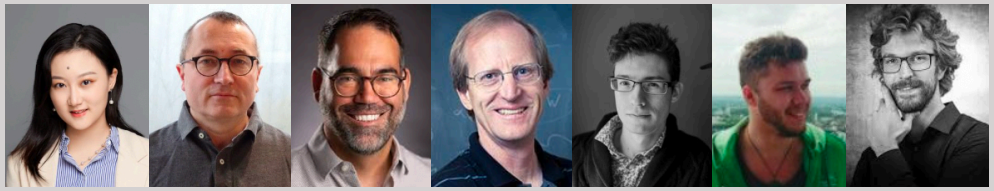
AI for amplitudes: Building sentences from words

Julie Pagès

LITP workshop on theoretical physics and AI — University of Michigan — May 19, 2026



Based on [Cai, Charton, Cranmer, Dixon, Merz, Nolte, Wilhelm, 2405.06107, 2501.05743] and w.i.p.

SLAC NATIONAL ACCELERATOR LABORATORY

Stanford University

U.S. DEPARTMENT of ENERGY

Outline

1. The Language of Amplitudes

- Bootstrap method
- The Symbol
- Linear relations
- The sewing matrix

2. Learning Amplitudes with AI

- Transformers
- AI learned features
- HI (Human Intelligence) learned features

3. Joining forces with AI and HI

- New model
- The “good bases” problem
- Solving loop 9

1

The Language of Amplitudes

Amplitudes

Scattering amplitudes connect
fundamental interactions to measurable quantities.

We are entering a precision era in experimental data.

↪ high precision theoretical prediction required.

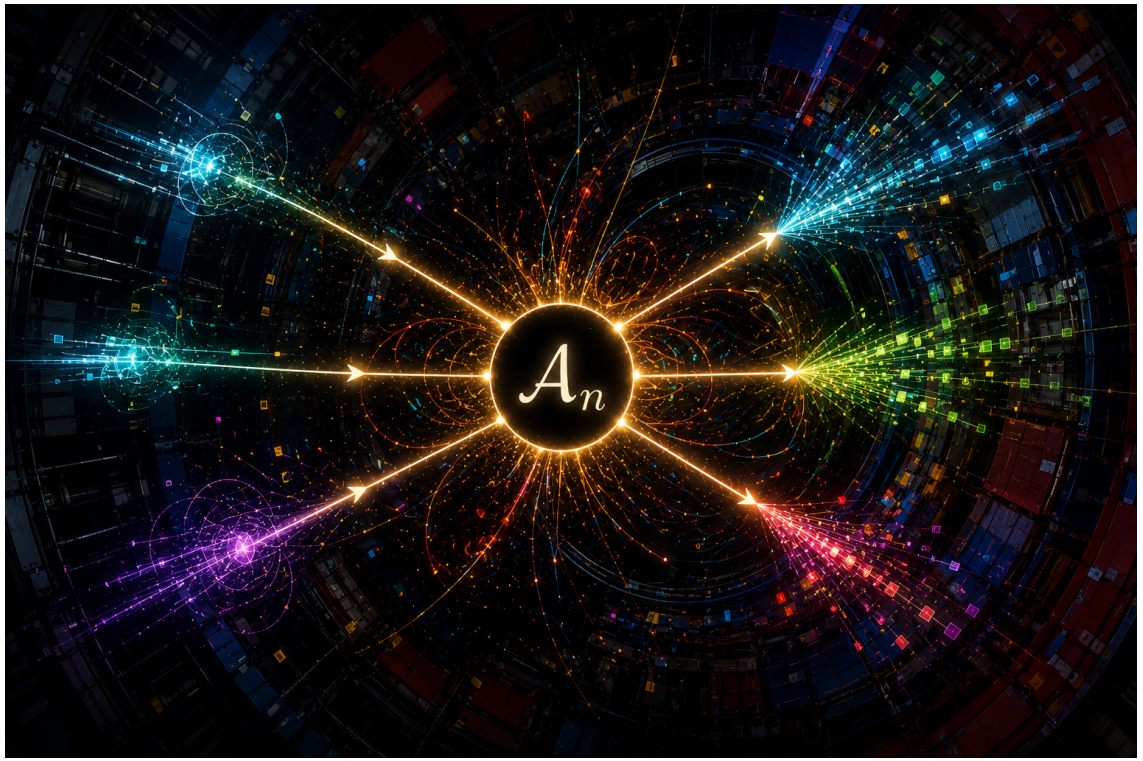
- Traditional methods takes Lagrangians and Feynman diagrams as the fundamental building blocks.

  ↪impractical at high loop order

- Amplitude as fundamental building blocks.

