In this note for an expository talk in Stanford’s Kiddie seminar I discuss some relations between topology and algorithms, in particular algorithms approximating roots of polynomials. Most of the material comes from [Sma87], [McM87] or [Far03], all of which are excellently written. Hence this exposition doesn’t deviate too much from those references.

Mathematicians have always been interested in finding roots of polynomials. Without that problem it’s unlikely we would have had Galois theory, group theory or complex analysis. For polynomials of degree $d \leq 4$ we can solve for the roots exactly using radicals, in general this is impossible. So for polynomials of degree $d \geq 5$ or for practical applications, you instead might want to approximate roots using some algorithm. In this note we discuss two results concerning the difficulty of finding algorithms that do this. Depending on one’s definition of algorithm, there is either Smale’s result giving a lower bound using topology or McMullen’s work giving a non-existence result using complex dynamics. We also briefly discuss the extension of Smale’s ideas to motion planning.

1. Smale’s lower bound on the complexity of finding roots

The problem we interested in solving with our “algorithms” is:

Given a monic complex polynomial $f$ of degree $d$ and an $\epsilon > 0$, find all the roots of $f$ with a precision of at least $\epsilon$.

In [Sma87] Smale proved that to solve this problem for $\epsilon$ sufficiently small, any algorithm must use at least $(\log_2 d)^{2/3}$ branchings, i.e. “IF, THEN”-statements.

We will start by explaining what exactly he means with an “algorithm” and branchings. Then we will give essentially a complete proof, modulo the computation of the $\mathbb{F}_2$-cohomology ring of braid groups.

1.1. “Algorithms” and their topology complexity. For Smale an algorithm is essentially a list of commands to do computations and occasionally check whether some number is smaller than some other number, branching into different directions based on the result.

We will give a definition that is a bit more general that his definition. Let’s enumerate the parts of the definition:

(i) An algorithm first of all has a space of inputs $I$, a space of intermediate states $P$ and a space of outputs $O$.

(ii) Next it consists of a “computation tree”, that is, a finite tree with a single root at the top, leaves at the bottom and internal vertices being either binary or ternary. The internal vertices with valence 2 are called computational nodes and those with valence 3 are called comparison nodes.

(iii) Each of the vertices is labeled by a continuous function; (a) the root is labeled by a function $I \to P$, (b) the computational nodes by functions $I \times P \to P$, (c) the comparison nodes by functions $I \times P \to \mathbb{R}$ and (d) the leaves by function $I \times P \to O$.

Definition 1.1. The topological complexity of an algorithm is the number of branching nodes.
Remark 1.2. Smale requires I, P and O to be real semi-algebraic subsets of Euclidean space and the functions to be rational functions. This is more in line with classical ideas of what operations one uses in solving or approximating roots to polynomials.

In our case we should have the inputs I equal to the space \( P(d) \) of monic polynomials of degree \( d \), which is homeomorphic to \( \mathbb{C}^d \) by looking at the coefficients. The outputs O should be the space \( R(d) \) of lists of \( d \) approximate roots, which is homeomorphic to \( \mathbb{C}^d \) as well.

At the moment this data is enough to describe an algorithm taking in polynomials and giving approximate roots. But it doesn’t solve our problem yet. For all we need to give a subspace \( S \subset I \times O \) of acceptable solutions. Here we pick

\[
S = \left\{ (f,(z_1, \ldots, z_d)) \in P(d) \times \mathbb{C}^d \mid f(z) = \prod_{i=1}^{d} (z - \zeta_i) \text{ and } |z_i - \zeta_i| < \epsilon \text{ for all } i \text{ in some ordering} \right\}
\]

Any \( x \in I \) gives a path through the tree and ends up at some \( S(x) \in O \). In particular, one applies the functions in the root, computation nodes or leaves, and picks branches at the branching nodes depending on whether the label function \( I \times P \to \mathbb{R} \) is \( \leq 0 \). An algorithm from I to O is said to solve for \( S \) if for all \( x \in I \) we have that \( (x, S(x)) \in S \).

We are interested in the following number \( TC(d, \epsilon) \).

