Sum-Check Protocol

July 10, 2025

Distributed Lab

zkdl-camp.github.io

github.com/ZKDL-Camp

Introduction

Recap on Poly-IOPs and NILPs

Almost all protocols we have seen so far work over **univariate** $\mathbb{F}[X]$ **polynomials**. *Typical idea:* we aggregate information into some polynomial: say, $p(X)$ (which is typically a combination of other polynomials), and then check whether $p(u) = 0$ for every $u \in \Omega$:

$$p(u) = 0 \text{ for all } u \in \Omega \iff p(X) = q(X)\prod_{u\in\Omega}(X - u)$$

We check this inequality at a random point $r \leftarrow\$ \mathbb{F}$ and achieve soundness of $1 - \deg p/|\mathbb{F}|$: see **Schwartz-Zippel Lemma**.

For appropriate domains (typically $\Omega = \{\omega^j\}_{j\in[2^t]}$) we reduce all polynomials computations to $O(n \log n)$ complexity.

Motivation

But what if we could work with much smaller degrees?

Sum-Check-based Protocols

- Instead of n-variate univariate polynomials $\mathbb{F}[X]$ we reduce the problem to $\log n$-variate multivariate polynomials $\mathbb{F}[X_1, \dots, X_v]$.
- The Schwartz-Zippel Lemma still holds for multivariate case:

$$\Pr_{(r_1,\dots,r_v) \leftarrow \$ \mathbb{S}}[f(r_1, \dots, r_v) = 0] \leq \frac{\deg f}{|\mathbb{S}|}, \quad \mathbb{S} \subseteq \mathbb{F}^v$$

- **Bad news:** divisibility theorems do not hold:

$$f(s_1, \dots, s_v) = 0 \;\not\Longleftrightarrow\; (X - s_1) \dots (X - s_v) \mid f(X_1, \dots, X_v)$$

Sum-Check meaning

Instead of divisibility checks, we use the **Sum-Check**:

$$\sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_v \in \{0,1\}} f(b_1, \dots, b_v) = H$$

Summary in a Nutshell

Univariate World:

$$p(X) = q(X) \prod_{u \in \Omega} (X - u)$$

Multivariate World:

$$\sum_{\mathbf{b} \in \{0,1\}^{\ell}} f(b_1, \ldots, b_\ell) = H$$

Succinct Arguments: Ligero (2022), Spartan (2020), Libra (2019), Hyrax(2017), GKR (2008), . . .

Applications: zkGPT (2025), deep-prove (using *Ceno*) (2024), vSQL (2017), . . .

Multivariate Polynomials. Multilinear Extensions.

Some technicalities

Definition (Monomial)

By **monomial** in v variables we call expression $\mu(\mathbf{X}) = X_1^{a_1} \dots X_v^{a_v}$ where $a_1, \dots, a_v$ are non-negative integers. The **degree** of a monomial is naturally defined as $\deg \mu \triangleq a_1 + \cdots + a_v$.

Definition (Multivariate Polynomial)

Function $f : \mathbb{F}^v \to \mathbb{F}$ is called an v**-variate polynomial**, which is denoted by $f \in \mathbb{F}[X_1, \dots, X_v]$, if $f(\mathbf{X})$ is a finite linear combination of v-variate monomials $\{\mu_1(\mathbf{X}), \dots, \mu_n(\mathbf{X})\}$. The **degree** of f is defined as $\deg f \triangleq \max_{i \in \{1,\dots,n\}} \deg \mu_i$.

Example

$f(X_1, X_2, X_3) = X_1^3 + 3X_1^2X_2^2 + X_3^2 + X_3$ is a linear combination of 3-variate monomials $\{X_1^3, X_1^2X_2^2, X_3^2, X_3\}$, thus $f \in \mathbb{F}[X_1, X_2, X_3]$. It has degree $\deg f = 4$, corresponding to the second monomial $X_1^2X_2^2$.

Multilinear Polynomials

Definition (Multilinear Polynomial)

Multivariate polynomial $f \in \mathbb{F}[X_1, \ldots, X_v]$ is called **multilinear** if it is linear in each of the variables. Formally we have:

$$f(X_1, \ldots, X_v) = \alpha_j X_j + \beta_j, \quad j \in \{1, \ldots, n\},$$

where α_j, β_j do not depend on X_j.

