0*,* x1*,* y1 in such a way

that the directed curve Γ goes through these points in the order of x0, x1, y1, y0, and back to x0 (We note that it is not possible for Γ to go in the order of x0, x1, y0, y1, since such a curve must leave either z0 or z1 in the exterior of Γ.) We also re-arrange the names of v0*,* w0*,* v1 and w1 so that v0, x0 are on the same side of z0, and w0, y0 are on the other side of z0; and that v1, x1 are on the same side of z1, and w1, y1 are on the other side.

We let *S* denote the part of Int(Γ) that is enclosed between x0y0 and x1y1. In other words, if we let Λ0 be the section of Γ from x0 to x1, and Λ1 be the section of Γ from y1 to y0, then *S* is the domain whose boundary is y0x0, Λ0, x1y1, plus Λ1. Let this boundary of *S* be Γ1. (See Figure [4](#_bookmark12)).

*Claim* 1. dist(Λ *,* Λ ) *>* 2−*q*(*n*)+2.

0 1

*Proof*. Let *s*0, *t*0, *s*1, and *t*1 be real numbers in [0*,* 1] such that *f* (*s*0)= x0, *f* (*t*0) = y0, *f* (*s*1) = x1, and *f* (*t*1) = y1. Without loss of generality, assume that 0 ≤ *s*0 *< s*1 *< t*1 *< t*0 *<* 1. Since x0 and y0 lie on the two sides of z0 with

distance at least 2−*n* from z , we must have |x

−y | *>* 2−*n*+1 −2−*q*(*n*)+3 *>* 2−*n*.

0 0 0

−*n*

Similarly, we also have |x1 − y | *>* 2 . From the assumption that *f* is

1

computable in time *p*(*n*), we have that *δ*(*s ,t* ) *>* 2−*p*(*n*) and *δ*(*s ,t* ) *>* 2−*p*(*n*).

0 0 1 1

It follows that, for any *s* ∈ [*s ,s* ] and *t* ∈ [*t ,t* ], *δ*(*s, t*) *>* 2−*p*(*n*). Then, by

0 1 1 0

the property of the inverse modulus *p*, we see that dist(Λ *,* Λ ) *>* 2−*p*(*p*(*n*)) =

0 1

2−*q*(*n*)+2.

*Claim* 2. v0 ∈ Λ0 and w0 ∈ Λ1.







**v**0

**x**0

**z *’***

0

**z**0

**y**0 **w**

0



***’***

***’***

**y**

1

*S*

**w**

1

**z*’***1

**z**1 **v**1

**x**1

Fig. 4. Domain *S*.

*Proof*. According to the definition of line *L*' , it is on the side of *L*

that

0 0

does not contain z1. Therefore, when we trace the curve Γ from x0 to x1, it

must first meet v0 before x1. Thus, v0 ∈ Λ0. Similarly, if we trace Γ backward from y0 toward y1, then we must meet w0 before y1, and so w0 ∈ Λ1.

Let z' be the point on x y that has |z − z' | = 2−*q*(*n*)+2, and z'

the point

0 0 0 0 0 1

on x y

that has |z

— z' | = 2−*q*(*n*)+2.

1 1 1 1

*Claim* 3. (a) Let *R*0 be the rectangle formed by the line segments z0z'

0

and

z' y0. Then, Λ0 ∩ *R*0 = ∅.

0

* 1. Let *R*1 be the rectangle formed by the line segments z1z' and z' x1.

1

1

Then, Λ1 ∩ *R*1 = ∅.

1

* 1. Let *R*2 be the rectangle formed by the line segments z1z'

1

and z' y1.

Then, Λ0 ∩ *R*2 = ∅.

*Proof*. (a) For the sake of contradiction, assume that Λ0 ∩ *R*1 /= ∅. It is clear that Λ0 does not run through the line segment z' y0. In addition,

0

from Claim 1, we know that dist(Λ *,* y ) *>* 2−*q*(*n*)+2. Therefore, Λ must

0 0 0

meet the halfline z−−w→

at some point w. From Claim 2, we know that this

0 0

intersection point w must lie on z−−w→ beyond the point w . Furthermore, since

0 0 0

dist(Λ *,* Λ ) *>* 2−*q*(*n*)+2, w must be at least 2−*q*(*n*)+2 beyond w , and y

must

0 1 0 0

be at least 2−*q*(*n*)+3 beyond the point w in *L*' (see Figure [5](#_bookmark13)). Now, consider

0 0

the section of Λ1 from w0 to y0. Since Λ1 cannot pass through z' y0, it must go from w0 to y0 passing through the area between w and z' y0. However,

0

0

this means that dist(w*,* Λ ) ≤ 2−*q*(*n*)+2, contradicting Claim 1.