Definition 1.3. Let \( TC(d, \epsilon) \) be the minimum over the topological complexity of all algorithms from \( I \to O \) solving for \( S \). We call this number the topological complexity of finding approximate roots.

The more precise version of Smale’s theorem is then:

Theorem 1.4 (Smale). For \( \epsilon > 0 \) sufficiently small (depending on \( d \)) we have that \( TC(d, \epsilon) \geq (\log_2 d)^{2/3} \).

To prove this we will convert the problem into first a question about topology and then a question about algebra.

1.2. Covering numbers. Look at the map from roots to polynomials, given by \((z_1, \ldots, z_d) \mapsto \prod_{i=1}^{d} (z - z_i)\). This is a continuous map \( \pi : R(d) \cong \mathbb{C}^d \to \mathbb{C}^d \cong P(d) \). Our algorithm \( s \) is a not necessarily continuous map \( \mathbb{C}^d \to \mathbb{C}^d \) that approximately satisfies \( \pi \circ s \). So in a sense finding an algorithm is trying to find an approximate section to \( \pi \).

The first step to getting to topology is showing that \( s \) can used to construct an actual section that is continuous over a subspace of the degree \( d \) monic polynomials. The idea is that if all roots are distinct and a certain bounded distance away from each other, then to a sufficiently good approximation to a root one can associate an actual root.

Let’s first the problem that the roots should be distinct. In \( R(d) \) the set where there are no double roots is the complement of \( \{(z_1, \ldots, z_d) \mid z_i = z_j \text{ for some } i \neq j \} \). This is the ordered configuration space \( F_d(C) \). In \( P(d) \) we simply remove the image of this set and this image can also be identified with the polynomials with vanishing discriminant. Sending a polynomial with non-vanishing discriminant to its \( d \) distinct roots identifies that subspace with the unordered configuration space \( C_d(C) \).

We conclude that in removing double roots, we could potentially reduce the situation to the \( d! \)-fold covering map \( F_d(C) \to C_d(C) \), which we again denote by \( \pi \). This is good for two reasons: covering maps are easy maps to study and we know an enormous amount about the algebraic topology of configuration spaces.

Our goal will now be the following. Find a deformation retract \( X \) of \( C_d(C) \) and a finite open cover \( U_i \) with continuous sections \( s_i \) of \( \pi \) over \( U_i \). The idea of \( X \) is that it makes the second condition hold, i.e. the roots are all a certain bounded distance away from each other. The \( U_i \) then correspond to subsets of \( C_d(C) \) that are hit by different paths down the computation tree and the \( s_i \) are constructed out of \( s \) by choosing the closest actual roots to the approximate roots given by \( s \).

Lemma 1.5. There exist an open neighborhood \( N \) of \( \pi(\{(z_1, \ldots, z_d) \mid z_i = z_j \text{ for some } i \neq j \}) \) such that \( X := C_d(C) \setminus N \) is a deformation retract of \( C_d(C) \).
Proof. One uses that there exists a triangulation of \( P(d) \) such that \( \pi(\{(z_1, \ldots, z_d) \mid z_i = z_j \text{ for some } i \neq j\}) \) is a subcomplex.

By construction \( X \) will have the property that there exists an \( \epsilon(d) > 0 \) such that for all point of \( X \) all roots will be distance at least \( 2\epsilon(d) \) from each other.

Fix an algorithm for our approximate root finding problem. Let \( V_i \subset X \) be those polynomials where the algorithm ends at leaf \( i \). Note that the index \( i \) goes from 0 to the number \( b \) of comparison nodes, i.e. there are \( b + 1 \) leaves. Restricted to \( V_i \), \( s \) is continuous. Thu \( V_i \) can be described as those points for which a certain combination of inequalities on the values of continuous functions holds, so is a closed subset of an open subset of a nice space and thus by the Tietze extension theorem we can find an open neighborhood \( U_i \) of \( V_i \) in \( X \) and an extension of \( s \) to \( U_i \). To get \( s_i \) we modify this extension of \( s \) as follows: for each \( f \in U_i \), we have that each \( z_i \in s(f) \) has a unique closest actual root \( \zeta_i \) of \( f \) and we set \( s_i(f) = (\zeta_1, \ldots, \zeta_n) \). Note that \( s_i \) is not the restriction of \( s \) to \( U_i \), even though the notation suggests that.

