Efficient multivariate low-degree tests
via interactive oracle proofs of proximity

Sarah Bordage
Ecole Polytechnique, Institut Polytechnique de Paris & Inria

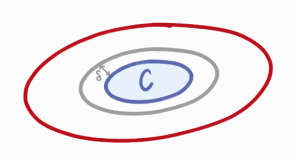
Joint work with Daniel Augot and Jade Nardi

WCC 2022
March 8, 2022

Topic of this talk

Proximity testing for a code

Given a (linear) code $C \subset \mathbb{F}^n$ and oracle access to $f \in \mathbb{F}^n$, algorithmically distinguish between $\begin{cases} f \in C \\ f \text{ is } \delta\text{-far from } C \end{cases}$ with $q = o(n)$ queries to f.

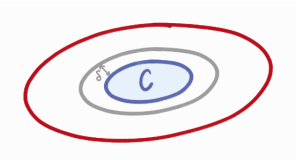


Given domain L of size n, identify $\mathbb{F}^n$ with $\mathbb{F}^L$: **codewords** of linear code $C \subset \mathbb{F}^n$ ⟷ **functions** in $\mathbb{F}^L$.

Topic of this talk

Proximity testing for a code

Given a (linear) code $C \subset \mathbb{F}^n$ and oracle access to $f \in \mathbb{F}^n$, algorithmically distinguish between $\begin{cases} f \in C \\ f \text{ is } \delta\text{-far from } C \end{cases}$ with $q = o(n)$ queries to f.



Given domain L of size n, identify $\mathbb{F}^n$ with $\mathbb{F}^L$: **codewords** of linear code $C \subset \mathbb{F}^n$ ⟷ **functions** in $\mathbb{F}^L$.

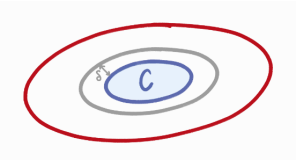
Multivariate polynomial codes

Assume $L \subset \mathbb{F}$ and $d < |L|$.

Topic of this talk

Proximity testing for a code

Given a (linear) code $C \subset \mathbb{F}^n$ and oracle access to $f \in \mathbb{F}^n$, algorithmically distinguish between $\begin{cases} f \in C \\ f \text{ is } \delta\text{-far from } C \end{cases}$ with $q = o(n)$ queries to f.



Given domain L of size n, identify $\mathbb{F}^n$ with $\mathbb{F}^L$: **codewords** of linear code $C \subset \mathbb{F}^n$ ⟷ **functions** in $\mathbb{F}^L$.

Multivariate polynomial codes

Assume $L \subset \mathbb{F}$ and $d < |L|$.

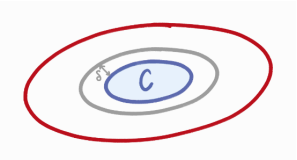
- **Tensor product of RS codes:**

  $\mathrm{RS}[L, d]^{\otimes m} = \{f : L^m \to \mathbb{F} \mid f \text{ evaluation of a poly in } \mathbb{F}[X_1, \ldots, X_m] \text{ with individual degrees} < d\}$

Topic of this talk

Proximity testing for a code

Given a (linear) code $C \subset \mathbb{F}^n$ and oracle access to $f \in \mathbb{F}^n$, algorithmically distinguish between $\begin{cases} f \in C \\ f \text{ is } \delta\text{-far from } C \end{cases}$ with $q = o(n)$ queries to f.



Given domain L of size n, identify $\mathbb{F}^n$ with $\mathbb{F}^L$: **codewords** of linear code $C \subset \mathbb{F}^n$ ⟷ **functions** in $\mathbb{F}^L$.

Multivariate polynomial codes

Assume $L \subset \mathbb{F}$ and $d < |L|$.

- **Tensor product of RS codes:**

  $\mathrm{RS}[L, d]^{\otimes m} = \{f : L^m \to \mathbb{F} \mid f \text{ evaluation of a poly in } \mathbb{F}[X_1, \ldots, X_m] \text{ with individual degrees} < d\}$

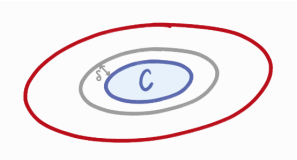
- **Reed-Muller codes:**

  $\mathrm{RM}[L, d, m] = \{f : L^m \to \mathbb{F} \mid f \text{ evaluation of a poly in } \mathbb{F}[X_1, \ldots, X_m] \text{ of total degree} < d\}$

Topic of this talk

Proximity testing for a code

Given a (linear) code $C \subset \mathbb{F}^n$ and oracle access to $f \in \mathbb{F}^n$, algorithmically distinguish between $\begin{cases} f \in C \\ f \text{ is } \delta\text{-far from } C \end{cases}$ with $q = o(n)$ queries to f.



Given domain L of size n, identify $\mathbb{F}^n$ with $\mathbb{F}^L$: **codewords** of linear code $C \subset \mathbb{F}^n$ ⟷ **functions** in $\mathbb{F}^L$.

Multivariate polynomial codes

Assume $L \subset \mathbb{F}$ and $d < |L|$.