1

The proofs for parts (b) and (c) are similar.

**v**0



**x**0

**z ’**

0

**y**

0

**z**0

**w**0



1

*R*0

0

Fig. 5. Domain *S*.

Now, define *T* = {z ∈ *S* : dist(z*,* Γ ) *>* 2−*q*(*n*)}. Then, z ∈ *T* . Let *C* be the connected component of *T* that contains z0. We claim that z1 belongs to the same component *C* and, hence, there is a path in *T* connecting z0 and z1, and the theorem follows.

1 0

To see this, let z'' be the point on z z' with |z' − z''| = 2−*q*(*n*). Then, z''

0 0 0 0 0 0

is on the boundary of *C*, and the boundary of *C* around z'' is a line segment parallel to *L*0. We follow this line segment in the direction toward y0. Then,

0

−*q*(*n*)

by Claim 3(a), it must reach a point z2 that is near Λ1 within distance 2 .

−*q*(*n*)

(This line segment must get near y0 ∈ Λ1 within distance 2 if not any

other point in Λ1.) Then, the boundary of *C* continuous on “parallel” to the curve Λ1. By Claim 1, it must then meet a point z3 that is near the line

segment x y within distance 2−*q*(*n*). (Note that the boundary of *C* does not

1 1

necessarily go continuously parallel to Λ1. It may “jump” from one section of

Λ to another section of Λ . However, it will be always within distance 2−*q*(*n*)

1

1

from some point in Λ1 before it gets near the line segment x1y1.) By Claim

3(b), we know that this point z has distance at least 2−*q*(*n*)+2 away from the

3

line segment z' x1. Thus, z3 must be close to a point u ∈ z' y1. Now, by Claim

1 1

3(c), the boundary of *C* must contain the line segment uz''. Therefore, z'' is

1 1

on the boundary of *C*, and it is obvious that z1 is in the component *C*. This

completes the claim and the theorem.

From Corollary 4.3 of Chou and Ko [[5](#_bookmark20)], we get the following result about the path problem:

Corollary 5.2 *Assume that* Γ *is a Jordan curve represented by a polynomial- time computable function which has a polynomial inverse modulus of continu- ity. Then, there exist an oracle Turing machine M, two polynomial functions p*(*n*)*, r*(*n*) *and a constant n*0 *such that the following holds:*

*For any oracles φ*1*, φ*2*, ψ*1*, ψ*2 *representing, respectively, two points* z0*,* z1 *in*

Int(Γ) *and any input n* ≥ *n , if* dist(z *,* Γ) *>* 2−*n and* dist(z *,* Γ) *>* 2−*n,*

0 0 0

*then M ﬁnds, using p*(*n*) *cells of storage space, a path π between* z0 *and* z1

*that lies within* Int(Γ) *with distance* dist(*π,* Γ) ≥ 2−*r*(*n*)*.*

Theorem 6.1 of Chou and Ko [[5](#_bookmark20)] showed a #*P* lower bound for the short- est path problem. More precisely, it states that, if *FP* /= #*P* , then there exists a domain *S* which has a polynomial-time computable boundary with respect to which the shortest path problem is not solvable in polynomial time. We observe that the boundary Γ of *S* construction in this proof actually has a polynomial inverse modulus of continuity. This means that the #*P* lower bound also holds for domains whose boundaries have polynomial-time repre- sentations with polynomial inverse moduli.

Corollary 5.3 *In the following,* (*a*)⇒(*b*)⇒(*c*)*:*

1. *P* = *PSP ACE.*
2. *For any Jordan curve* Γ *which has a polynomial-time representation with a polynomial inverse modulus, the path problem (with m* = *q*(*n*) *for some polynomial q) is solvable in polynomial time.*
3. *FP* = #*P .*

The computation of analytic continuation of a function defined on a two- dimensional domain is an important problem in computational complex anal- ysis. A simple approach, originated by Weierstrass (see, e.g., Henrici [[8](#_bookmark23)]), is to compute the analytic continuation through a path from a source point to the target point. This approach requires the fast computation of a path in- side the domain with a reasonable distance from the boundary of the domain. Our result of Theorem [5.1](#_bookmark11) gives an exponential space upper bound for this approach.