Example

For example, $f(X_1, X_2, X_3) = X_1X_2 + 3X_1X_3 + X_2X_3$ is a multilinear polynomial in 3 variables. For instance, for X_1:

$$f(X_1, X_2, X_3) = (X_2 + 3X_3)X_1 + X_2X_3.$$

Similarly, $f(X_1, \ldots, X_v) = \prod_{i=1}^{v} X_i$ is a multilinear polynomial.

Boolean Hypercube

Definition (Boolean Hypercube)

By v**-dimensional boolean hypercube** we simply denote the set $\{0, 1\}^v$ (which is a set of binary strings of length v).

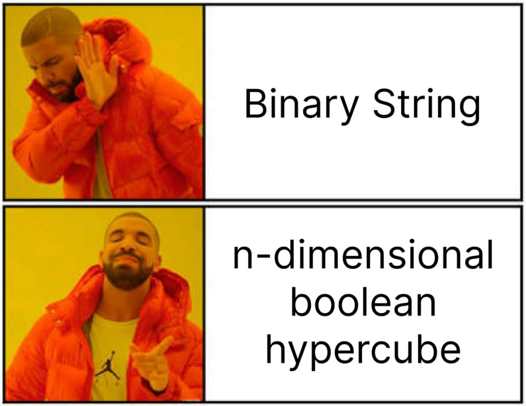

Figure: Cryptographers *love* overcomplicating things.

Boolean Hypercube Extensions

Definition (Boolean Hypercube Extension)

Suppose we are given the values on the boolean cybercube $f : \{0,1\}^v \to \mathbb{F}$. We call $\widetilde{f} : \mathbb{F}^v \to \mathbb{F}$ an **extension** if $\widetilde{f}(\mathbf{b}) = f(\mathbf{b})$ for every $\mathbf{b} \in \{0,1\}^v$.

	0	1
1	1	2
0	2	3

Function $f : \{0,1\}^2 \to \mathbb{F}_5$

Extend →

	0	1	2	3	4
0	1	2	0	0	2
1	2	3	1	1	3
2	0	1	4	4	1
3	0	1	4	4	1
4	2	3	1	1	3

Extension $\widetilde{f} : \mathbb{F}_5^2 \to \mathbb{F}_5$

$\widetilde{f}(X_1, X_2) = X_1^2 + X_2^2 + 1$

Multilinear Extensions

Definition (MLE)

An extension $\widetilde{f} : \mathbb{F}^v \to \mathbb{F}$ of $f : \{0,1\}^V \to \mathbb{F}$ is called **multilinear** if $\widetilde{f} \in \mathbb{F}[X_1, \dots, X_v]$ is a multilinear polynomial.

Example

For the previous example $f(0,0) = 1, f(1,0) = f(0,1) = 2, f(1,1) = 3$ (over $\mathbb{F}_5$) the multilinear extension is given by $\widetilde{f}(X_1, X_2) = X_1 + X_2 + 1$.

The question though is how many extensions $\widetilde{f}$ we can build.

- If $\widetilde{f}$ is a multivariate polynomial, there might be infinite number of choices: for example above, take $\widetilde{f}_n(X_1, X_2) = X_1^n + X_2^n + 1$.
- However, if $\widetilde{f}$ is multilinear, it is **unique**.
- Additionally, is there an analogy to the *Lagrange Interpolation* for such case: how to build $\widetilde{f}$ practically?

Lagrange Interpolation of multilinear polynomials

Theorem (Lagrange Interpolation of Multilinear Polynomials)

Any function over the v-dimensional hypercube $f : \{0,1\}^v \to \mathbb{F}$ has a unique v-variate multilinear extension $\widetilde{f} \in \mathbb{F}[X_1, \ldots, X_v]$. It is defined using the ***Lagrange interpolation of multilinear polynomials****:*

$$\widetilde{f}(\mathbf{X}) = \sum_{\boldsymbol{b} \in \{0,1\}^v} f(\boldsymbol{b}) \cdot \text{eq}(\mathbf{X}; \boldsymbol{b}),$$

where the set $\{\text{eq}(\mathbf{X}; \boldsymbol{b})\}_{\boldsymbol{b} \in \{0,1\}^v}$ is referred to as ***the set of multilinear Lagrange basis polynomials*** *over $\{0,1\}^v$. Each basis polynomial (among 2^v) $\text{eq}(\mathbf{X}; \boldsymbol{b})$ is defined as:*

$$\text{eq}(\mathbf{X}; \boldsymbol{b}) \triangleq \prod_{i=1}^{v} \{X_i b_i + (1 - X_i)(1 - b_i)\}.$$