  ↪ hidden simplicity in QFT

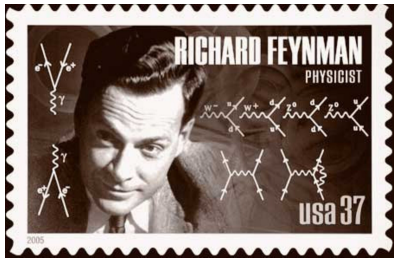
One of the holy grails of the Amplitude program:



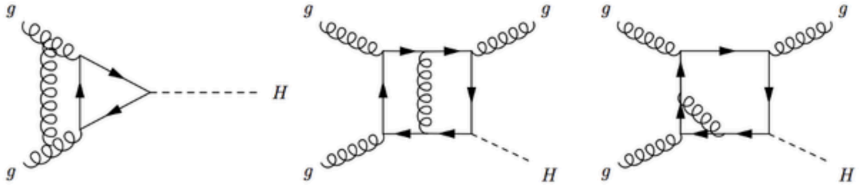
Exact — all loop-order — expressions for scattering amplitudes

Amplitude methods

Feynman diagrams



1. Draw



2. Evaluate with Feynman rules from Lagrangian : I_i

3. Perform all loop integrations: $A_i = \int I_i$

4. Final result $\mathscr{A} = \sum_i A_i$

Bootstrap method

1. Define ansatz $\mathscr{A} = \sum_j c_j F_j$
   - F_j are functions of kinematic variables
   - $c_j \in \mathbb{Q}$ unknown coefficients
2. Use relations: find solution $\tilde{c}_j$ to a set of linear equations relating the c_j
3. Final result: $\mathscr{A} = \sum_j \tilde{c}_j F_j$

Loop Integration → Linear Algebra

The 3-gluon form factor $\sim gg \to Hg$

$\mathcal{N}$=4 super Yang-Mills is a close cousin of QCD

QCD state of the art = 3 loops at large N_c

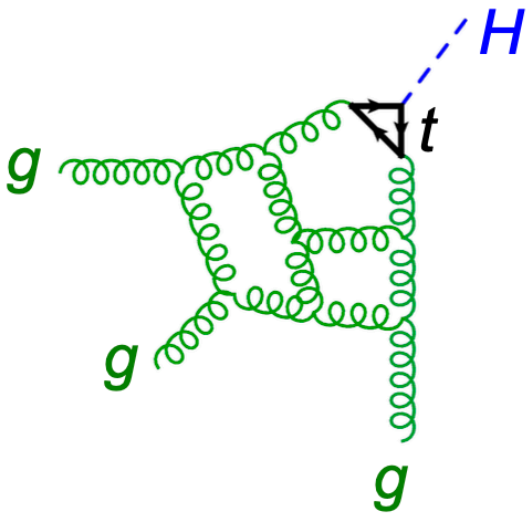
in $gg \to Hg$

[Chen, Guan, Mistlberger, 2504.06490]

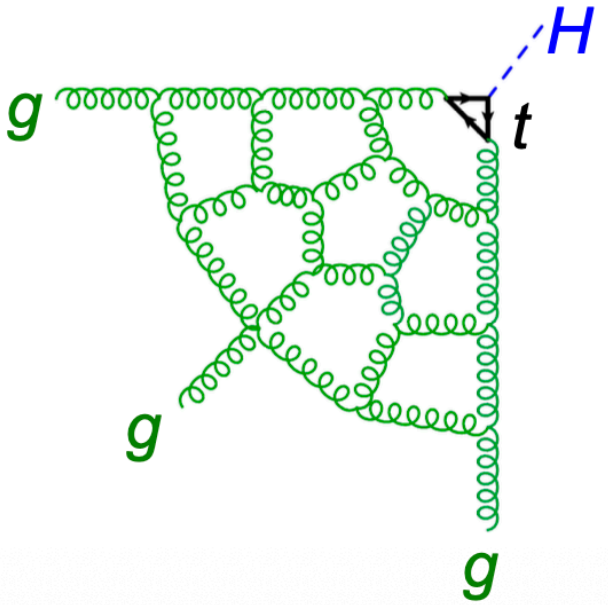
$\mathcal{N} = 4$ SYM state of the art = 8 loops in planar case

in 3-point form factor $\mathcal{F}_3$ of trϕ^2 operator

[Dixon, Gürdoğan, McLeod, Wilhem, 2012.12286, 2112.06243, 2204.11901]



Same maximal transcendental part through 3 loops



Towards all-loop expressions

The BDS ansatz: motivated by IR divergences, one-loop as building block for higher loops [Bern, Dixon ,Smirnov, hep-th/0505205]
Schematically,

$$\mathscr{A}_{\text{BDS}} = \mathscr{A}_{\text{tree}} \exp M_{\text{1L}}$$

Not true for arbitrary process but motivates definition

$$\mathscr{A}_{\text{all}} = \mathscr{A}_{\text{BDS}} \exp R$$

where the IR-finite remainder R has to be computed at loop by loop.