- **Tensor product of RS codes:**

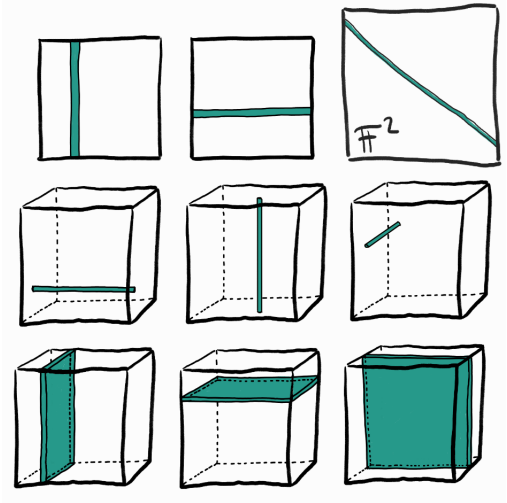
  $\mathrm{RS}[L, d]^{\otimes m} = \{f : L^m \to \mathbb{F} \mid f \text{ evaluation of a poly in } \mathbb{F}[X_1, \ldots, X_m] \text{ with individual degrees} < d\}$

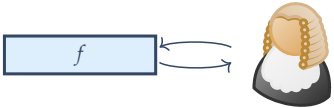
- **Reed-Muller codes:**

  $\mathrm{RM}[L, d, m] = \{f : L^m \to \mathbb{F} \mid f \text{ evaluation of a poly in } \mathbb{F}[X_1, \ldots, X_m] \text{ of total degree} < d\}$

Proximity tests for polynomial codes ⟷ **Low-degree tests**

Multivariate polynomial codes are locally testable

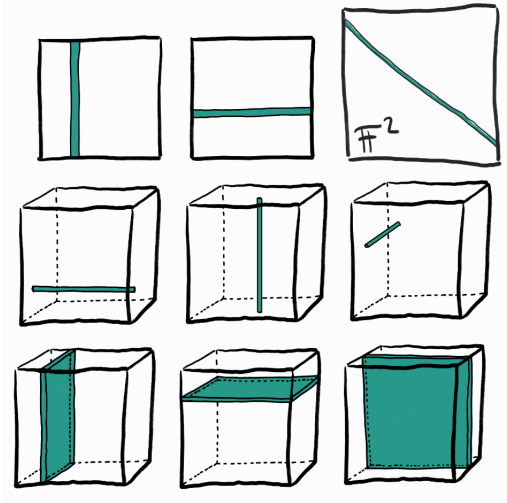


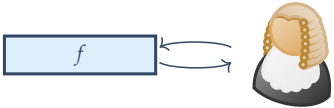


Axis-parallel tests
→ **Test individual degree**

Random line tests
→ **Test total degree**
(Require evaluation domain = $\mathbb{F}^m$)

Multivariate polynomial codes are locally testable





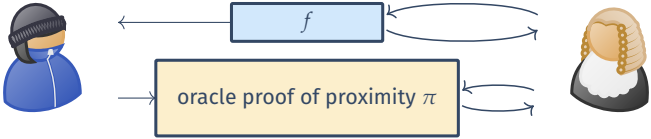
Axis-parallel tests
→ **Test individual degree**

Random line tests
→ **Test total degree**
(Require evaluation domain = $\mathbb{F}^m$)

Oracle access to f only
→ **At least d queries**

Testing proximity in the PCP model

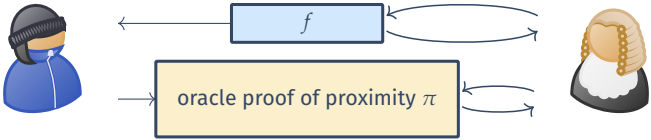
Probabilistically Checkable Proof of Proximity (PCPP):



- Ask a prover to provide an **auxiliary proof** π

Testing proximity in the PCP model

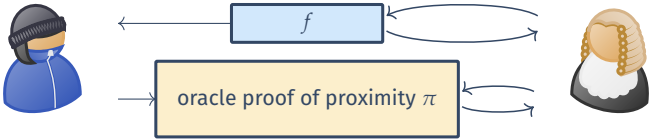
Probabilistically Checkable Proof of Proximity (PCPP):



- Ask a prover to provide an **auxiliary proof** π
- Enable sublinear tests for **non-locally testable codes** (e.g. Reed-Solomon codes)

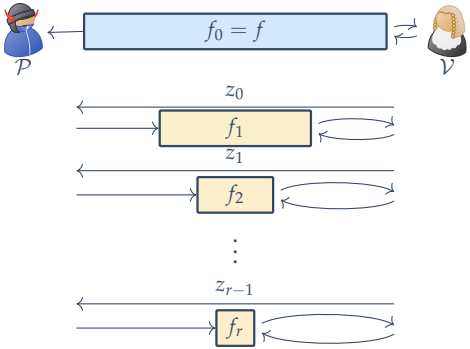
Testing proximity in the PCP model

Probabilistically Checkable Proof of Proximity (PCPP):



- Ask a prover to provide an **auxiliary proof** π
- Enable sublinear tests for **non-locally testable codes** (e.g. Reed-Solomon codes)
- Prover must compute in advance the answers to all possible queries ⟶ **impractical**

Interactive Oracle Proof of Proximity



DEF IOPP for a Code

An **Interactive Oracle Proof of Proximity (IOPP)** $(\mathcal{P}, \mathcal{V})$ for C with soundness error $s : (0,1] \to [0,1]$ satisfies:

Completeness

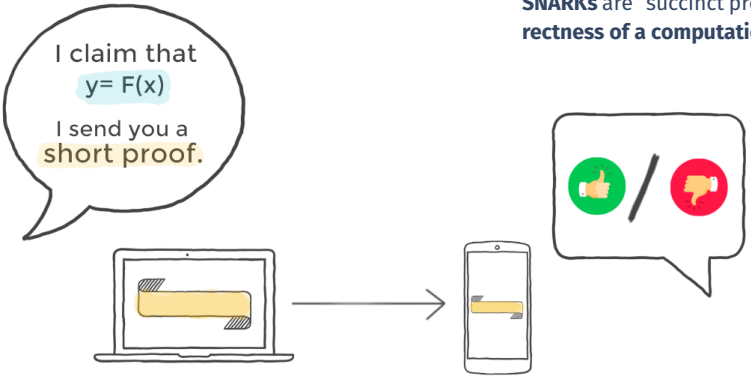
If $f \in C$, then $\exists \mathcal{P} \Pr[\langle \mathcal{P}(f), \mathcal{V}^f \rangle = 1] = 1$.

