Reconstruction of non-degenerate homogeneous depth-3 circuits

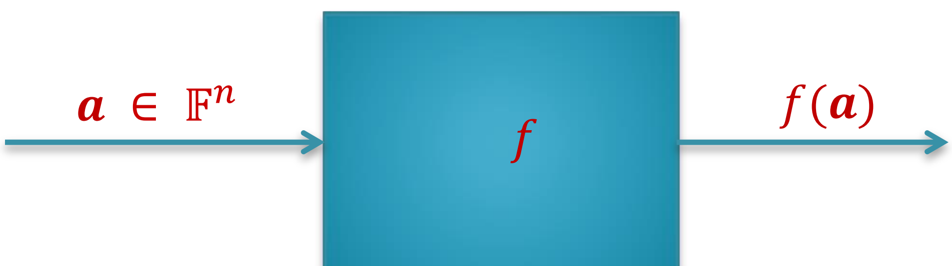
Chandan Saha

Joint work with Neeraj Kayal

Reconstruction of circuits

- Let $f(\boldsymbol{x})$ be a n-variate degree-d polynomial computed by a circuit of size s from a class C.

- Reconstruction problem. Given black-box access to f, output a *small* circuit computing f.



Black-box access to f
(membership query access to f)

Reconstruction of circuits

- Let $f(\boldsymbol{x})$ be a n-variate degree-d polynomial computed by a circuit of size s from a class C.

- Reconstruction problem. Given black-box access to f, output a *small* circuit computing f.

- Size of the output circuit. Ideally, $poly(s)$.
- Proper learning. Output a circuit from class C.

Reconstruction of circuits

- Let $f(\mathbf{x})$ be a n-variate degree-d polynomial computed by a circuit of size s from a class $\mathcal{C}$.

- Reconstruction problem. Given black-box access to f, output a *small* circuit computing f.

- Efficiency. Ideally, $poly(d, s)$. But, even $N = \binom{n+d}{n}$ time reconstruction is non-trivial, for $n \ll s \ll N$, as exhaustive search over size-s circuits takes $\exp(s)$ time.

Reconstruction implies lower bounds

- Fortnow & Klivans (2009): A randomized poly-time reconstruction algorithm for $\mathcal{C}$ implies there's a function in BPEXP that does not have poly-size circuits from $\mathcal{C}$.

- Volkovich (2016): A deterministic poly-time recon. algorithm for $\mathcal{C}$ can be used to construct a function in EXP that doesn't have poly-size circuits from $\mathcal{C}$.

Reconstruction implies lower bounds

- Fortnow & Klivans (2009): A randomized poly-time reconstruction algorithm for $\mathcal{C}$ implies there's a function in BPEXP that does not have poly-size circuits from $\mathcal{C}$.

- Volkovich (2016): A deterministic poly-time recon. algorithm for $\mathcal{C}$ can be used to construct a function in EXP that doesn't have poly-size circuits from $\mathcal{C}$.

- Efficient reconstruction algorithms have focussed on classes $\mathcal{C}$ for which non-trivial lower bounds are known.

Reconstruction implies lower bounds

- Fortnow & Klivans (2009): A randomized poly-time reconstruction algorithm for $\mathcal{C}$ implies there's a function in BPEXP that does not have poly-size circuits from $\mathcal{C}$.

- Volkovich (2016): A deterministic poly-time recon. algorithm for $\mathcal{C}$ can be used to construct a function in EXP that doesn't have poly-size circuits from $\mathcal{C}$.

- Efficient reconstruction algorithms have focussed on classes $\mathcal{C}$ for which non-trivial lower bounds are known.

Does lower bound imply efficient reconstruction ?

Reconstruction is inherently hard

- Reconstruction is akin to <u>approximating the minimum circuit size</u>.

- Minimum Circuit Size Problem (MCSP). Given a truth-table T of size $N = 2^n$ and an integer s, check if the function defined by T has a circuit of size at most s.

Reconstruction is inherently hard

- Reconstruction is akin to approximating the minimum circuit size.

- Minimum Circuit Size Problem (MCSP). Given a truth-table T of size $N = 2^n$ and an integer s, check if the function defined by T has a circuit of size at most s.

- Allender & Hirahara (2017): There's a $\epsilon(N) = o(1)$ such that approximating the minimum circuit size to within $N^{1-\epsilon}$ factor cannot be done in $poly(N)$ time, assuming the existence of one-way function.

Reconstruction is inherently hard

- Reconstruction is akin to approximating the minimum circuit size.

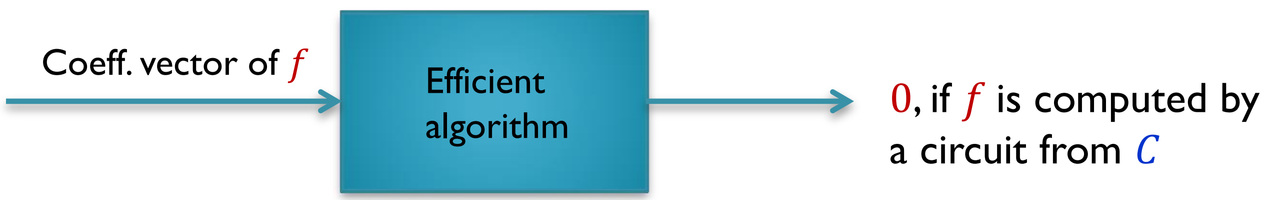
- Minimum Circuit Size Problem (MCSP). Given a truth-table T of size $N = 2^n$ and an integer s, check if the function defined by T has a circuit of size at most s.

- Drawing analogy between Boolean and arithmetic circuits, reconstruction is expected to be a hard problem even if f is given verbosely as a list of $N = \binom{n+d}{n}$ coefficients.

Natural lower bound to reconstruction?

- Razborov & Rudich (1997); Forbes, Shpilka & Volk (2017); Grochow, Kumar, Saks & Saraf (2017):

Constructivity.



Separator for C

Natural lower bound to PAC learning

- Carmosino, Impagliazzo, Kabanets & Kolokolova (2016): The natural lower bound framework for $AC^0[p]$ circuits can be used to give quasi-polynomial time PAC learning algorithm for the same class.

- Linial, Mansour & Nisan (1993): Similar result for AC^0 .

- Jackson, Klivans & Servedio (2002): Similar result for AC^0 with poly-logarithmic majority gates.

Natural lower bound to PAC learning

- Carmosino, Impagliazzo, Kabanets & Kolokolova (2016): The natural lower bound framework for $AC^0[p]$ circuits can be used to give quasi-polynomial time PAC learning algorithm for the same class.

- Linial, Mansour & Nisan (1993): Similar result for AC^0 .

- Jackson, Klivans & Servedio (2002): Similar result for AC^0 with poly-logarithmic majority gates.

- These learning algorithms are not proper.

Natural lower bound to reconstruction?