Lagrange Interpolation: Example

Suppose we want to build the MLE for $f : \{0,1\}^2 \to \mathbb{F}_{11}$ given by:

$$f(0,0) = 3, \quad f(0,1) = 4, \quad f(1,0) = 1, \quad f(1,1) = 2$$

Step 1. Define multilinear Lagrange basis polynomials:

$$\mathsf{eq}(X_1, X_2; (0,0)) = (1 - X_1)(1 - X_2), \quad \mathsf{eq}(X_1, X_2; (0,1)) = (1 - X_1)X_2,$$
$$\mathsf{eq}(X_1, X_2; (1,0)) = X_1(1 - X_2), \quad \mathsf{eq}(X_1, X_2; (1,1)) = X_1 X_2$$

Step 2. Find the appropriate linear combination:

$$\begin{aligned}\widetilde{f}(X_1, X_2) &= \sum_{\boldsymbol{b} \in \{0,1\}^2} f(\boldsymbol{b}) \cdot \mathsf{eq}(\mathbf{X}; \boldsymbol{b}) \\ &= 3(1 - X_1)(1 - X_2) + 4(1 - X_1)X_2 + X_1(1 - X_2) + 2X_1X_2 \\ &= \boxed{-2X_1 + X_2 + 3}\end{aligned}$$

Fact: Generally, $\widetilde{f}(\boldsymbol{r})$ for $\boldsymbol{r} \xleftarrow{\$} \mathbb{F}^v$ can be computed in $O(2^v)$ time.

Sum-Check MatMul Protocol

MatMul Sum-Check Protocol

MatMul Protocol

Goal: Verify that $C = AB$ for matrices $A, B, C \in \mathbb{F}^{n \times n}$.

Naïve approach: Send A, B to the verifier $\mathcal{V}$, then $\mathcal{V}$ computes AB.

Time Complexity: $\mathcal{O}(n^3)$, *Space Complexity:* $\mathcal{O}(n^2)$.

Freiveld's Protocol: Send A, B, C to the verifier $\mathcal{V}$. Sample $\boldsymbol{r} \xleftarrow{\$} \mathbb{F}^n$ and verify that $A(B\boldsymbol{r}) = C\boldsymbol{r}$.

Time Complexity: $\mathcal{O}(n^2)$, *Space Complexity:* $\mathcal{O}(n^2)$.

Sum-Check Protocol. Apply the sum-check to some particular equation formed by multilinear extensions of matrices.

Time Complexity: $\mathcal{O}(n^2)$, *Space Complexity:* $\mathcal{O}(\log n)$.

Question

Where the hell should the Sum-Check be applied here?

Matrices Multilinear Extensions

For simplicity assume $n = 2^k$ for some k.

Idea: Instead of perceiving A, B, C as the collection of n^2 field elements, perceive them as functions $f_A, f_B, f_C : \{0,1\}^{\log n} \times \{0,1\}^{\log n} \to \mathbb{F}$, mapping two binarized indices of the matrix to the corresponding value. For example,

$$f_A(\boldsymbol{i}, \boldsymbol{j}) = A_{i,j}, \quad \text{where} \quad \boldsymbol{i} = (i_1, \dots, i_{\log n}),\ \boldsymbol{j} = (j_1, \dots, j_{\log n}).$$

Example

Suppose $A = \begin{bmatrix} 0 & 1 \\ 2 & 0 \end{bmatrix} \in \mathbb{F}_{13}^{2\times 2}$. Then, f_A is defined as:

$$f_A(0,0) = f_A(1,1) = 0, \quad f_A(0,1) = 1, \quad f_A(1,0) = 2$$

Its MLE is given by $\widetilde{f_A}(X,Y) = (1-X)Y + 2X(1-Y) = 2X + Y - 3XY$.