We draw the following conclusion from this discussion.

**Proposition 1.6.** If our problem is solved by an algorithm with \( b \) comparison nodes, then there exists an open cover \( U_i \) of \( (a \text{ deformation retract of }) \ C_d(\mathbb{C}) \) such that over each \( U_i \) the projection \( \pi: \pi^{-1}(X) \rightarrow X \) has a section \( s_i : U_i \rightarrow \pi^{-1}(X) \).

This leads to the definition of the covering number of a map \( f : X \rightarrow Y \).

**Definition 1.7.** The covering number \( \text{cov}(f) \) of \( f : X \rightarrow Y \) is the smallest number \( k \) such that there exists an open cover \( \{U_1, \ldots, U_k\} \) of \( Y \) with sections \( s_i : U_i \rightarrow X \) of \( f \).

**Remark 1.8.** This is a relative version of the Lusternik-Schnirelman category of a space \( X \): the Lusternik-Schnirelman is the smallest number \( k \) such that there exists an open cover \( \{U_1, \ldots, U_k\} \) of \( X \) such that all \( U_i \) are contractible.

From the previous discussion we conclude.

**Corollary 1.9.** For \( \epsilon \leq \epsilon(d) \), \( \text{cov}(\pi) - 1 \leq TC(d, \epsilon) \).

1.3. **Cup-length.** So we have now reduced the problem of determining the topological complexity of our problem to determining the covering number. There is a classical way to do this, essentially generalizes a method to bound the Lusternik-Schnirelman category.

This uses the notion of cup-length of a map \( f : X \rightarrow Y \) of spaces. The idea is that the map \( f : X \rightarrow Y \) induces a map on cohomology with coefficients in some field \( k \) (we’ll take \( k = \mathbb{F}_2 \) for the best bound):

\[
f^* : H^*(Y; k) \rightarrow H^*(X; k)
\]

and we can look at the kernel \( K \) of this map.

**Definition 1.10.** The cup-length \( \cup(f) \) of \( f \) is the maximal number \( k \) of elements \( \gamma_1, \ldots, \gamma_k \) in \( K \) such that the product \( \gamma_1 \cup \ldots \cup \gamma_k \) is non-zero.

Note that this only depends on the homotopy types of \( X, Y \) and the homotopy class of \( f : X \rightarrow Y \). We can thus replace \( \pi : \pi^{-1}(X) \rightarrow X \) with the covering map \( F_d(\mathbb{C}) \rightarrow C_d(\mathbb{C}) \), which we also denote by \( \pi \).

**Proposition 1.11.** We have that \( \text{cup}(f) < \text{cov}(f) \).

Proof. Suppose for contradiction that the cup-length of \( f \) is at least the covering number of \( f \). Then there is an open cover \( V_1, \ldots, V_k \) of \( Y \) with sections \( s_i : V_i \rightarrow X \) of \( f \) and elements \( \gamma_1, \ldots, \gamma_k \) in the kernel of \( H^*(Y; k) \rightarrow H^*(X; k) \) such that \( \gamma_1 \cup \ldots \cup \gamma_k \) is non-zero.

Consider the map \( H^*(Y) \rightarrow H^*(V_i) \) induced by inclusion. This factors as \( s_i^* f^* \) and thus sends \( \gamma_i \) to zero. By the long exact sequence of a pair this means that \( \gamma_i \) lifts to \( \eta_i \in H^*(Y, V_i) \). We have that \( \eta_1 \cup \ldots \cup \eta_k \in H^*(Y, \bigcup V_i) = H^*(Y, Y) = 0 \). But by naturality \( \eta_1 \cup \ldots \cup \eta_k \) is mapped to \( \gamma_1 \cup \ldots \cup \gamma_k \) by the map in the long exact sequence, and so we get a contradiction. □
Corollary 1.12. For $\epsilon$ sufficiently large and $k$ any field, $\text{cov}(\pi) \leq \text{TC}(d, \epsilon)$.