- Can we hope to get such natural lower bound to reconstruction translations for arithmetic circuits?

Natural lower bound to reconstruction?

- Can we hope to get such natural lower bound to reconstruction translations for arithmetic circuits?

- Klivans & Shpilka (2006); Forbes & Shpilka (2013): Gave efficient reconstruction for <u>read-once oblivious ABP</u> (ROABP) and <u>non-commutative ABP</u>. (Natural lower bounds were known for these models.)

Natural lower bound to reconstruction?

- Can we hope to get such natural lower bound to reconstruction translations for arithmetic circuits?

- There are a few challenges:
  - Exact learning. Two polynomials differ at many points. If the output is an arithmetic circuit then it has to compute f exactly.

Natural lower bound to reconstruction?

- Can we hope to get such natural lower bound to reconstruction translations for arithmetic circuits?

- There are a few challenges:
  - Exact learning. Two polynomials differ at many points. If the output is an arithmetic circuit then it has to compute f exactly.
  - Depth reduction. Constant depth arithmetic circuits are too powerful.
  - Homogenization. Makes reconstruction challenging even for classes with exponential lower bounds.

Homogeneous depth-3 circuits

$$f = T_1 + T_2 + \cdots + T_s$$

- Each term T_i is a product of d linear forms in n variables.

Homogeneous depth-3 circuits

$$f \;=\; T_1 + T_2 + \cdots + T_s \longleftarrow$$ A (n, d, s) homogeneous depth-3 circuit

- Each term T_i is a product of d linear forms in n variables.

Homogeneous depth-3 circuits

$$f \;=\; T_1 + T_2 + \cdots + T_s$$

- Each term T_i is a product of d linear forms in n variables.
- Nisan & Wigderson (1997): Showed a $(n/d)^{\Omega(d)}$ lower bound on s for $d \leq n$.
- Kayal, S., Tavenas (2016): Showed a $2^{\Omega(n)}$ lower bound on s for $d \geq n$.

Homogeneous depth-3 circuits

$$f \;=\; T_1 + T_2 + \cdots + T_s$$

- Each term T_i is a product of d linear forms in n variables.
- Nisan & Wigderson (1997): Showed a $(n/d)^{\Omega(d)}$ lower bound on s for $d \leq n$.
- Kayal, S., Tavenas (2016): Showed a $2^{\Omega(n)}$ lower bound on s for $d \geq n$.

Both the lower bound proofs are natural.

Homogeneous depth-3 circuits

$$f = T_1 + T_2 + \cdots + T_s$$

- Each term T_i is a product of d linear forms in n variables.
- Klivans & Shpilka (2003): Can we reconstruct homogeneous depth-3 circuits efficiently ?

Our result

$$f \quad = \; T_1 + T_2 + \cdots + T_s$$

- Theorem. Let $n \geq (3d)^2$ and $s \leq (n/3d)^{d/3}$. There is a randomized $poly(n, s)$ time algorithm to reconstruct ***non-degenerate*** (n, d, s) homogeneous depth-3 circuits.

Our result

$$f \;=\; T_1 + T_2 + \cdots + T_s$$

- Theorem. Let $n \geq (3d)^2$ and $s \leq (n/3d)^{d/3}$. There is a randomized $poly(n, s)$ time algorithm to reconstruct *non-degenerate* (n, d, s) homogeneous depth-3 circuits.
- Proper learning. The output is a (n, d, s) homogeneous depth-3 circuit.

Our result

$$f \;=\; T_1 + T_2 + \cdots + T_s$$

- Theorem. Let $n \geq (3d)^2$ and $s \leq (n/3d)^{d/3}$. There is a randomized $poly(n, s)$ time algorithm to reconstruct *non-degenerate* (n, d, s) homogeneous depth-3 circuits.

- The algorithm works under two restrictions:
  - Degree restriction: $n \geq (3d)^2$
  - Non-degeneracy: Next slide…

Our result

$$f \;\; = T_1 + T_2 + \cdots + T_s$$

- Theorem. Let $n \geq (3d)^2$ and $s \leq (n/3d)^{d/3}$. There is a randomized $poly(n, s)$ time algorithm to reconstruct *non-degenerate* (n, d, s) homogeneous depth-3 circuits.

- The algorithm works under two restrictions:
  - Degree restriction: $n \geq (3d)^2$ Let's ignore it!
  - Non-degeneracy: Next slide…

Non-degeneracy condition

$$f \ = \ T_1 + T_2 + \cdots + T_s$$

- Let $k = \frac{\log s}{\log(n/ed)}$, $U := \langle \partial^k f \rangle$ and $U_i := \langle \partial^k \, T_i \rangle$.
- Clearly, $U \subseteq \ U_1 + U_2 + \cdots + U_s$.

- Non-degeneracy*: $U \ = \ U_1 \oplus U_2 \oplus \cdots \oplus U_s$

Non-degeneracy condition

$$f \;=\; T_1 + T_2 + \cdots + T_s$$

- Let $k = \frac{\log s}{\log(n/ed)}$, $U \coloneqq \langle \partial^k f \rangle$ and $U_i \coloneqq \langle \partial^k T_i \rangle$.
- Clearly, $U \subseteq U_1 + U_2 + \cdots + U_s$.
- Non-degeneracy*: $U \underset{\text{equality}}{=} \underbrace{U_1 \oplus U_2 \oplus \cdots \oplus U_s}_{\text{direct sum}}$

Non-degeneracy condition

$$f \;=\; T_1 + T_2 + \cdots + T_s$$

$k \;=\; O(1)$ if $s = poly(n)$

- Let $k = \dfrac{\log s}{\log(n/ed)}$, $U \coloneqq \langle \partial^k f \rangle$ and $U_i \coloneqq \langle \partial^k T_i \rangle$.
- Clearly, $U \subseteq U_1 + U_2 + \cdots + U_s$.

- Non-degeneracy*: $U \;=\; \underbrace{U_1 \oplus U_2 \oplus \cdots \oplus U_s}_{\text{direct sum}}$

equality

Non-degeneracy condition

$$f \;=\; T_1 + T_2 + \cdots + T_s$$

- Let $k = \frac{\log s}{\log(n/ed)}$, $U \coloneqq \langle \partial^k f \rangle$ and $U_i \coloneqq \langle \partial^k T_i \rangle$.
- Clearly, $U \subseteq U_1 + U_2 + \cdots + U_s$.
- Non-degeneracy*: $U = U_1 \oplus U_2 \oplus \cdots \oplus U_s$
- A random homogeneous depth-3 circuit is almost surely non-degenerate.