Theorem 5.4 *Assume that S is a simply connected domain on the two- dimensional plane whose boundary* Γ *is represented by a polynomial-time com- putable function which has a polynomial modulus of continuity, and that* z0 *is a point in* Int(Γ) *with* dist(z *,* Γ) = 2−*n*0 *. Also assume that function g is an- alytic on S, and that the power series of g at* z0 ∈ Int(Γ) *is polynomial-time computable. Then, there exists a polynomial q*(*n*) *such that for any integer*

0

*n* ≥ *n*0

*and any point* z ∈ Int(Γ) *with* dist(z*,* Γ) ≥ 2−*n, the power series of G*

*at* z *is computable using at most* 2*q*(*n*+*k*) *cells.*

Proof. Our proof is based on the analysis in Section 3.6 of Henrici [[8](#_bookmark23)], and we only present a simple sketch here.

From Corollary [5.2](#_bookmark14), we can find, in polynomial space a path *π* connecting

z and z with dist(*π,* Γ) ≥ 2−*p*(*n*) for some polynomial *p*. Then, we select points

0

z0*,* z1*,* z2*,... ,* z*m*, where *m* = 2*O*(*q*(*n*)), such that (i) |z*k*−1 −z*k*| are all equal and

less than *h* = 2−*q*(*n*) for *k* = 1*,... ,m* − 1, (ii) z*m* = z and |z*m*−1 −z*m*|≤ *h*, and

(iii) the radius of convergence of the power series of *g* at point z*k* is greater

than or equal to *h*. Let Σ∞ *a* (z )w*n* be the power series of *g* at point z ,

*n*=0

*n*

*k*

*k*

for *k* = 0*,* 1*,... , m*.

From Henrici’s analysis, we know that there exists a constant *c* such that an approximation to *an*(z*k*) can be computed from the first *n* · 2*cm* terms of

−*n*

the power series of *g* at z0 so that the error is bounded by 2 . To be more

precise, define a double sequence {*a*(*k*)}, for *k* = 0*,* 1*,... ,m* and *n* = 1*,* 2*,.. .*,

*n*

recursively as follows:

(1)

*c*

*a*(*k*) := Σ

*n*

*i*=*n*

*i* *a*(*k*−1)*hi*−*n,*

for *k* = 1*,* 2*,... , m*, and *n* = 1*,* 2*,.. .*. Then, it satisfies the condition that

*n*2

*n*

*i*

|*a*(*m*) − *a* (z

)|≤ 2−(*n*+1).

*n n m*

In the above formula ([1](#_bookmark15)), it is assumed that all arithmetic operations are calculated with the exact real-number arithmetics. It is not hard to verify, though, that if we carry out all calculations over dyadic rationals of 2*dn* bits for some constant *d >* 0, the extra error is bounded by 2−(*n*+1). Thus, this procedure works in our finite-precision model.

Finally, we note that the value of *a*(*k*) can be computed recursively using formula ([1](#_bookmark15)). This recursive computation can be viewed as a computation tree of height *m* and width 2*cm*, with each node *a*(*k*) at level *k* having *i*2*c* children. It is easy to see that this computation tree can be traversed in the depth-first order using only storage space *O*(*m*) (at any time of the traversal, the machine only needs to store one node at each level). At each node of the tree, the local computation can be done using only 2*O*(*q*(*n*)) cells of storage. Therefore, the total storage space required is only *m* · 2*O*(*q*(*n*)) = 2*O*(*q*(*n*)).

*i*

*i*

# Conclusion

In the previous sections, we studied the notion of polynomial inverse modulus of two-dimension Jordan curves. We presented evidence that the property of polynomial inverse modulus helps to reduce the complexity of some impor- tant problems about two-dimensional domains, including the inverse function problem, the membership problem, the path problem, and the analytic contin- uation problem. In addition, we observe that most Jordan curves constructed, in the earlier studies of two-dimensional domains, for the lower bound results for these problems satisfy this property of polynomial inverse modulus of con- tinuity. This includes the constructions for the lower bounds for the mem- bership problem and the path problem, as well as the fractals constructed for the nonrecursiveness result for the area problem (see Ko [[10](#_bookmark25)] and Ko and Weihrauch [[11](#_bookmark26)]).