So our goal will be to compute the cup-length of $\pi$, which turns out to be highest if one uses $\mathbb{F}_2$-coefficients.

1.4. The mod 2 cohomology of braid groups. So now we get to the wonderful world of computing the cohomology of $C_d(\mathbb{C})$ and $F_d(\mathbb{C})$. Here we use a trick and quote some results.

First of all, let $B_{r_d}$ denote the braid group on $d$ strands. By keeping track of the endpoints of the string one sees there is a homomorphism to the permutation group $S_d$ for $B_{r_d} \rightarrow S_d$. The kernel $\text{PB}_{r_d}$ is called the pure braid group. Then we note that $C_d(\mathbb{C}) = K(B_{r_d}, 1)$ and $F_d(\mathbb{C}) = K(\text{PB}_{r_d}, 1)$, i.e. the spaces we care about are classifying spaces for the braid group and pure braid respectively. A lot is known about the cohomology of braid groups.

A more modern perspective allows one to see this as parts of the free or $E_{\infty}$-spaces we care about are free objects. In particular $\bigcup_d C_d(\mathbb{C})$ and $\bigcup_d B_{r_d}$ allow one to see these as parts of the free $E_{\infty}$-space on a point. And the map $\bigcup_d C_d(\mathbb{C}) \rightarrow \bigcup_d S_d$ is induced by this.

Using hard computations of many mathematicians, we know exactly what operations exist on $E_{\infty}$-spaces and since the spaces of interest are free this determines their homology. One conclusion is that $\pi_* : H_*(C_d(\mathbb{C}); \mathbb{F}_2) \rightarrow H_*(B_{r_d}; \mathbb{F}_2)$ is injective, hence the map in cohomology surjective. (The experts might be concerned about the Browder bracket, but none of these appear since the algebras we consider are free on one generator.)

Lemma 1.13. The map $\pi^* : H^*(C_d(\mathbb{C}); \mathbb{F}_2) \rightarrow H^*(F_d(\mathbb{C}); \mathbb{F}_2)$ is zero in positive degrees.

Proof. There is a commutative diagram

$$
\begin{array}{ccc}
C_d(\mathbb{C}) & \rightarrow & B_{r_d} \\
\downarrow & & \downarrow \\
F_d(\mathbb{C}) & \rightarrow & E_{r_d} \simeq *
\end{array}
$$

with the bottom map surjective on cohomology and the top-right composite zero in positive degrees. This implies that the left-hand map is zero on cohomology in positive degrees.

As a consequence, the cup-length of $\pi$ is just the cup-length of the ring $H^*(C_d(\mathbb{C}); \mathbb{F}_2)$ for $* > 0$.

Proposition 1.14 (Fuchs, Cohen). $H^*(C_d(\mathbb{C}); \mathbb{F}_2)$ is generated by $a_{k,m}$ for $k \geq 0$, $m \geq 1$ in degree $2^k 2^{m-1}$ with relations $a_{2, k}^2 = 0$ and $a_{m, k_1} \cdots a_{m, k_t} = 0$ if $2^{m_1+\ldots+m_t+k_1+\ldots+k_t} > d$.

Proof. The bigraded Hopf algebra $\bigoplus_{d \geq 0} H_*(C_d(\mathbb{C}); \mathbb{F}_2)$ is the $E_2$ mod-2 Dyer-Lashof algebra. One needs to compute the primitives under the coproduct to get the generators of cohomology. Now spend a lot of time trying to understand the notation in [CLM76].

So finally, we have reduced the problem to find the largest possible sequence of distinct pairs $(m_i, k_i)$ such that $m_1 + k_1 + \ldots + m_t + k_t < \log_2(d)$. Using “squares”, i.e. all $(m_i, k_i)$ such that $m_i \leq M$ and $k_i \leq M$ gives the bound $t = (\log_2 d)^{2/3}$. This concludes the proof.

Corollary 1.15. We have that for $\epsilon$ sufficiently small it holds that $\text{TC}(d, \epsilon) > (\log_2 d)^{2/3}$.