Non-degeneracy condition

- Can we get rid of non-degeneracy condition entirely? If yes, then…

Non-degeneracy condition

- Can we get rid of non-degeneracy condition entirely? If yes, then…
  - Lower bound for depth-3 circuits: (homogenization) If $f(\boldsymbol{x})$ is computed by a (n, d, s) depth-3 circuit then $z^d f(\boldsymbol{x}/z)$ is computed by $(n+1, d, s)$ homogeneous depth-3 circuit. Thus, we get efficient reconstruction for depth-3 circuits, and [FK09] implies a lower bound for the same class!

Non-degeneracy condition

- Can we get rid of non-degeneracy condition entirely? If yes, then…

  - Reconstruction for general circuits: (depth reduction)

    We get $n^{O(\sqrt{d})}$ time reconstruction for circuits of size $poly(n)$ via the depth reduction to depth-3 result. [Gupta, Kamath, Kayal, Saptharishi (2013); Tavenas (2013); Koiran (2012); Agrawal & Vinay (2008)]

Non-degeneracy condition

- Thus, getting an unconditional translation from natural lower bound proofs to efficient reconstruction seems extremely challenging even for homogeneous depth-3 circuits.

Non-degeneracy condition

- Thus, getting an unconditional translation from natural lower bound proofs to efficient reconstruction seems extremely challenging even for homogeneous depth-3 circuits.

- However, it may be possible to use the natural lower bound framework of a model to do efficient reconstruction for the same model under some *non-degeneracy condition* that originates from the lower bound proof.

Non-degeneracy condition

$$f \;=\; T_1 + T_2 + \cdots + T_s$$

- Let $k = \frac{\log s}{\log(n/ed)}$, $U \coloneqq \langle \partial^k f \rangle$ and $U_i \coloneqq \langle \partial^k \, T_i \rangle$.
- Non-degeneracy*: $U \;=\; U_1 \oplus U_2 \oplus \cdots \oplus U_s$
- Fact: A crucial aspect of the [NW95] lower bound proof is that each U_i is "simple" in the sense that it is a low-dimensional space.

Non-degeneracy condition

$$f \;=\; T_1 + T_2 + \cdots + T_s$$

- Let $k = \frac{\log s}{\log(n/ed)}$, $U := \langle \partial^k f \rangle$ and $U_i := \langle \partial^k T_i \rangle$.

- Non-degeneracy*: $U = U_1 \oplus U_2 \oplus \cdots \oplus U_s$

- The non-degeneracy condition exploits this fact and reduces the reconstruction problem to <u>decomposing the space</u> U into a direct sum of "simple" spaces.

Non-degeneracy condition

$$f \;=\; T_1 + T_2 + \cdots + T_s$$

- Let $k = \frac{\log s}{\log(n/ed)}$, $U \coloneqq \langle \partial^k f \rangle$ and $U_i \coloneqq \langle \partial^k \, T_i \rangle$.

- Non-degeneracy*: $U = U_1 \oplus U_2 \oplus \cdots \oplus U_s$

- The non-degeneracy condition exploits this fact and reduces the reconstruction problem to decomposing the space U into a direct sum of "simple" spaces.

A priori, it is not clear if this decomposition can be done efficiently.

Conceptual contribution

- A paradigm for handling large fan-in sum gates.

  - *Step* 1: Reduce the problem of finding children of a sum gate to decomposition of a suitable space U into "simpler" spaces (using the lower bound framework).

  - *Step* 2: Define an appropriate space $\mathcal{S}$ of linear operators on U. The structure of $\mathcal{S}$ (in our case, the *irreducible invariant subspaces* of U induced by $\mathcal{S}$) helps retrieve the "simpler" spaces efficiently.

Conceptual contribution

- A paradigm for handling <u>large fan-in sum gates</u>.

- We feel that this paradigm has the potential to give efficient reconstruction for other circuit models for which natural lower bounds are known.

Conceptual contribution

- A paradigm for handling large fan-in sum gates.

- We feel that this paradigm has the potential to give efficient reconstruction for other circuit models for which natural lower bounds are known.

- Prior work on efficient reconstruction (barring those on ROABP / non-commutative ABP / read-once formula) could only handle very low fan-in sum gates.

Related results

Restricted depth-3 circuits

- Beimel, Bergadano, Bshouty, Kushilevitz & Varricchio (2000): Randomized $poly(n, d, s)$ time reconstruction for depth-3 powering circuits and set-multilinear depth-3 circuits.

- Klivans & Shpilka (2003): Randomized $poly(n, 2^d, s)$ time reconstruction for general depth-3 circuits.

- The output hypothesis is an ROABP.

Restricted depth-3 circuits

- Shpilka (2007): Randomized $qpoly(n, d, |\mathbb{F}|)$ time reconstruction for depth-3 circuits with top fan-in two.
- Karnin & Shpilka (2009): Deterministic reconstruction for depth-3 circuits in $poly(n). |\mathbb{F}|^{(\log d)^{O(s^3)}}$ time.
- Sinha (2016): Randomized reconstruction for depth-3 circuits with top fan-in two over $\mathbb{R}$ in $poly(n, d)$ time.

Restricted depth-3 circuits

- Shpilka (2007): Randomized $qpoly(n, d, |\mathbb{F}|)$ time reconstruction for depth-3 circuits with top fan-in two.
- Karnin & Shpilka (2009): Deterministic reconstruction for depth-3 circuits in $poly(n). |\mathbb{F}|^{(\log d)^{O(s^3)}}$ time.
- Sinha (2016): Randomized reconstruction for depth-3 circuits with top fan-in two over $\mathbb{R}$ in $poly(n, d)$ time.
- These learning algorithms are proper*.

Restricted depth-4 circuits

- Gupta, Kayal & Lokam (2012): Randomized $poly(s)$ time reconstruction for size s <u>multilinear depth-4 circuits with top fan-in two</u>.

- This learning is also proper.

Reconstruction under non-degeneracy

- Kayal (2012): Randomized $poly(n, s^{\log_d s})$ time reconstruction for depth-3 powering circuits.
- García-Marco, Koiran & Pecatte (2018): Randomized $poly(n, s)$ time reconstruction for depth-3 powering circuits for $s \leq \binom{n+1}{2}$ and $d \geq 5$.

Reconstruction under non-degeneracy

- Kayal (2012): Randomized $poly(n, s^{\log_d s})$ time reconstruction for depth-3 powering circuits.
- García-Marco, Koiran & Pecatte (2018): Randomized $poly(n, s)$ time reconstruction for depth-3 powering circuits for $s \leq \binom{n+1}{2}$ and $d \geq 5$.
- Gupta, Kayal & Qiao (2013): Randomized $poly(n, s)$ time reconstruction for fan-in two regular formulas.
- Kayal, Nair & S. (2018): Randomized $poly(n, d)$ time reconstruction for constant width homogeneous ABP.