Soundness

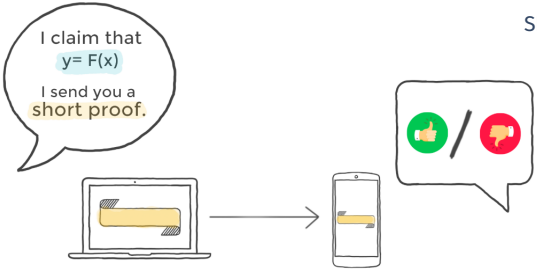
If f is δ-far from C,
Then, for all unbounded $\tilde{\mathcal{P}}$, $\Pr[\langle \tilde{\mathcal{P}}, \mathcal{V}^f \rangle = 1] \leq s(\delta)$.

Motivation: (ZK-)SNARKs

SNARKs are “succinct proofs” that enable to check **correctness of a computation**, with/without **ZK**.



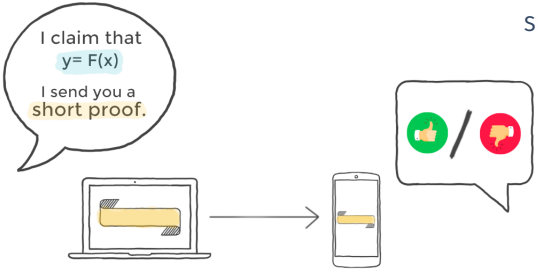
Motivation: (ZK-)SNARKs



SNARKs are "succinct proofs" that enable to check **correctness of a computation**, with/without **ZK**.

SNARKs from IOPs using hash functions [BCS16]:

Motivation: (ZK-)SNARKs

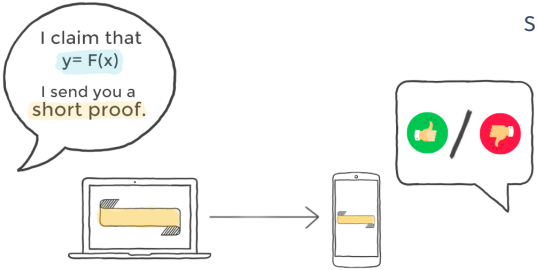


SNARKs are "succinct proofs" that enable to check **correctness of a computation**, with/without **ZK**.

SNARKs from IOPs using hash functions [BCS16]:

- Oracles ← vector commitments (Merkle trees)

Motivation: (ZK-)SNARKs

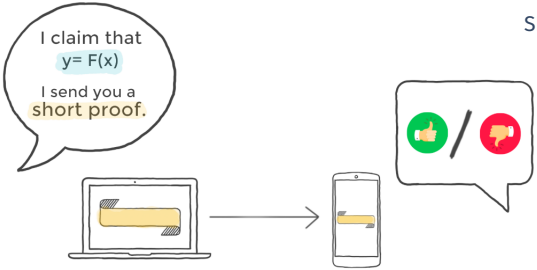


SNARKs are "succinct proofs" that enable to check **correctness of a computation**, with/without **ZK**.

SNARKs from IOPs using hash functions [BCS16]:

- Oracles $\leftarrow$ vector commitments (Merkle trees)
- Non-interactive via $\approx$ Fiat-Shamir transform

Motivation: (ZK-)SNARKs

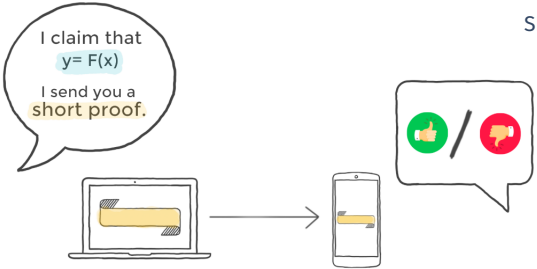


SNARKs are "succinct proofs" that enable to check **correctness of a computation**, with/without **ZK**.

SNARKs from IOPs using hash functions [BCS16]:

- Oracles ← vector commitments (Merkle trees)
- Non-interactive via ≈ Fiat-Shamir transform
- Features: **transparent setup**, **PQ security**

Motivation: (ZK-)SNARKs

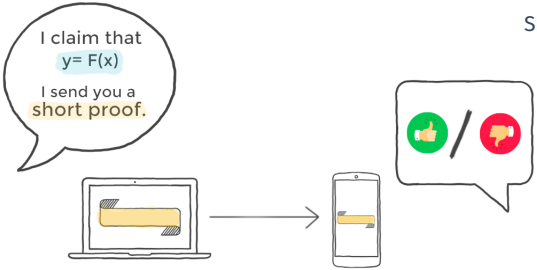


SNARKs are "succinct proofs" that enable to check **correctness of a computation**, with/without **ZK**.

SNARKs from IOPs using hash functions [BCS16]:

- Oracles ← vector commitments (Merkle trees)
- Non-interactive via ≈ Fiat-Shamir transform
- Features: **transparent setup**, **PQ security**
- **Key component = IOP of Proximity** for codes, with
  - good **query/soundness trade-off**
  - **efficient prover**

Motivation: (ZK-)SNARKs



SNARKs are "succinct proofs" that enable to check **correctness of a computation**, with/without **ZK**.

