November 6, 2026

Admin:
- HW 5 out.
- No class next Wednesday
- Office hours by appointment only next week!

Today: derandomization of special cases of PIT.

We'll see deterministic algorithms to test sparse polynomials. We'll also see a search-to-decision reduction for PIT.

Suppose we are given a $\Sigma\Pi$ circuit and we want to decide if it computes the zero polynomial. If we can see the circuit, then we just check if all monomials have a coefficient of zero in the output.

We'll design a stronger algorithm: we will decide if a $\Sigma\Pi$ circuit computes the zero polynomial only by making evaluation queries. (This is called <u>black-box</u> PIT. It's as if we are given a black box that implements the circuit, but we cannot look inside at the circuit itself.)

Main idea: reduce # of variables, keeping nonzero polynomials nonzero.

Suppose we are given oracle access to a polynomial

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{m} c_i \cdot x^{a^{(i)}}$$

of degree $\leq d$. How do we reduce # of variables?

Idea 1: Kronecker substitution

$$g(y) = f(y, y^{d+1}, y^{(d+1)^2}, \ldots, y^{(d+1)^{n-1}})$$

Expand

$$g(y) = \sum_{i=1}^{m} c_i \, y^{\sum_{j=1}^{n} a_j^{(i)} (d+1)^{j-1}}.$$

Each x-monomial in f is mapped to a unique y-monomial in $g(y)$. This means no cancellations take place, so $f = 0$ iff $g = 0$.

g is univariate of degree

$$\deg(g) \leq \deg(f) \cdot (d+1)^{n-1}$$

$$< (d+1)^n.$$

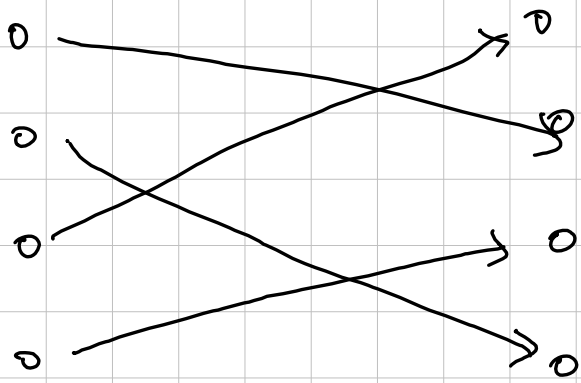
So we can test $g = 0$ by evaluating at $(d+1)^n$ points. This is too expensive.

We'll improve this by reducing the degree of the univariate substitution into f.

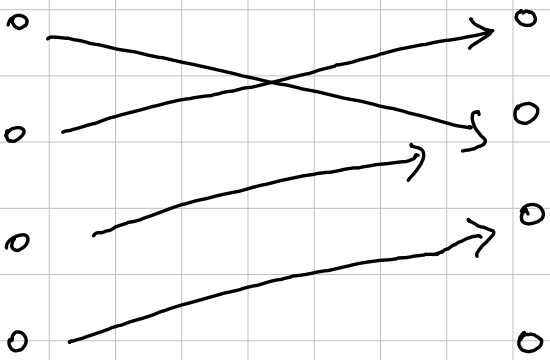
Instead of ensuring no monomials of f collide, it's enough to ensure that one monomial doesn't collide with any others. In a picture,

f monomials g monomials

Kronecker:



What we want:



Let's understand when two monomials might collide. Say we make the substitution

$$x_i \mapsto y^{k_i}.$$

Monomials x^a and x^b map to

$$x^a \to y^{\sum_i a_i k_i} \qquad x^b \to y^{\sum_i b_i k_i}$$

These collide if $\sum_i a_i k_i = \sum_i b_i k_i$.
Or in other words, if

$$p_{a,b}(z_1, \dots, z_n) := \sum_{i=1}^{n} (a_i - b_i) z_i$$

is nonzero at $z = k$.