Reconstruction under non-degeneracy

- Kayal (2012): Randomized $poly(n, s^{\log_d s})$ time reconstruction for depth-3 powering circuits.
- García-Marco, Koiran & Pecatte (2018): Randomized $poly(n, s)$ time reconstruction for depth-3 powering circuits for $s \leq \binom{n+1}{2}$ and $d \geq 5$.
- Gupta, Kayal & Qiao (2013): Randomized $poly(n, s)$ time reconstruction for fan-in two regular formulas.
- Kayal, Nair & S. (2018): Randomized $poly(n, d)$ time reconstruction for constant width homogeneous ABP.

Kayal, S., Saptharishi (2014): Super-poly lower bound known

Reconstruction under non-degeneracy

- Kayal (2012): Randomized $poly(n, s^{\log_d s})$ time reconstruction for depth-3 powering circuits.
- García-Marco, Koiran & Pecatte (2018): Randomized $poly(n, s)$ time reconstruction for depth-3 powering circuits for $s \leq \binom{n+1}{2}$ and $d \geq 5$.
- Gupta, Kayal & Qiao (2013): Randomized $poly(n, s)$ time reconstruction for fan-in two regular formulas.
- Kayal, Nair & S. (2018): Randomized $poly(n, d)$ time reconstruction for constant width homogeneous ABP.

Kumar (2017): Linear width lower bound known

Back to homogeneous depth-3 circuits

The algorithm

$$f = T_1 + T_2 + \cdots + T_s$$

- Let $k = \frac{\log s}{\log(n/ed)}$, $U := \langle \partial^k f \rangle$ and $U_i := \langle \partial^k T_i \rangle$.

  - *Step* 1: Compute a basis of U.
  - *Step* 2: Decompose $U = U_1 \oplus U_2 \oplus \cdots \oplus U_s$.
  - *Step* 3: Compute T_i from a basis of U_i.

The algorithm

$$f = T_1 + T_2 + \cdots + T_s$$

- Let $k = \frac{\log s}{\log(n/ed)}$, $U := \langle \partial^k f \rangle$ and $U_i := \langle \partial^k T_i \rangle$.

Main step

- $Step$ 1: Compute a basis of U.
- $Step$ 2: Decompose $U = U_1 \oplus U_2 \oplus \cdots \oplus U_s$.
- $Step$ 3: Compute T_i from a basis of U_i.

$Step$ 1: Computing a basis of U

- $Fact$ 1: From black-box access to f, we can compute black-box access to $\frac{\partial f}{\partial x}$ in $poly(n, d)$ time.
- $Fact$ 2: From black-box access to $f_1, f_2, \ldots, f_m$, we can compute black-box access to elements of a basis of $\langle f_1, f_2, \ldots, f_m \rangle$ in randomized $poly(n, d, m)$ time.

$Step$ 1: Computing a basis of U

- $Fact$ 1: From black-box access to f, we can compute black-box access to $\frac{\partial f}{\partial x}$ in $poly(n, d)$ time.
- $Fact$ 2: From black-box access to $f_1, f_2, \dots, f_m$, we can compute black-box access to elements of a basis of $\langle f_1, f_2, \dots, f_m \rangle$ in randomized $poly(n, d, m)$ time.

  - Compute black-box access to elements of $\partial^k f$ in $poly(n, s)$ time using $Fact$ 1.
  - Compute black-box access to elements of a basis $\Gamma = (g_1, \dots, g_m)$ of U using $Fact$ 2.

$Step$ 1: Computing a basis of U

- $Fact$ 1: From black-box access to f, we can compute black-box access to $\frac{\partial f}{\partial x}$ in $poly(n,d)$ time.
- $Fact$ 2: From black-box access to $f_1, f_2, \ldots, f_m$, we can compute black-box access to elements of a basis of $\langle f_1, f_2, \ldots, f_m \rangle$ in randomized $poly(n,d,m)$ time.

  - Compute black-box access to elements of $\partial^k f$ in $poly(n,s)$ time using $Fact$ 1. $\boxed{|\partial^k f| = \binom{n+k-1}{k} = poly(n,s)}$
  - Compute black-box access to elements of a basis $\Gamma = (g_1, \ldots, g_m)$ of U using $Fact$ 2.

$Step$ 3: Computing T_i from U_i

- From $Step$ 2, we have black-box access to elements of a basis $(g_{i,1}, \dots, g_{i,m_i})$ of U_i.
- Let $Deg(k)$ be the set of all degree-k monomials in the $\boldsymbol{x}$-variables. $|Deg(k)| = \binom{n+k-1}{k} = poly(n, s)$.

$Step$ 3: Computing T_i from U_i

- From $Step$ 2, we have black-box access to elements of a basis $(g_{i,1}, \dots, g_{i,m_i})$ of U_i.
- Let $Deg(k)$ be the set of all degree-k monomials in the $\boldsymbol{x}$-variables.
  - For $\alpha \in Deg(k)$, solve for $c_{\alpha,i,j} \in \mathbb{F}$ such that

$$\sum_{i\in[s]} c_{\alpha,i,1} \cdot g_{i,1} + \dots + c_{\alpha,i,m_i} \cdot g_{i,m_i} = \frac{\partial^k f}{\partial \alpha}$$

$Step$ 3: Computing T_i from U_i

- From $Step$ 2, we have black-box access to elements of a basis $(g_{i,1}, \ldots, g_{i,m_i})$ of U_i.
- Let $Deg(k)$ be the set of all degree-k monomials in the $\boldsymbol{x}$-variables.
  - For $\alpha \in Deg(k)$, solve for $c_{\alpha,i,j} \in \mathbb{F}$ such that

$$\sum_{i\in[s]} c_{\alpha,i,1} \cdot g_{i,1} + \cdots + c_{\alpha,i,m_i} \cdot g_{i,m_i} = \frac{\partial^k f}{\partial \alpha}$$

We have black-box access

$Step$ 3: Computing T_i from U_i

- From $Step$ 2, we have black-box access to elements of a basis $(g_{i,1}, \dots, g_{i,m_i})$ of U_i.
- Let $Deg(k)$ be the set of all degree-k monomials in the $\boldsymbol{x}$-variables.
  - For $\alpha \in Deg(k)$, solve for $c_{\alpha,i,j} \in \mathbb{F}$ such that

$$\sum_{i\in[s]} c_{\alpha,i,1} \cdot g_{i,1} + \cdots + c_{\alpha,i,m_i} \cdot g_{i,m_i} = \frac{\partial^k f}{\partial \alpha}$$
$$= \frac{\partial^k T_1}{\partial \alpha} + \cdots + \frac{\partial^k T_s}{\partial \alpha}$$

  - Such a solution satisfies

$$c_{\alpha,i,1} \cdot g_{i,1} + \cdots + c_{\alpha,i,m_i} \cdot g_{i,m_i} = \frac{\partial^k T_i}{\partial \alpha}$$