SNARKs from IOPs using hash functions [BCS16]:

- Oracles ← vector commitments (Merkle trees)
- Non-interactive via ≈ Fiat-Shamir transform
- Features: **transparent setup**, **PQ security**
- **Key component = IOP of Proximity** for codes, with
  - good **query/soundness trade-off**
  - **efficient prover**
- Size of SNARK:
  - linear in query complexity of IOP
  - polylog in proof length of IOP

Our results

IOPPs for testing proximity to multivariate polynomial codes.

Length is N, number of variables is m.

Regarding SNARKs applications, constant rate codes ⟶ shorter proofs ($m =$ constant)

	Type	Prover	Verifier	Query	Length	Rounds
[BBHR18]	RS $(m = 1)$	$< 11N$	$< 11 \log N$	$< 2 \log N$	$< N$	$< \log N$
This work	RS$^{\otimes m}$	$< 11N$	$< 11 \log N$	$< 2 \log N$	$< N$	$< \log N$
This work	RM	$< (2m+7)N$	$< 2^m \left(\frac{5}{4} + \frac{7}{m}\right) \log N$	$< \frac{2^m}{m} \log N$	$< \frac{N}{2^m - 1}$	$< \frac{\log N}{m}$

(Complexities counted in $\mathbb{F}$-ops and field elements)

Reed-Solomon IOP of Proximity

Reed-Solomon IOPP

DEF RS code

Given $L \subset \mathbb{F}$, $d < |L|$,
$\mathsf{RS}[L, d] = \left\{ f_{|L} : L \to \mathbb{F} \mid f \in \mathbb{F}[X], \deg f < d \right\}$

The **FRI** protocol is a "**F**ast **R**eed-Solomon **I**OPP" [BBHR18]

Setting:

- L is coset of mult. or add. subgroup of $\mathbb{F}$,
  $|L| =$ power of 2
- $d =$ power of 2

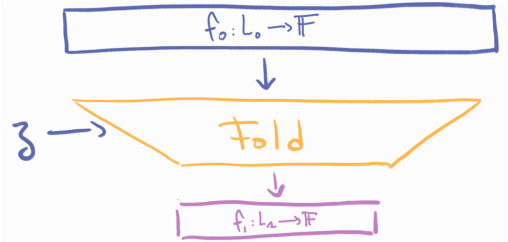
Reed-Solomon IOPP

DEF RS code

Given $L \subset \mathbb{F}$, $d < |L|$,

$\mathrm{RS}[L, d] = \left\{ f_{|L} : L \to \mathbb{F} \mid f \in \mathbb{F}[X], \deg f < d \right\}$

The **FRI** protocol is a "**F**ast **R**eed-Solomon **I**OPP" [BBHR18]



Setting:

- L is coset of mult. or add. subgroup of $\mathbb{F}$,
  $|L| =$ power of 2
- $d =$ power of 2

Recursively halve the size of the problem via "random folding".

Halve the size of the problem

Assume $\mathbb{F}$ has a multiplicative subgroup L of order 2^n.
The square map $q : x \mapsto x^2$ is 2-to-1 from L to $q(L)$.

Halve the size of the problem

Assume $\mathbb{F}$ has a multiplicative subgroup L of order 2^n.
The square map $q : x \mapsto x^2$ is 2-to-1 from L to $q(L)$.

Goal: proximity to RS$[L, d]$ $\longrightarrow$ proximity to RS$[q(L), d/2]$.

Halve the size of the problem

Assume $\mathbb{F}$ has a multiplicative subgroup L of order 2^n.
The square map $q : x \mapsto x^2$ is 2-to-1 from L to $q(L)$.

Goal: proximity to RS$[L, d]$ ⟶ proximity to RS$[q(L), d/2]$.

Given **arbitrary** function $f : L \to \mathbb{F}$,

Halve the size of the problem

Assume $\mathbb{F}$ has a multiplicative subgroup L of order 2^n.
The square map $q : x \mapsto x^2$ is 2-to-1 from L to $q(L)$.

Goal: proximity to RS$[L, d]$ ⟶ proximity to RS$[q(L), d/2]$.

Given **arbitrary** function $f : L \to \mathbb{F}$,

- Decompose f into two parts:

$$f(x) = g_0(x^2) + x g_1(x^2), \text{ with } \deg g_i \leq \frac{\deg f}{2}.$$

Halve the size of the problem

Assume $\mathbb{F}$ has a multiplicative subgroup L of order 2^n.
The square map $q : x \mapsto x^2$ is 2-to-1 from L to $q(L)$.

Goal: proximity to RS$[L, d]$ ⟶ proximity to RS$[q(L), d/2]$.

Given **arbitrary** function $f : L \to \mathbb{F}$,

- Decompose f into two parts:

$$f(x) = g_0(x^2) + x g_1(x^2), \text{ with } \deg g_i \leq \frac{\deg f}{2}.$$

- For $z \in \mathbb{F}$, define **Fold** $[f, z] : q(L) \to \mathbb{F}$ by

$$\textbf{Fold}\,[f, z]\,(y) = g_0(y) + z g_1(y).$$

Halve the size of the problem

Assume $\mathbb{F}$ has a multiplicative subgroup L of order 2^n.
The square map $q : x \mapsto x^2$ is 2-to-1 from L to $q(L)$.

Goal: proximity to RS$[L, d]$ ⟶ proximity to RS$[q(L), d/2]$.

Given **arbitrary** function $f : L \to \mathbb{F}$,

- Decompose f into two parts:

$$f(x) = g_0(x^2) + x g_1(x^2), \text{ with } \deg g_i \leq \frac{\deg f}{2}.$$

- For $z \in \mathbb{F}$, define **Fold** $[f, z] : q(L) \to \mathbb{F}$ by

$$\textbf{Fold}\,[f, z]\,(y) = g_0(y) + z g_1(y).$$

How to compute Fold $[f, z]$?
Any $y \in q(L)$ has 2 distinct square roots $x, -x \in L$.