The Trick

Given functions $f_A, f_B, f_C : \{0,1\}^{\log n} \times \{0,1\}^{\log n} \to \mathbb{F}$, we build the corresponding MLEs $\widetilde{f}_A, \widetilde{f}_B, \widetilde{f}_C : \mathbb{F}^{\log n} \times \mathbb{F}^{\log n} \to \mathbb{F}$. Now what?

Lemma

$$\widetilde{f}_C(\mathbf{x}, \mathbf{y}) = \sum_{\boldsymbol{b} \in \{0,1\}^{\log n}} \widetilde{f}_A(\mathbf{x}, \boldsymbol{b}) \cdot \widetilde{f}_B(\boldsymbol{b}, \mathbf{y})$$

Reasoning. Both sides are multilinear polynomials in $\mathbf{x}$ and $\mathbf{y}$. Since MLE over $\{0,1\}^{2\log n}$ is unique, it suffices to check the equality only over $\boldsymbol{i}, \boldsymbol{j} \in \{0,1\}^{\log n}$. Then,

$$\widetilde{f}_C(\boldsymbol{i}, \boldsymbol{j}) = \sum_{\boldsymbol{b} \in \{0,1\}^{\log n}} \widetilde{f}_A(\boldsymbol{i}, \boldsymbol{b}) \cdot \widetilde{f}_B(\boldsymbol{b}, \boldsymbol{j})$$

Note that this is exactly the check $C_{i,j} = \sum_{b=1}^{n} A_{i,b} B_{b,j}$!

Now, apply Sum-Check on $h(\mathbf{z}) = \widetilde{f}_A(\boldsymbol{r}_1, \mathbf{z}) \widetilde{f}_B(\mathbf{z}, \boldsymbol{r}_2)$ for $\boldsymbol{r}_1, \boldsymbol{r}_2 \xleftarrow{\$} \mathbb{F}^{\log n}$.

Idea of Spartan

This protocol might sound too abstract, but this idea of using matrix MLEs is used in **Spartan**! Recall that in QAP we check:

QAP Check: $$\sum_i z_i \ell_i(X) \cdot \sum_i z_i r_i(X) - \sum_i z_i o_i(X) = 0, \quad X \in \Omega$$

Spartan General Idea:

1. Commit to the MLE extension $\widetilde{f}_Z(\mathbf{Y})$ of the solution-witness $\mathbf{z}$.
2. In universal setup, find MLEs $\widetilde{f}_L, \widetilde{f}_R, \widetilde{f}_O$.
3. Reduce R1CS satisfability to **zero-check** on

$$\zeta(\mathbf{X}) = \sum_{\boldsymbol{b}} \widetilde{f}_L(\mathbf{X}, \boldsymbol{b})\widetilde{f}_Z(\boldsymbol{b}) \cdot \sum_{\boldsymbol{b}} \widetilde{f}_R(\mathbf{X}, \boldsymbol{b})\widetilde{f}_Z(\boldsymbol{b}) - \sum_{\boldsymbol{b}} \widetilde{f}_O(\mathbf{X}, \boldsymbol{b})\widetilde{f}_Z(\boldsymbol{b})$$

4. Apply some dark magic to make above work.

Sum-Check Protocol

Sum-Check Protocol Goal

Sum-Check Goal

Prover $\mathcal{P}$ wants to convince the verifier $\mathcal{V}$ that:

$$\sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_v \in \{0,1\}} f(b_1, \dots, b_v) = H$$

Note: $f \in \mathbb{F}[X_1, \dots, X_v]$ is not necessarily a multilinear polynomial.

Naïve IP: $\mathcal{V}$ takes f and computes the sum. It requires the time and space $\mathcal{O}(2^v)$ — not gud for *succinct* argument systems.

Round 1. The prover $\mathcal{P}$ sends the value $C_1 \in \mathbb{F}$ which is the claimed value of H. Then, the prover computes:

$$f_1(X_1) := \sum_{(b_2,\dots,b_v) \in \{0,1\}^{v-1}} f(X_1, b_2, \dots, b_v)$$

Question: If f is multilinear, then what form does f_1 have?