$Step$ 3: Computing T_i from U_i

- From $Step$ 2, we have black-box access to elements of a basis $(g_{i,1}, \dots, g_{i,m_i})$ of U_i.
- Let $Deg(k)$ be the set of all degree-k monomials in the $\boldsymbol{x}$-variables.
  - Well known identity for homogeneous polynomials

$$T_i = \frac{(d-2k)!}{(d-k)!} \cdot \sum_{\alpha \in Deg(k)} \binom{k}{\alpha} \cdot \alpha \cdot \frac{\partial^k T_i}{\partial \alpha}$$

Identifying α with its exponent vector

$Step$ 3: Computing T_i from U_i

- From $Step$ 2, we have black-box access to elements of a basis $(g_{i,1}, \dots, g_{i,m_i})$ of U_i.
- Let $Deg(k)$ be the set of all degree-k monomials in the $\boldsymbol{x}$-variables.
  - Well known identity for homogeneous polynomials

$$T_i = \frac{(d-2k)!}{(d-k)!} \cdot \sum_{\alpha \in Deg(k)} \binom{k}{\alpha} \cdot \alpha \cdot \frac{\partial^k T_i}{\partial \alpha}$$

Thanks to Gaurav Sinha for showing us this argument for executing Step 3 !

A few definitions

- Let U be a space and $\mathcal{S}$ a space of linear operators on U.
- Definition. A space $V \subseteq U$ is an *invariant subspace* of U (induced by $\mathcal{S}$) if $\mathcal{S}V \subseteq V$.

A few definitions

- Let U be a space and $\mathcal{S}$ a space of linear operators on U.
- Definition. A space $V \subseteq U$ is an *invariant subspace* of U (induced by $\mathcal{S}$) if $\mathcal{S}V \subseteq V$. Moreover, V is *irreducible* if there's no invariant subspace properly contained in V.

A few definitions

- Let U be a space and $\mathcal{S}$ a space of linear operators on U.
- Definition. A space $V \subseteq U$ is an *invariant subspace* of U (induced by $\mathcal{S}$) if $\mathcal{S}V \subseteq V$. Moreover, V is *irreducible* if there's no invariant subspace properly contained in V.

- Definition. The *closure* of vector $v \in U$ with respect to $\mathcal{S}$ is the smallest invariant subspace of U containing v.

A few definitions

- Let U be a space and $\mathcal{S}$ a space of linear operators on U.
- Definition. A space $V \subseteq U$ is an *invariant subspace* of U (induced by $\mathcal{S}$) if $\mathcal{S}V \subseteq V$. Moreover, V is *irreducible* if there's no invariant subspace properly contained in V.
- Definition. The *closure* of vector $v \in U$ with respect to $\mathcal{S}$ is the smallest invariant subspace of U containing v.
- *Fact* 3: Given $v \in \mathbb{F}^m$ and a set of matrices $\{M_1, \ldots, M_t\}$ in $\mathbb{F}^{m \times m}$, the closure of v with respect to $\langle M_1, \ldots, M_t \rangle$ can be computed in deterministic $poly(m)$ time.

$Step$ 2: Decomposing U

- The idea:
    - Define a suitable space $\mathcal{S}$ of linear operators on U such that $U_1, \ldots, U_s$ are irreducible invariant subspaces of U induced by $\mathcal{S}$.
    - Pick vectors in U carefully such that the closures of these vectors with respect to $\mathcal{S}$ give $U_1, \ldots, U_s$.

Step 2: Decomposing U

- The idea:

  - Define a suitable space $\mathcal{S}$ of linear operators on U such that $U_1, \dots, U_s$ are irreducible invariant subspaces of U induced by $\mathcal{S}$.

  - Pick vectors in U carefully such that the closures of these vectors with respect to $\mathcal{S}$ give $U_1, \dots, U_s$.

Simultaneous block diagonalization of a basis of $\mathcal{S}$.

Space of linear operators on U

- The shifted differential operator space:
  - $\boldsymbol{\mathcal{SD}}_k := \left\langle \beta \cdot \frac{\partial^k}{\partial \alpha} \ : \ \alpha, \beta \in Deg(k) \right\rangle.$
  - $\boldsymbol{\mathcal{S}} = \boldsymbol{\mathcal{SD}}_{k,U} := \langle \psi \in \boldsymbol{\mathcal{SD}}_k : \psi(U) \subseteq U \rangle.$

Space of linear operators on U

- The shifted differential operator space:
  - $\mathcal{SD}_k := \left\langle \beta \cdot \frac{\partial^k}{\partial \alpha} \;:\; \alpha, \beta \in Deg(k) \right\rangle$.
  - $\mathcal{S} = \mathcal{SD}_{k,U} := \langle \psi \in \mathcal{SD}_k : \psi(U) \subseteq U \rangle$.

- Observation. A basis $(\psi_1, \dots, \psi_t)$ of $\mathcal{S}$ can be computed in $poly(n, s)$ time from a basis $\Gamma = (g_1, \dots, g_m)$ of U.

Space of linear operators on U

- The shifted differential operator space:
  - $$\mathcal{SD}_k := \left\langle \beta \cdot \frac{\partial^k}{\partial \alpha} \;:\; \alpha, \beta \in Deg(k) \right\rangle.$$
  - $$\mathcal{S} = \mathcal{SD}_{k,U} := \langle \psi \in \mathcal{SD}_k : \psi(U) \subseteq U \rangle.$$

- Observation. A basis $(\psi_1, \ldots, \psi_t)$ of $\mathcal{S}$ can be computed in $poly(n, s)$ time from a basis $\Gamma = (g_1, \ldots, g_m)$ of U.
- *Proof.* Solve for $c_{\alpha,\beta}$ and $d_{i,j}$ in $\mathbb{F}$ such that

$$\sum_{\alpha,\beta \in Deg(k)} c_{\alpha,\beta} \cdot \beta \cdot \frac{\partial^k g_i}{\partial \alpha} = \sum_{j \in [m]} d_{i,j} \cdot g_j$$

for every $i \in [m]$.

Space of linear operators on U

- The shifted differential operator space:
  - $\boldsymbol{\mathcal{SD}}_k := \left\langle \beta \cdot \frac{\partial^k}{\partial \alpha} \ : \ \alpha, \beta \in Deg(k) \right\rangle$.
  - $\boldsymbol{\mathcal{S}} = \boldsymbol{\mathcal{SD}}_{k,U} := \langle \psi \in \boldsymbol{\mathcal{SD}}_k : \psi(U) \subseteq U \rangle$.

- Observation. A basis $(\psi_1, \dots, \psi_t)$ of $\boldsymbol{\mathcal{S}}$ can be computed in $poly(n, s)$ time from a basis $\Gamma = (g_1, \dots, g_m)$ of U.
- *Proof.* Solve for $c_{\alpha,\beta}$ and $d_{i,j}$ in $\mathbb{F}$ such that

$$\textstyle\sum_{\alpha,\beta \in Deg(k)} \ c_{\alpha,\beta} \cdot \beta \cdot \frac{\partial^k g_i}{\partial \alpha} \ = \ \sum_{j \in [m]} d_{i,j} \cdot g_j$$

for every $i \in [m]$.