Halve the size of the problem

Assume $\mathbb{F}$ has a multiplicative subgroup L of order 2^n.
The square map $q : x \mapsto x^2$ is 2-to-1 from L to $q(L)$.

Goal: proximity to RS$[L, d]$ ⟶ proximity to RS$[q(L), d/2]$.

Given **arbitrary** function $f : L \to \mathbb{F}$,

- Decompose f into two parts:

$$f(x) = g_0(x^2) + x g_1(x^2), \text{ with } \deg g_i \leq \frac{\deg f}{2}.$$

- For $z \in \mathbb{F}$, define **Fold** $[f, z] : q(L) \to \mathbb{F}$ by

$$\textbf{Fold}\,[f, z]\,(y) = g_0(y) + z g_1(y).$$

How to compute Fold $[f, z]$?
Any $y \in q(L)$ has 2 distinct square roots $x, -x \in L$.

Linear system $\implies$ $g_0(y) = \frac{f(x)+f(-x)}{2}$ and $g_1(y) = \frac{f(x)-f(-x)}{2x}$.

The 3 key properties

Key properties of folding operators

1. Completeness:

$$f \in \mathsf{RS}[L, d] \implies \textbf{Fold}\,[f, z] \in \mathsf{RS}[q(L), d/2] \qquad \text{for all } z.$$

2. Local computability:

Each entry of **Fold** $[f, z]$ depends on $l = 2$ entries of f.

The 3 key properties

Key properties of folding operators

1. Completeness:

$f \in \mathsf{RS}[L, d] \implies \textbf{Fold}\,[f, z] \in \mathsf{RS}[q(L), d/2]$ for all z.

2. Local computability:

Each entry of **Fold** $[f, z]$ depends on $l = 2$ entries of f.

3. Distance preservation:

f is δ-far from $\mathsf{RS}[L, d] \implies$ **Fold** $[f, z]$ is δ'-far from $\mathsf{RS}[q(L), d/2]$ w.h.p over z $(\delta' \approx \delta)$

IOPP: COMMIT phase

$\mathsf{RS}[L, d = 2^r]$

Honest prover computes:

$\mathcal{P}$

$\mathcal{V}$

$f_0 = f : L \to \mathbb{F}$

$z_0 \xleftarrow{\$} \mathbb{F}$

$f_1 = \textbf{Fold}\,[f_0, z_0]$

f_1

z_1

$f_2 = \textbf{Fold}\,[f_1, z_1]$

f_2

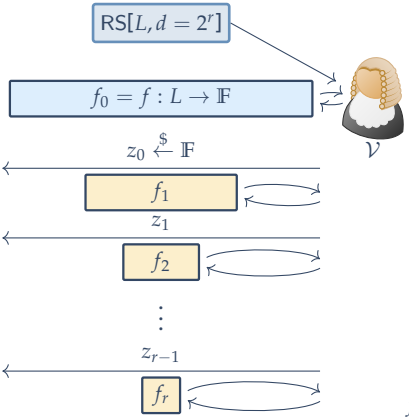
$\vdots$

z_{r-1}

$f_r = \textbf{Fold}\,[f_{r-1}, z_{r-1}] \equiv c \in \mathbb{F}$

f_r

IOPP: QUERY phase



Global consistency test:

Sample $s \in L$ and check

$f_1(s^2) \stackrel{?}{=} \textbf{Fold}\,[f_0, z_0]\,(s^2)$

$f_2(s^4) \stackrel{?}{=} \textbf{Fold}\,[f_1, z_1]\,(s^4)$

$\vdots$

$f_r(s^{2^r}) \stackrel{?}{=} \textbf{Fold}\,[f_{r-1}, z_{r-1}]\,(s^{2^r})$

Final test: $f_r \stackrel{?}{=} c \in \mathbb{F}$

IOPP

$\mathsf{RS}[L, d = 2^r]$

Honest prover computes:

$\mathcal{P}$

$f_0 = f : L \to \mathbb{F}$

$\mathcal{V}$

Global consistency test:

$z_0 \xleftarrow{\$} \mathbb{F}$

$f_1 = \textbf{Fold}\,[f_0, z_0]$ — f_1 — Sample $s \in L$ and check

$f_1(s^2) \stackrel{?}{=} \textbf{Fold}\,[f_0, z_0]\,(s^2)$

z_1

$f_2 = \textbf{Fold}\,[f_1, z_1]$ — f_2 — $f_2(s^4) \stackrel{?}{=} \textbf{Fold}\,[f_1, z_1]\,(s^4)$

$\vdots$ $\vdots$ $\vdots$

z_{r-1}

$f_r = \textbf{Fold}\,[f_{r-1}, z_{r-1}] \equiv c \in \mathbb{F}$ — f_r — $f_r(s^{2^r}) \stackrel{?}{=} \textbf{Fold}\,[f_{r-1}, z_{r-1}]\,(s^{2^r})$

Final test: $f_r \stackrel{?}{=} c \in \mathbb{F}$

Completeness: If $f \in C$, Verifier accepts.

Soundness: $\exists \delta_0 \in (0,1), \forall \delta < \delta_0, \Pr[\text{ accept } \mid f \text{ is } \delta\text{-far}] \approx (1-\delta)$ (assuming $|\mathbb{F}|$ is large enough.)