Sum-Check, Round #1

Assume prover sends $s_1(X_1)$. Verify $\mathcal{V}$ needs to check:

1. $s_1(X_1)$ is indeed $f_1(X_1)$.
2. $s_1(X_1)$ (and thus $f_1(X_1)$) is consistent with the claimed C_1.

Second is easy: check $s_1(0) + s_1(1) = H$. Indeed:

$$\begin{aligned} s_1(0) + s_1(1) \\ &= \sum_{(b_2,\dots,b_v)\in\{0,1\}^{v-1}} f(0, b_2, \dots, b_v) + \sum_{(b_2,\dots,b_v)\in\{0,1\}^{v-1}} f(1, b_2, \dots, b_v) \\ &= \sum_{(b_1,\dots,b_v)\in\{0,1\}^{v}} f(b_1, \dots, b_v) = H \end{aligned}$$

For the second, apply the Schwartz-Zippel Lemma: pick $r_1 \xleftarrow{\$} \mathbb{F}$ and check whether $s_1(r_1) = f_1(r_1)$. Computing $s_1(r_1)$ is trivial, but *how to compute* $f_1(r_1)$ *effectively*?

Sum-Check, Subsequent Rounds

Idea: Apply the same procedure again!

Round 2. The prover $\mathcal{P}$ computes $s_2(X_2)$ claimed to equal:

$$f_2(X_2) = \sum_{(b_3,\dots,b_v)\in\{0,1\}^{v-2}} f(r_1, X_2, b_3, \dots, b_v).$$

For the consistency check, $\mathcal{V}$ verifies that $s_2(0) + s_2(1) = s_1(r_1)$.
Then, the verification boils down to checking whether $s_2(r_2) = f_2(r_2)$.

Round j. The prover $\mathcal{P}$ computes $s_j(X_j)$ claimed to equal:

$$f_j(X_j) = \sum_{(b_{j+1},\dots,b_v)\in\{0,1\}^{v-j}} f(r_1, \dots, r_{j-1}, X_j, b_{j+1}, \dots, b_v).$$

For the consistency check, $\mathcal{V}$ verifies that $s_j(0) + s_j(1) = s_{j-1}(r_{j-1})$.
Then, the verification boils down to checking whether $s_j(r_j) = f_j(r_j)$.

Sum-Check, Wrap-up

Last Round. The verifier $\mathcal{V}$ picks a random $r_v \leftarrow\$ \mathbb{F}$ and verifies whether $s_v(r_v) = \mathcal{O}^f(r_1, \ldots, r_v)$ where $\mathcal{O}^f(\cdot)$ is an oracle access to the function f (e.g., commitment + opening).

Lemma (Sum-Check Soundness Lemma)

Let $f \in \mathbb{F}[X_1, \ldots, X_v]$ be a multivariate polynomial of degree at most d in each variable, defined over the finite field $\mathbb{F}$. For any given $H \in \mathbb{F}$, let $\mathcal{L}$ be the language of all polynomials f (given as an oracle) such that

$$H = \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_v \in \{0,1\}} f(b_1, \ldots, b_v).$$

The sumcheck protocol is an IOP for $\mathcal{L}$ with the completness error $\delta_C = 0$ and the soundness error $\delta_S \leq vd/|\mathbb{F}|$.

Sum-Check Performance

Lemma (Sum-Check Performance)

Assume the average cost of calling $\mathcal{O}^f(\cdot)$ *is* T, $d = \deg f$, *and* $n = 2^\nu$. *Then, the following is true about the performance of Sum-Check:*

- ***Proof Size:*** $\mathcal{O}(d \log n)$.
- ***Verifier Time:*** $\mathcal{O}(d \log n) + T$.
- ***Prover Time:*** $\mathcal{O}(nT)$.

However, this is an IP. How to turn it to the *non-interactive* protocol?

Simply apply the **Fiat-Shamir heuristic**! At round j, the transcript is $\tau = (H, s_1, r_1, \dots, s_{j-1}, r_{j-1}, s_j)$, thus the randomness can be sampled simply as $r_j \leftarrow \mathcal{O}^R(\tau)$ for a random oracle $\mathcal{O}^R(\cdot)$.

Now the coding time!

Thank you for your attention

♥

zkdl-camp.github.io

github.com/ZKDL-Camp