We have black-box access to these polynomials

Nice properties of $\mathcal{S}$

- $Claim$ 1. $U_1, \ldots, U_s$ are invariant subspaces of U induced by $\mathcal{S}$.

Nice properties of $\mathcal{S}$

- $Claim$ 1. $U_1, \dots, U_s$ are invariant subspaces of U induced by $\mathcal{S}$.

… follows from the non-degeneracy condition

Nice properties of $\mathcal{S}$

- $Claim$ 1. $U_1, \dots, U_s$ are invariant subspaces of U induced by $\mathcal{S}$.

- $Claim$ 2. There is an operator $\psi \in \mathcal{S}$ having distinct eigenvalues.

Nice properties of $\mathcal{S}$

- $Claim\ 1.$ $U_1, \ldots, U_s$ are invariant subspaces of U induced by $\mathcal{S}$.

- $Claim\ 2.$ There is an operator $\psi \in \mathcal{S}$ having distinct eigenvalues.

- $Claim\ 3.$ $U_1, \ldots, U_s$ are *irreducible* invariant subspaces of U induced by $\mathcal{S}$.

Nice properties of $\mathcal{S}$

- *Claim* 1. $U_1, \ldots, U_s$ are invariant subspaces of U induced by $\mathcal{S}$.

- *Claim* 2. There is an operator $\psi \in \mathcal{S}$ having distinct eigenvalues.

- *Claim* 3. $U_1, \ldots, U_s$ are *irreducible* invariant subspaces of U induced by $\mathcal{S}$.

- The proofs of *Claim* 2 and *Claim* 3 are a bit technical.

Step 2: Decomposing U

- The algorithm.
  1. Compute a basis $(\psi_1, \ldots, \psi_t)$ of $\boldsymbol{\mathcal{S}}$ from $\Gamma = (g_1, \ldots, g_m)$. Let $M_\Gamma(\psi_i)$ be the $m \times m$ matrix associated with ψ_i.

Once a basis Γ of U is fixed, every operator $\psi \in \boldsymbol{\mathcal{S}}$ can be identified with a unique matrix $M_\Gamma(\psi)$.

$Step$ 2: Decomposing U

- The algorithm.
  1. Compute a basis $(\psi_1, \ldots, \psi_t)$ of $\boldsymbol{S}$ from $\Gamma = (g_1, \ldots, g_m)$. Let $M_\Gamma(\psi_i)$ be the $m \times m$ matrix associated with ψ_i.
  2. Pick $r_1, \ldots, r_t \in \mathbb{F}$ randomly. Let $M_\Gamma = \sum_{i \in [t]} r_i \cdot M_\Gamma(\psi_i)$.

$Step$ 2: Decomposing U

- The algorithm.
  1. Compute a basis $(\psi_1, \dots, \psi_t)$ of $\boldsymbol{\mathcal{S}}$ from $\Gamma = (g_1, \dots, g_m)$. Let $M_\Gamma(\psi_i)$ be the $m \times m$ matrix associated with ψ_i.
  2. Pick $r_1, \dots, r_t \in \mathbb{F}$ randomly. Let $M_\Gamma = \sum_{i \in [t]} r_i \cdot M_\Gamma(\psi_i)$.
  3. Factor the characteristic polynomial $h(y)$ of M_Γ. If h is not square-free, output 'Fail'. Else, let $h = h_1 \cdot h_2 \cdots h_l$.

$Step$ 2: Decomposing U

- The algorithm.
  1. Compute a basis $(\psi_1, \dots, \psi_t)$ of $\boldsymbol{\mathcal{S}}$ from $\Gamma = (g_1, \dots, g_m)$. Let $M_\Gamma(\psi_i)$ be the $m \times m$ matrix associated with ψ_i.
  2. Pick $r_1, \dots, r_t \in \mathbb{F}$ randomly. Let $M_\Gamma = \sum_{i\in[t]} r_i \cdot M_\Gamma(\psi_i)$.
  3. Factor the characteristic polynomial $h(y)$ of M_Γ. If h is not square-free, output 'Fail'. Else, let $h = h_1 \cdot h_2 \cdots h_l$.
  4. Find the null spaces $N_1, \dots, N_l$ of $h_1(M_\Gamma), \cdots, h_l(M_\Gamma)$.

$Step$ 2: Decomposing U

- The algorithm.
  1. Compute a basis $(\psi_1, \dots, \psi_t)$ of $\mathcal{S}$ from $\Gamma = (g_1, \dots, g_m)$. Let $M_\Gamma(\psi_i)$ be the $m \times m$ matrix associated with ψ_i.
  2. Pick $r_1, \dots, r_t \in \mathbb{F}$ randomly. Let $M_\Gamma = \sum_{i \in [t]} r_i \cdot M_\Gamma(\psi_i)$.
  3. Factor the characteristic polynomial $h(y)$ of M_Γ. If h is not square-free, output 'Fail'. Else, let $h = h_1 \cdot h_2 \cdots h_l$.
  4. Find the null spaces $N_1, \dots, N_l$ of $h_1(M_\Gamma), \cdots, h_l(M_\Gamma)$.
  5. For every $j \in [l]$, pick a $v \in N_j$ and compute the closure of v with respect to $\langle M_\Gamma(\psi_1), \dots, M_\Gamma(\psi_t) \rangle$.

$Step$ 2: Decomposing U

- The algorithm.
  1. Compute a basis $(\psi_1, \dots, \psi_t)$ of $\boldsymbol{\mathcal{S}}$ from $\Gamma = (g_1, \dots, g_m)$. Let $M_\Gamma(\psi_i)$ be the $m \times m$ matrix associated with ψ_i.
  2. Pick $r_1, \dots, r_t \in \mathbb{F}$ randomly. Let $M_\Gamma = \sum_{i \in [t]} r_i \cdot M_\Gamma(\psi_i)$.
  3. Factor the characteristic polynomial $h(y)$ of M_Γ. If h is not square-free, output 'Fail'. Else, let $h = h_1 \cdot h_2 \cdots h_l$.
  4. Find the null spaces $N_1, \dots, N_l$ of $h_1(M_\Gamma), \cdots, h_l(M_\Gamma)$.
  5. For every $j \in [l]$, pick a $v \in N_j$ and compute the closure of v with respect to $\langle M_\Gamma(\psi_1), \dots, M_\Gamma(\psi_t) \rangle$.
  6. Let $\{W_1, \dots, W_p\}$ be the set of these closure spaces. If $p \neq s$, return 'Fail'. Else, return bases of $\{\Gamma \cdot W_1, \dots, \Gamma \cdot W_s\}$.