IOPP

$\mathsf{RS}[L, d = 2^r]$

Honest prover computes:

$\mathcal{P}$

$f_0 = f : L \to \mathbb{F}$

$\mathcal{V}$

Global consistency test:

$z_0 \xleftarrow{\$} \mathbb{F}$

$f_1 = \textbf{Fold}\,[f_0, z_0]$ — f_1 — Sample $s \in L$ and check

$f_1(s^2) \stackrel{?}{=} \textbf{Fold}\,[f_0, z_0]\,(s^2)$

z_1

$f_2 = \textbf{Fold}\,[f_1, z_1]$ — f_2 — $f_2(s^4) \stackrel{?}{=} \textbf{Fold}\,[f_1, z_1]\,(s^4)$

$\vdots$ $\vdots$ $\vdots$

z_{r-1}

$f_r = \textbf{Fold}\,[f_{r-1}, z_{r-1}] \equiv c \in \mathbb{F}$ — f_r — $f_r(s^{2^r}) \stackrel{?}{=} \textbf{Fold}\,[f_{r-1}, z_{r-1}]\,(s^{2^r})$

Final test: $f_r \stackrel{?}{=} c \in \mathbb{F}$

Completeness: If $f \in C$, Verifier accepts.

Soundness: $\exists \delta_0 \in (0,1), \forall \delta < \delta_0, \Pr[\text{ accept } \mid f \text{ is } \delta\text{-far}] \approx (1-\delta)^t$. (assuming $|\mathbb{F}|$ is large enough.)

How to fold multivariate polynomials

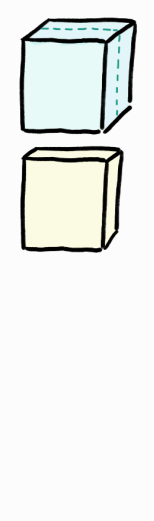
Folding tensor product of RS codes

The tensor structure enables to fold along **one variable at a time**.

Folding tensor product of RS codes

The tensor structure enables to fold along **one variable at a time**.

- $\mathsf{RS}^{\otimes m} \longrightarrow \mathsf{RS}^{\otimes m-1}$:

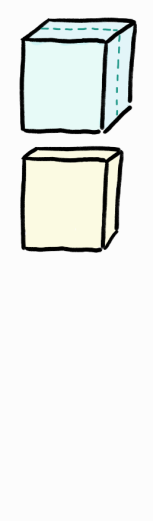


Folding tensor product of RS codes

The tensor structure enables to fold along **one variable at a time**.

- $\mathrm{RS}^{\otimes m} \to \mathrm{RS}^{\otimes m-1}$:
  - Write $f : \prod_{i=1}^{m} L_i \to \mathbb{F}$ as

$$f(x_1, x_2, \ldots, x_m) = g_0(x_1^2, x_2, \ldots, x_m) + x_1 g_1(x_1^2, x_2, \ldots, x_m)$$



Folding tensor product of RS codes

The tensor structure enables to fold along **one variable at a time**.

- $\mathsf{RS}^{\otimes m} \longrightarrow \mathsf{RS}^{\otimes m-1}$: $(q : x \mapsto x^2)$
  - Write $f : \prod_{i=1}^{m} L_i \to \mathbb{F}$ as

$$f(x_1, x_2, \ldots, x_m) = g_0(x_1^2, x_2, \ldots, x_m) + x_1 g_1(x_1^2, x_2, \ldots, x_m)$$

  - For $z \in \mathbb{F}$, define **Fold** $[f, z] : q(L_1) \times \prod_{i=2}^{m} L_i \to \mathbb{F}$ by

$$\textbf{Fold}\,[f, z]\,(y, \boldsymbol{x}) = g_0(y, \boldsymbol{x}) + z g_1(y, \boldsymbol{x})$$

Folding tensor product of RS codes

The tensor structure enables to fold along **one variable at a time**.

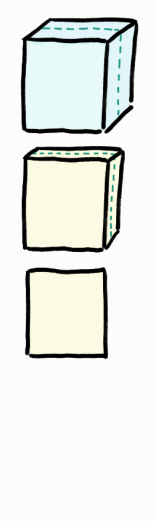
- $\mathsf{RS}^{\otimes m} \longrightarrow \mathsf{RS}^{\otimes m-1}$: $(q : x \mapsto x^2)$
  - Write $f : \prod_{i=1}^{m} L_i \to \mathbb{F}$ as

$$f(x_1, x_2, \ldots, x_m) = g_0(x_1^2, x_2, \ldots, x_m) + x_1 g_1(x_1^2, x_2, \ldots, x_m)$$

  - For $z \in \mathbb{F}$, define **Fold** $[f, z] : q(L_1) \times \prod_{i=2}^{m} L_i \to \mathbb{F}$ by

$$\textbf{Fold}\,[f, z]\,(y, \boldsymbol{x}) = g_0(y, \boldsymbol{x}) + z g_1(y, \boldsymbol{x})$$

  - After $\log d$ rounds, expected x_1-degree $= 0$



Folding tensor product of RS codes

The tensor structure enables to fold along **one variable at a time**.

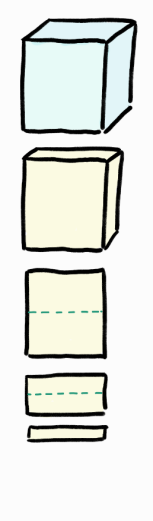
- $\mathsf{RS}^{\otimes m} \longrightarrow \mathsf{RS}^{\otimes m-1}$: $\quad (q : x \mapsto x^2)$
  - Write $f : \prod_{i=1}^{m} L_i \to \mathbb{F}$ as

$$f(x_1, x_2, \ldots, x_m) = g_0(x_1^2, x_2, \ldots, x_m) + x_1 g_1(x_1^2, x_2, \ldots, x_m)$$

  - For $z \in \mathbb{F}$, define **Fold** $[f, z] : q(L_1) \times \prod_{i=2}^{m} L_i \to \mathbb{F}$ by

$$\textbf{Fold}\,[f, z]\,(y, \boldsymbol{x}) = g_0(y, \boldsymbol{x}) + z g_1(y, \boldsymbol{x})$$

  - After $\log d$ rounds, expected x_1-degree $= 0$
- $\mathsf{RS}^{\otimes m-1} \longrightarrow \mathsf{RS}^{\otimes m-2}$, starting with $f : \prod_{i=2}^{m} L_i \to \mathbb{F}$



Folding tensor product of RS codes

The tensor structure enables to fold along **one variable at a time**.