2. Related results

I want to talk about two related results. The first alters what the allowed type of algorithm is, the second applies the Smale’s ideas to a fashionable bit of topology known as motion planning.
2.1. Iterative root finding. Of course Smale’s notion of algorithm is not the only way one may try to get at approximate roots. In particular, it only allows for a finite number of steps. What if you alter the class of “algorithms” to include iterative procedures that should eventually converge to the roots? There is a theorem of McMullen [McM87] that says that for polynomials of degree at least 4 under certain restrictions such procedures do not exist.

McMullen defines a purely iterative algorithm as a rational map $T : P(d) \to \text{Rat}_k$. This means that we are trying to find approximate roots of polynomials by repeatedly applying degree $k$ rational functions which coefficients depend as a rational function on the coefficients of the polynomials. Since we might inadvertently end up dividing by zero, we only want for each $f \in P(d)$ that iterated applications $T_f$ converge for a dense open subset of $\mathbb{P}^1$ to a root of $f$. We say $T_f$ is conditionally convergent. Then $T$ is said to be generally convergent if $T_f$ is conditionally convergent for $f$ in a dense open subset of $P(d)$.

Theorem 2.1 (McMullen). (i) There is no generally convergent purely iterative algorithm for polynomials of degree at least 4.
(ii) In degrees $< 4$ they exist and we can completely classify them.

One example of a purely iterative algorithm is Newton’s method. It is known that Newton’s method fails to be generally convergent for polynomials of degree $\geq 3$. For example, there can large basins for which Newton’s method does not convergence but gets trapped around periodic orbits. This was a surprise to me, maybe it is to you as well, dear reader.

2.2. Robot motion planning. In [Far03] One place where Smale’s techniques have been particularly successful is motion planning. In that case given a space $X$, thought of as a configuration space for some physical system, we want to find a path in $X$ connecting given start- and endpoints. This is thus some function $X \times X \to PX$ that is a not necessarily continuous section to the evaluation map $\pi : PX \to X \times X$.

One might be interested in the topological complexity of this problem in the sense of Smale, or care about “instabilities” in a motion planning algorithm – that is, values at which the path that the algorithm gives suddenly changes dramatically – but one might want to find the analogue of covering number for this problem. That is, we want to find the minimum number of elements in an open cover $\{U_1, \ldots, U_k\}$ of $X \times X$ such that over each $U_i$ we have a continuous section $s_I$ of $\pi$. We denote this by $\text{TC}(X)$.

A classical example is that of a planar robot arm with $k$ circular joints. The configuration space $X$ of this arm is $(S^1)^k$ and the topological complexity of any planning algorithm is bounded below by the covering number of $(S^1)^k$.

This is computable via the following list of properties of $\text{TC}(\cdot)$.

Proposition 2.2. We have that

(i) $\text{TC}(X \times Y) \leq \text{TC}(X) + \text{TC}(Y) - 1$ and more generally $\text{TC}(\prod X_i) \leq 1 + \sum_i (\text{TC}(X_i) - 1)$.
(ii) $\text{TC}(S^n) = 2$ if $n$ is odd and $\text{TC}(S^n) = 3$ if $n$ is even.
(iii) For even field $k$, $\text{TC}(X)$ is strictly bounded below by the cup-length of the kernel of $H^*(X \times X; k) \to H^*(X; k)$.

In particular, properties (i) and (ii) tells us that $\text{TC}((S^1)^k) \leq k + 1$. On the other hand $H^*((S^1)^k \times (S^1)^k; \mathbb{Q})$ is an exterior algebra on $2k$ generator $a_1, \ldots, a_n, b_1, \ldots, b_n$, $H^*((S^1)^k; \mathbb{Q})$ is an exterior algebra on $k$ generators $c_1, \ldots, c_k$. The map $H^*(X \times X; \mathbb{Q}) \to H^*(X; \mathbb{Q})$ sends both $a_i$ and $b_i$ to $c_i$. This means that the kernel is spanned by all monomials that contain both $a_i$ and $b_i$ for some $1 \leq i \leq k$. The cup-length is at least $k$, given by taking $a_i b_i$ for $1 \leq i \leq k$. So we obtain

$$k < \text{TC}((S^1)^k) \leq k + 1$$

which proves that $\text{TC}((S^1)^k) = k + 1$. 
References