Step 2: Decomposing U

- Analyzing the algorithm.
  1. Compute a basis $(\psi_1, \ldots, \psi_t)$ of $\mathcal{S}$ from $\Gamma = (g_1, \ldots, g_m)$. Let $M_\Gamma(\psi_i)$ be the $m \times m$ matrix associated with ψ_i.
  2. Pick $r_1, \ldots, r_t \in \mathbb{F}$ randomly. Let $M_\Gamma = \sum_{i \in [t]} r_i \cdot M_\Gamma(\psi_i)$.

A random operator in $\mathcal{S}$

Step 2: Decomposing U

- Analyzing the algorithm.
  1. Compute a basis $(\psi_1, \dots, \psi_t)$ of $\mathcal{S}$ from $\Gamma = (g_1, \dots, g_m)$. Let $M_\Gamma(\psi_i)$ be the $m \times m$ matrix associated with ψ_i.
  2. Pick $r_1, \dots, r_t \in \mathbb{F}$ randomly. Let $M_\Gamma = \sum_{i \in [t]} r_i \cdot M_\Gamma(\psi_i)$.
  3. Factor the characteristic polynomial $h(y)$ of M_Γ. If h is not square-free, output 'Fail'. Else, let $h = h_1 \cdot h_2 \cdots h_l$.

h is square-free w.h.p (by $\mathit{Claim}\ 2$).

$\mathit{Claim}\ 2$. There is an operator $\psi \in \mathcal{S}$ having distinct eigenvalues.

Step 2: Decomposing U

- Analyzing the algorithm.
  1. Compute a basis $(\psi_1, \dots, \psi_t)$ of $\boldsymbol{\mathcal{S}}$ from $\Gamma = (g_1, \dots, g_m)$. Let $M_\Gamma(\psi_i)$ be the $m \times m$ matrix associated with ψ_i.
  2. Pick $r_1, \dots, r_t \in \mathbb{F}$ randomly. Let $M_\Gamma = \sum_{i\in[t]} r_i \cdot M_\Gamma(\psi_i)$.
  3. Factor the characteristic polynomial $h(y)$ of M_Γ. If h is not square-free, output 'Fail'. Else, let $h = h_1 \cdot h_2 \cdots h_l$.

h is square-free w.h.p (by *Claim* 2).

Claim 2. ⇒ A random operator in $\boldsymbol{\mathcal{S}}$ has distinct eigenvalues.

Step 2: Decomposing U

- Analyzing the algorithm.
  1. Compute a basis $(\psi_1, \dots, \psi_t)$ of $\boldsymbol{\mathcal{S}}$ from $\Gamma = (g_1, \dots, g_m)$. Let $M_\Gamma(\psi_i)$ be the $m \times m$ matrix associated with ψ_i.
  2. Pick $r_1, \dots, r_t \in \mathbb{F}$ randomly. Let $M_\Gamma = \sum_{i\in[t]} r_i \cdot M_\Gamma(\psi_i)$.
  3. Factor the characteristic polynomial $h(y)$ of M_Γ. If h is not square-free, output 'Fail'. Else, let $h = h_1 \cdot h_2 \cdots h_l$.
  4. Find the null spaces $N_1, \dots, N_l$ of $h_1(M_\Gamma), \cdots, h_l(M_\Gamma)$.

Once a basis Γ of U is fixed, every U_i can be identified with a space $U_{i,\Gamma} \subseteq \mathbb{F}^m$.

Claim 4. Every N_j is contained in some $U_{i,\Gamma}$. (Proof later)

Step 2: Decomposing U

- Analyzing the algorithm.
  1. Compute a basis $(\psi_1, \dots, \psi_t)$ of $\boldsymbol{\mathcal{S}}$ from $\Gamma = (g_1, \dots, g_m)$. Let $M_\Gamma(\psi_i)$ be the $m \times m$ matrix associated with ψ_i.
  2. Pick $r_1, \dots, r_t \in \mathbb{F}$ randomly. Let $M_\Gamma = \sum_{i \in [t]} r_i \cdot M_\Gamma(\psi_i)$.
  3. Factor the characteristic polynomial $h(y)$ of M_Γ. If h is not square-free, output 'Fail'. Else, let $h = h_1 \cdot h_2 \cdots h_l$.
  4. Find the null spaces $N_1, \dots, N_l$ of $h_1(M_\Gamma), \cdots, h_l(M_\Gamma)$.
  5. For every $j \in [l]$, pick a $v \in N_j$ and compute the closure of v with respect to $\langle M_\Gamma(\psi_1), \dots, M_\Gamma(\psi_t) \rangle$.
  6. Let $\{W_1, \dots, W_p\}$ be the set of these closure spaces. If $p \neq s$, return 'Fail'. Else, return bases of $\{\Gamma \cdot W_1, \dots, \Gamma \cdot W_s\}$.

Obs. $\{W_1, \dots, W_s\}$ are the spaces $\{U_{1,\Gamma}, \dots, U_{s,\Gamma}\}$ (by *Claim* 3)

$Step$ 2: Decomposing U

- $Claim$ 4. Every N_j is contained in some $U_{i,\Gamma}$.
- $Proof$. N_j is the null space of $h_j(M_\Gamma)$. The statement of the claim is independent of the choice of basis Γ. Assume that Γ is a basis formed by taking union of bases of $U_1, \ldots, U_s$.

$Step$ 2: Decomposing U

- $Claim$ 4. Every N_j is contained in some $U_{i,\Gamma}$.
- $Proof$. N_j is the null space of $h_j(M_\Gamma)$. By $Claim$ 1,

$$M_\Gamma = \begin{pmatrix} R_1 & & \\ & \ddots & \\ & & R_s \end{pmatrix},$$ is a block-diagonal matrix.

$Step$ 2: Decomposing U

- $Claim$ 4. Every N_j is contained in some $U_{i,\Gamma}$.
- $Proof$. N_j is the null space of $h_j(M_\Gamma)$. By $Claim$ 1,

$$M_\Gamma = \begin{pmatrix} R_1 & & \\ & \ddots & \\ & & R_s \end{pmatrix},$$ is a block-diagonal matrix.

Let p_i be the characteristic polynomial of R_i. Then,

$$h = p_1 \cdot p_2 \cdots p_s = h_1 \cdot h_2 \cdots h_l$$

$Step$ 2: Decomposing U

- $Claim$ 4. Every N_j is contained in some $U_{i,\Gamma}$.
- $Proof$. N_j is the null space of $h_j(M_\Gamma)$. By $Claim$ 1,

$$M_\Gamma = \begin{pmatrix} R_1 & & \\ & \ddots & \\ & & R_s \end{pmatrix},$$ is a block-diagonal matrix.

Let p_i be the characteristic polynomial of R_i. h_j divides some p_i.

Step 2: Decomposing U

- *Claim* 4. Every N_j is contained in some $U_{i,\Gamma}$.
- *Proof*. N_j is the null space of $h_j(M_\Gamma)$. By *Claim* 1,

$$M_\Gamma = \begin{pmatrix} R_1 & & \\ & \ddots & \\ & & R_s \end{pmatrix},$$ is a block-diagonal matrix.