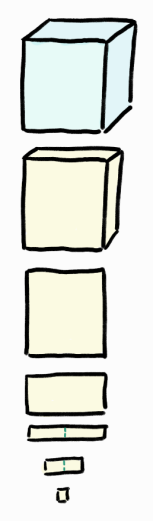
- $\mathsf{RS}^{\otimes m} \longrightarrow \mathsf{RS}^{\otimes m-1}$: $(q : x \mapsto x^2)$
  - Write $f : \prod_{i=1}^{m} L_i \to \mathbb{F}$ as

$$f(x_1, x_2, \ldots, x_m) = g_0(x_1^2, x_2, \ldots, x_m) + x_1 g_1(x_1^2, x_2, \ldots, x_m)$$

  - For $z \in \mathbb{F}$, define **Fold** $[f, z] : q(L_1) \times \prod_{i=2}^{m} L_i \to \mathbb{F}$ by

$$\textbf{Fold}\,[f, z]\,(y, \boldsymbol{x}) = g_0(y, \boldsymbol{x}) + z g_1(y, \boldsymbol{x})$$

  - After $\log d$ rounds, expected x_1-degree $= 0$
- $\mathsf{RS}^{\otimes m-1} \longrightarrow \mathsf{RS}^{\otimes m-2}$, starting with $f : \prod_{i=2}^{m} L_i \to \mathbb{F}$
- ... $\longrightarrow$ RS code of dimension 1



Folding tensor product of RS codes

The tensor structure enables to fold along **one variable at a time**.

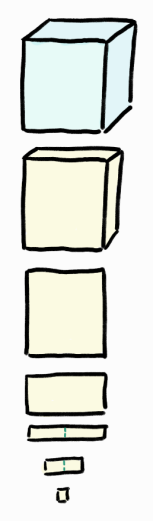
- $\mathrm{RS}^{\otimes m} \longrightarrow \mathrm{RS}^{\otimes m-1}$: $(q : x \mapsto x^2)$
  - Write $f : \prod_{i=1}^{m} L_i \to \mathbb{F}$ as

$$f(x_1, x_2, \ldots, x_m) = g_0(x_1^2, x_2, \ldots, x_m) + x_1 g_1(x_1^2, x_2, \ldots, x_m)$$

  - For $z \in \mathbb{F}$, define **Fold** $[f, z] : q(L_1) \times \prod_{i=2}^{m} L_i \to \mathbb{F}$ by

$$\textbf{Fold}\,[f, z]\,(y, \boldsymbol{x}) = g_0(y, \boldsymbol{x}) + z g_1(y, \boldsymbol{x})$$

  - After $\log d$ rounds, expected x_1-degree $= 0$
- $\mathrm{RS}^{\otimes m-1} \longrightarrow \mathrm{RS}^{\otimes m-2}$, starting with $f : \prod_{i=2}^{m} L_i \to \mathbb{F}$
- ... $\longrightarrow$ RS code of dimension 1



☑ **Completeness** ☑ **Local computability** ☑ **Distance preservation**

Folding Reed-Muller code

In the total degree case, we fold along **every variable at the same time**.

Folding Reed-Muller code

In the total degree case, we fold along **every variable at the same time**.

Folding Reed-Muller code

In the total degree case, we fold along **every variable at the same time**.

Multivariate decomposition

Let $f(\boldsymbol{X}) \in \mathbb{F}[X_1, \ldots, X_m]$.
There is a unique sequence of polynomials $(g_{\boldsymbol{u}})_{\boldsymbol{u} \in \{0,1\}^m}$ such that

$$f(\boldsymbol{X}) = \sum_{\boldsymbol{u} \in \{0,1\}^m} \boldsymbol{X}^{\boldsymbol{u}} g_{\boldsymbol{u}}(X_1^2, \ldots, X_m^2), \qquad \deg g_{\boldsymbol{u}} \leq \left\lfloor \frac{\deg f - w_{\mathsf{H}}(\boldsymbol{u})}{2} \right\rfloor$$

Folding Reed-Muller code

In the total degree case, we fold along **every variable at the same time**.

Multivariate decomposition

Let $f(\boldsymbol{X}) \in \mathbb{F}[X_1, \ldots, X_m]$.
There is a unique sequence of polynomials $(g_{\boldsymbol{u}})_{\boldsymbol{u} \in \{0,1\}^m}$ such that

$$f(\boldsymbol{X}) = \sum_{\boldsymbol{u} \in \{0,1\}^m} \boldsymbol{X}^{\boldsymbol{u}} g_{\boldsymbol{u}}(X_1^2, \ldots, X_m^2), \qquad \deg g_{\boldsymbol{u}} \leq \left\lfloor \frac{\deg f - w_{\mathsf{H}}(\boldsymbol{u})}{2} \right\rfloor$$

The folding of $f : L^m \to \mathbb{F}$ is a function **Fold** $[f, \boldsymbol{z}] : q(L)^m \to \mathbb{F}$ defined using the $g_{\boldsymbol{u}}$'s. $(q : x \mapsto x^2)$

Reduce $\mathsf{RM}[L, d, m] \longrightarrow \mathsf{RM}[q(L), d/2, m]$.