Idea 2: Fix a monomial x^a that appears in f. Let

$$p_a(z_1, \dots, z_n) := \prod_{b \neq a} p_{a,b}(z),$$

where the product is over monomials

x^b of f. This is a polynomial of degree $\leq m$ (= sparsity of f) in n variables.

If $p_a(k) \neq 0$, then $x_i \to y^{k_i}$ isolates the monomial x^a.

So far, we've reduced PIT to PIT. Not much progress.

Idea 3: Instead of

$$p_{a,b}(z_1, \ldots, z_n) = \sum_i (a_i - b_i) z_i,$$

consider

$$q_{a,b}(z) := \sum_{i=1}^{n} (a_i - b_i) z^i.$$

If $q_{a,b}(k) \neq 0$, then $x_i \to y^{k^i}$ causes no collision between x^a and x^b.

Consider

$$q_a(z) := \prod_{b \neq a} q_{a,b}(z).$$

If $q_a(k) \neq 0$, then $x \to y^{k^i}$ isolates x^a. The polynomial q_a has degree

$$\deg(q_a) \leq m \cdot n,$$

so among any $mn+1$ choices of k, at least one is guaranteed to work. This gives us a small set of substitutions guaranteed to isolate a monomial, but there's one issue: the polynomial $g(y)$ still has high degree!

Idea 4: Pick a prime number p and use

$$x_i \to y^{k^i \bmod p}$$

Collisions happen if

$$q_{a,b}(k) = 0 \mod p$$

x^a is isolated if

$$q_a(k) \neq 0 \mod p.$$

Recall $\deg(q_a) = mn$. So pick a prime $p > mn$. Since q_a has at most mn roots, there is some $k \in \mathbb{Z}/p\mathbb{Z}$ so that

$$q_a(k) \neq 0.$$

This guarantees x^a is isolated. The substitution

$$g_k(y) := f(y^{k \bmod p}, y^{k^2 \bmod p}, \dots)$$

has degree $\leq dp \leq 2dmn$,

so we can test $g_k = 0$ in polynomial time.

Given an m-sparse poly f of deg $\leq d$, here's our final algo:

1) Pick a prime $mn < p \leq 2mn$

2) For every $0 \leq k < p$, let $g_k(y) \equiv f(y^{k \bmod p}, \ldots, y^{k^n \bmod p})$

   2a) Test $g_k \stackrel{?}{=} 0$ by evaluating at $mdn + 1$ points.

3) If $g_k \neq 0$ for some k, output $f \neq 0$

   If $g_k = 0$ for all k, output $f = 0$.

Next, we'll look at the search version of PIT

SEARCH - PIT

Input: arithmetic ckt C

Output: decide if $C(x) = 0$
if $C(x) \neq 0$, output a point $a \in \mathbb{C}^n$ such that $C(a) \neq 0$.

We'll show that an efficient algo for PIT implies an efficient algo for SEARCH - PIT.

Lemma (formula) $\forall$ PIT $\in P \Rightarrow$ SEARCH-(formula) $\forall$ PIT $\in P$

Proof In words, we find the point $a \in \mathbb{C}^n$ one coordinate at a time.

Algorithm: given a formula C of size s,

1) Decide if $C(x) = 0$. If so, return "$C = 0$"

2) For $i \leftarrow 1$ to n:

   2a) For $a_i \leftarrow 0$ to s
   test $C(a_1, \ldots, a_i, x_{i+1}, \ldots, x_n) = 0$

   2b) If $C(a_1, \ldots, a_i, x_{i+1}, \ldots) \neq 0$,
   fix this value of a_i,
   continue to $i+1$.

3) Output "$C \neq 0$" and $(a_1, \ldots, a_n)$.

Complexity: Poly-time, assuming PITEP

Correctness: if $C(x) \neq 0$, then $\deg(C) \leq s$. Schwartz-Zippel guarantees $\exists a \in \{0, 1, \ldots, 5\}^n$ such that $C(a) \neq 0$. □