Let p_i be the characteristic polynomial of R_i. h_j divides some p_i.

For $v \in N_j$,

$$p_i(M_\Gamma) = \begin{pmatrix} p_i(R_1) & & \\ & \ddots & \\ & & p_i(R_s) \end{pmatrix}$$

Step 2: Decomposing U

- *Claim* 4. Every N_j is contained in some $U_{i,\Gamma}$.
- *Proof*. N_j is the null space of $h_j(M_\Gamma)$. By *Claim* 1,

$$M_\Gamma = \begin{pmatrix} R_1 & & \\ & \ddots & \\ & & R_s \end{pmatrix},$$ is a block-diagonal matrix.

Let p_i be the characteristic polynomial of R_i. h_j divides some p_i.

For $v \in N_j$,

$$p_i(M_\Gamma) \cdot v = \begin{pmatrix} p_i(R_1) & & \\ & \ddots & \\ & & p_i(R_s) \end{pmatrix} \cdot v = 0$$

$Step$ 2: Decomposing U

- $Claim$ 4. Every N_j is contained in some $U_{i,\Gamma}$.
- $Proof$. N_j is the null space of $h_j(M_\Gamma)$. By $Claim$ 1,

$$M_\Gamma = \begin{pmatrix} R_1 & & \\ & \ddots & \\ & & R_s \end{pmatrix},$$ is a block-diagonal matrix.

Let p_i be the characteristic polynomial of R_i. h_j divides some p_i.

For $v \in N_j$,

$$p_i(M_\Gamma) \cdot v = \begin{pmatrix} p_i(R_1) & & \\ & \ddots & \\ & & p_i(R_s) \end{pmatrix} \cdot \begin{bmatrix} v_1 \\ \vdots \\ v_s \end{bmatrix} = 0$$

$Step$ 2: Decomposing U

- $Claim$ 4. Every N_j is contained in some $U_{i,\Gamma}$.
- $Proof$. N_j is the null space of $h_j(M_\Gamma)$. By $Claim$ 1,

$$M_\Gamma = \begin{pmatrix} R_1 & & \\ & \ddots & \\ & & R_s \end{pmatrix},$$ is a block-diagonal matrix.

Let p_i be the characteristic polynomial of R_i. h_j divides some p_i.

For $v \in N_j$,

$$p_i(M_\Gamma) \cdot v = \begin{pmatrix} p_i(R_1) & & \\ & \ddots & \\ & & p_i(R_s) \end{pmatrix} \cdot \begin{bmatrix} v_1 \\ \vdots \\ v_s \end{bmatrix} = 0$$

$p_i(R_q) \cdot v_q = 0$ for every $q \in [s]$.

Step 2: Decomposing U

- *Claim* 4. Every N_j is contained in some $U_{i,\Gamma}$.
- *Proof*. N_j is the null space of $h_j(M_\Gamma)$. By *Claim* 1,

$$M_\Gamma = \begin{pmatrix} R_1 & & \\ & \ddots & \\ & & R_s \end{pmatrix},$$ is a block-diagonal matrix.

$p_i(R_q) \cdot v_q = 0$ for every $q \in [s]$. Pick any $q \neq i$.

$Step$ 2: Decomposing U

- $Claim$ 4. Every N_j is contained in some $U_{i,\Gamma}$.
- $Proof$. N_j is the null space of $h_j(M_\Gamma)$. By $Claim$ 1,

$$M_\Gamma = \begin{pmatrix} R_1 & & \\ & \ddots & \\ & & R_s \end{pmatrix},$$ is a block-diagonal matrix.

$p_i(R_q) \cdot v_q = 0$ for every $q \in [s]$. Pick any $q \neq i$. As h is square-free, there are polynomials e_1, e_2 such that

$$e_1 \cdot p_i + e_2 \cdot p_q = 1$$

Step 2: Decomposing U

- *Claim* 4. Every N_j is contained in some $U_{i,\Gamma}$.
- *Proof*. N_j is the null space of $h_j(M_\Gamma)$. By *Claim* 1,

$$M_\Gamma = \begin{pmatrix} R_1 & & \\ & \ddots & \\ & & R_s \end{pmatrix},$$ is a block-diagonal matrix.

$p_i(R_q) \cdot v_q = 0$ for every $q \in [s]$. Pick any $q \neq i$. As h is square-free, there are polynomials e_1, e_2 such that

$$e_1 \cdot p_i + e_2 \cdot p_q = 1$$

$$e_1(R_q) \cdot p_i(R_q) = I_m$$

Step 2: Decomposing U

- *Claim* 4. Every N_j is contained in some $U_{i,\Gamma}$.
- *Proof*. N_j is the null space of $h_j(M_\Gamma)$. By *Claim* 1,

$$M_\Gamma = \begin{pmatrix} R_1 & & \\ & \ddots & \\ & & R_s \end{pmatrix},$$ is a block-diagonal matrix.

$p_i(R_q) \cdot v_q = 0$ for every $q \in [s]$. Pick any $q \neq i$. As h is square-free, there are polynomials e_1, e_2 such that

$$e_1 \cdot p_i + e_2 \cdot p_q = 1$$
$$e_1(R_q) \cdot p_i(R_q) \cdot v_q = v_q = 0.$$

$\therefore v \in U_{i,\Gamma}$.

Summary

- We give an efficient reconstruction algorithm for non-degenerate homogeneous depth-3 circuits where both the algorithm and the non-degeneracy condition *originate from the [NW95] natural lower bound proof*.

Summary

- We give an efficient reconstruction algorithm for non-degenerate homogeneous depth-3 circuits where both the algorithm and the non-degeneracy condition *originate from the [NW95] natural lower bound proof.*

- In doing so, we give a *paradigm for handling large fan-in sum gates* by reducing the problem to decomposition of a suitable space U, and then solving this decomposition problem by defining an appropriate space $\mathcal{S}$ of operators on U and examining its structure.

Summary

- The particular operator space we work with is the *shifted differential operator space*. It shows the effectiveness of shifted derivatives in reconstruction problems.

Summary

- The particular operator space we work with is the *shifted differential operator space*. It shows the effectiveness of shifted derivatives in solving reconstruction problems.

- The paradigm has the *potential to give efficient reconstruction for other models* for which natural lower bounds are known. Homogeneous depth-4 circuits, constant depth multilinear circuits, regular formulas are instances of such models.

Summary

- The particular operator space we work with is the *shifted differential operator space*. It shows the effectiveness of shifted derivatives in solving reconstruction problems.

- The paradigm has the ***potential to give efficient reconstruction for other models*** for which natural lower bounds are known. Homogeneous depth-4 circuits, constant depth multilinear circuits, regular formulas are instances of such models.

Thanks!