Subtlety: need to be careful about the distinct degree bounds on the $g_{\boldsymbol{u}}$'s.

Folding Reed-Muller code

In the total degree case, we fold along **every variable at the same time**.

Multivariate decomposition

Let $f(\boldsymbol{X}) \in \mathbb{F}[X_1, \ldots, X_m]$.
There is a unique sequence of polynomials $(g_{\boldsymbol{u}})_{\boldsymbol{u} \in \{0,1\}^m}$ such that

$$f(\boldsymbol{X}) = \sum_{\boldsymbol{u} \in \{0,1\}^m} \boldsymbol{X}^{\boldsymbol{u}} g_{\boldsymbol{u}}(X_1^2, \ldots, X_m^2), \qquad \deg g_{\boldsymbol{u}} \leq \left\lfloor \frac{\deg f - w_{\mathsf{H}}(\boldsymbol{u})}{2} \right\rfloor$$

The folding of $f : L^m \to \mathbb{F}$ is a function **Fold** $[f, \boldsymbol{z}] : q(L)^m \to \mathbb{F}$ defined using the $g_{\boldsymbol{u}}$'s. $(q : x \mapsto x^2)$

Reduce $\mathsf{RM}[L, d, m] \rightarrow \mathsf{RM}[q(L), d/2, m]$.

Subtlety: need to be careful about the distinct degree bounds on the $g_{\boldsymbol{u}}$'s.

- ☑ **Completeness**
- ☑ **Local computability** (with $l = 2^m$)
- ☑ **Distance preservation**

Conclusion

What we have:

- Inspired by FRI protocol for RS-IOPP,
  we give concrete IOPPs for $\mathsf{RS}^{\otimes m}$ and RM codes with **similar parameters**.

THEOREM [Augot-Bordage-Nardi'21]

$\mathsf{RS}[L, d]^{\otimes m}$ has an IOPP $(\mathcal{P}, \mathcal{V})$ satisfying

$$\begin{bmatrix} \text{\# rounds} & = m \log d \\ \text{\# queries} & = 2m \log d + 1 \\ \text{prover time} & \leq 11|L^m| \\ \text{verifier time} & \leq 11 m \log d \\ \text{proof length} & < |L^m| \end{bmatrix}$$

THEOREM [Augot-Bordage-Nardi'21]

$\mathsf{RM}[L, d, m]$ has an IOPP $(\mathcal{P}, \mathcal{V})$ satisfying

$$\begin{bmatrix} \text{\# rounds} & = \log d \\ \text{\# queries} & = 2^m \log d + 1 \\ \text{prover time} & < (2m+7)|L^m| \\ \text{verifier time} & < 2^m(\frac{5}{4}m + 7)(\log |L|) \\ \text{proof length} & < |L^m|/(2^m - 1) \end{bmatrix}$$

Conclusion

What we have:

- Inspired by FRI protocol for RS-IOPP,
  we give concrete IOPPs for $\mathrm{RS}^{\otimes m}$ and RM codes with **similar parameters**.
- Folding-based IOPP framework$\longleftrightarrow$ definition of **sequence of codes with folding operators**

Conclusion

What we have:

- Inspired by FRI protocol for RS-IOPP,
  we give concrete IOPPs for $\mathrm{RS}^{\otimes m}$ and RM codes with **similar parameters**.
- Folding-based IOPP framework$\longleftrightarrow$ definition of **sequence of codes with folding operators**
  - also yields to IOPP for AG codes (next talk)

Conclusion

What we have:

- Inspired by FRI protocol for RS-IOPP,
  we give concrete IOPPs for $\mathrm{RS}^{\otimes m}$ and RM codes with **similar parameters**.
- Folding-based IOPP framework$\longleftrightarrow$ definition of **sequence of codes with folding operators**
  - also yields to IOPP for AG codes (next talk)

Conclusion

What we have:

- Inspired by FRI protocol for RS-IOPP,
  we give concrete IOPPs for $\mathrm{RS}^{\otimes m}$ and RM codes with **similar parameters**.
- Folding-based IOPP framework$\longleftrightarrow$ definition of **sequence of codes with folding operators**
  - also yields to IOPP for AG codes (next talk)

Future directions: narrow the gap between theoretically feasible and **practical**

- Practical IOPP with sublogarithmic query complexity? (in theory, $O(1)$ queries)
- Practical IOPP for $C^{\otimes m}$ where C is a generic linear code?

Conclusion

What we have:

- Inspired by FRI protocol for RS-IOPP,
  we give concrete IOPPs for $\mathrm{RS}^{\otimes m}$ and RM codes with **similar parameters**.
- Folding-based IOPP framework$\longleftrightarrow$ definition of **sequence of codes with folding operators**
  - also yields to IOPP for AG codes (next talk)

Future directions: narrow the gap between theoretically feasible and **practical**

- Practical IOPP with sublogarithmic query complexity? (in theory, $O(1)$ queries)
- Practical IOPP for $C^{\otimes m}$ where C is a generic linear code?

Conclusion

What we have:

- Inspired by FRI protocol for RS-IOPP,
  we give concrete IOPPs for $\mathrm{RS}^{\otimes m}$ and RM codes with **similar parameters**.
- Folding-based IOPP framework$\longleftrightarrow$ definition of **sequence of codes with folding operators**
  - also yields to IOPP for AG codes (next talk)

Future directions: narrow the gap between theoretically feasible and **practical**

- Practical IOPP with sublogarithmic query complexity? (in theory, $O(1)$ queries)
- Practical IOPP for $C^{\otimes m}$ where C is a generic linear code?

Thank you!