From these studies, we found that the notion of polynomial-time com-

putable Jordan curves is probably too general, and some of these curves might be too complicated to understand its complexity issues. On the other hand, polynomial-time computable Jordan curves with polynomial inverse moduli are a rich, natural subclass, with respect to which we are able to get more accurate characterization of the complexity of some fundamental problems related to two-dimensional domains. It suggests that this notion is worth fur- ther study to get better understanding of the mathematical and algorithmic properties of two-dimensional domains.

# References

1. Berg, G., W. Julian, B. Mines and F. Richman, *The constructive Jordan curve theorem*, Rocky Mountain J. Math. 5, 225–234, 1975.
2. Braverman, M., *Hyperbolic Julia sets are poly-time computable*, Proceedings of the 6th Workshop on Computability and Complexity in Analysis, Vol. 120, Electron. Notes Theoret. Comput. Sci., Amsterdam, 17–30, 2005.
3. Braverman, M. and M. Yampolsky, *Non-computable Julia sets*, J. Amer. Math. Soc. 19(3), 551–578, 2006.
4. Chou, A. W. and K.-I. Ko, *Computational Complexity of Two-Dimensional Regions*, SIAM. J. Comput. 24, 923–947, 1995.
5. Chou, A. W. and K.-I. Ko, *On the complexity of ﬁnding paths in a two-dimensional domain* *I: Shortest paths*, Math. Logic Quart. 50(6), 551–572, 2004.
6. Chou, A. W. and K.-I. Ko, *On the complexity of ﬁnding paths in a two-dimensional domain* *II: Piecewise straight-line paths*, Proceedings of the 6th Workshop on Computability and Complexity in Analysis, Vol. 120, Electron. Notes Theoret. Comput. Sci., 45–57, 2005.
7. Du, D.-Z. and K.-I. Ko, “Theory of Computational Complexity”, John Wiley & Sons, New York, 2000.
8. Henrici, P., “Applied and Computational Complex Analysis”, Vol. 1-3. John Wiley & Sons, New York, 1974.
9. Ko, K.-I., “Complexity Theory of Real Functions”, Birkh¨*a*user, Boston, 1991.
10. Ko, K.-I., *A polynomial-time computable curve whose interior has a nonrecursive measure*, Theor. Comput. Sci. 145, 241–270, 1995.
11. Ko, K.-I. and K. Weihrauch, *On the measure of two-dimensional regions with polynomial-time computable boundaries*, Proceedings of 11th IEEE Conference on Computational Complexity, 150–159, 1996.
12. Ko, K.-I. and F. Yu, *On the complexity of computing the logarithm and square root functions on a complex domain*, Proceedings of 11th International Computing and Combinatorics Conference (COCOON), Vol. 3595, Lecture Notes in Computer Science, Springer, 349–358, 2005.
13. Mu¨ller, N. T., *Constructive aspects of analytic functions*, Proceedings of Workshop on Computability and Complexity in Analysis, Vol. 190, Informatik Berichte, FernUniversit¨at Hagen, 105–114, 1995.
14. Pour-El, M. B. and J. I. Richards, “Computability in Analysis and Physics”, Perspectives in Mathematical Logic, Springer, Berlin, 1989.
15. Rettinger, R., *A fast algorithm for Julia sets of hyperbolic rational functions*, Proceedings of 6th Workshop on Computability and Complexity in Analysis, Vol. 120, Electron. Notes Theoret. Comput. Sci., 145–157, 2005.
16. Rettinger, R. and K. Weihrauch, *The computational complexity of some Julia sets*, Proceedings of the 35th ACM Symposium on Theory of Computing, pp. 177–185, 2003.
17. Weihrauch, K., “Computable Analysis”, Springer, Berlin, 2000.
18. Yu, F., A. Chou and K.-I. Ko, *On the complexity of ﬁnding circumscribed rectangles for a two-dimensional domain*, Proceedings of 2nd International Conference on Computability and Complexity in Analysis, Vol. 326, Informatik Berichte, FernUniversit¨at in Hagen, 341–355, 2005.