To reduce the function space, exponentiate only part of the one-loop result. Split $M_{\text{1L}} = \hat{M}_{\text{1L}} + \mathscr{E}_{\text{1L}}$ to define

$$\mathscr{A}_{\text{all}} = \underbrace{\mathscr{A}_{\text{tree}} \exp \hat{M}_{\text{1L}}}_{\mathscr{A}_{\text{BDS-like}}} \exp\left(\mathscr{E}_{1L} + R\right)$$

Bootstrap the IR-finite quantity $\mathscr{E}$:

$$\mathscr{E} = \exp\left[\mathscr{E}_{\text{1L}} + R\right]$$

respect Steinmann-like relations

smooth collinear limits

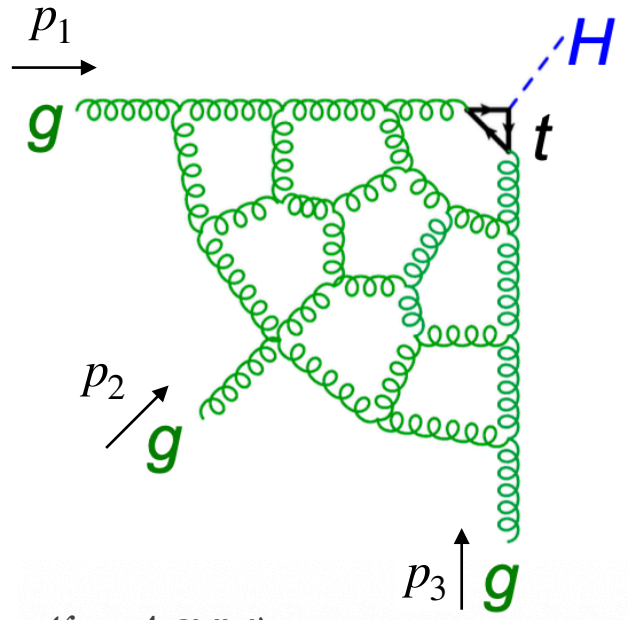
in $\mathcal{N} = 4$ SYM $\mathscr{E}^{(L)} = \sum_j c_j F_j^{(2L)}$

$\in \mathbb{Q}$

The 3-gluon form factor: function space

Functional dependence on kinematics become more and more complex at higher loop order

- $\mathscr{F}_3^{\text{tree}}$: function of kinematic variables (p_1, p_2, p_3)
- $\mathscr{E}_{1\text{L}}(u,v,w) = 2\left[\text{Li}_2\left(1-\frac{1}{u}\right)+\text{Li}_2\left(1-\frac{1}{v}\right)+\text{Li}_2\left(1-\frac{1}{w}\right)\right]$
- $\mathscr{E}_{\text{nL}}$: function of generalized polylogs or more complex



Cross ratios: $u = \frac{s_{12}}{q^2}$ $v = \frac{s_{23}}{q^2}$ $w = \frac{s_{13}}{q^2}$

$s_{ij} = (p_i + p_j)^2$ and $q^2 = s_{12} + s_{23} + s_{13} \Rightarrow u + v + w = 1$

Kinematic functions appear as nested structure (uniform transcendental weight of planar $\mathcal{N} = 4$ SYM)

↪ only weight $2L$ functions at loop L

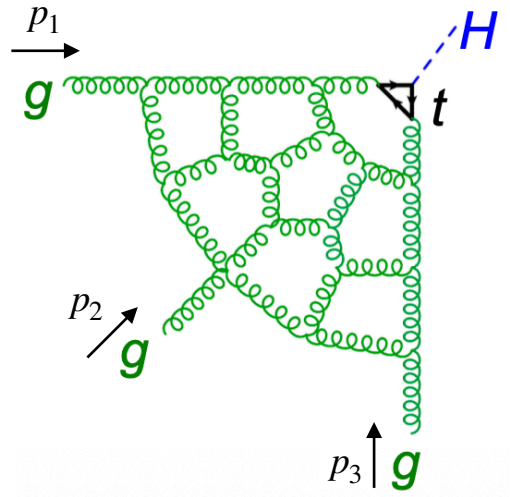
- Dilogarithms: $\text{Li}_2(x) = \int_0^x \frac{dt}{t}\int_0^t \frac{dt'}{1-t'}$, weight 2 (2 integrals)
- Generalized polylogarithms: $G(a_1, a_2, \ldots, a_n, x) = \int_0^x \frac{dt}{t-a_1} G(a_2, \ldots, a_n, t)$

The Symbol

Define transcendental function by their iterated derivative [Goncharov, Spradlin, Vergu, Volovich, 1006.5703]

$$dF = \sum_{s_k \in \mathcal{L}} F^{s_k}\, d\ln s_k$$

$$\mathcal{S}[F] = \sum_{s_{i_1},\ldots,s_{i_n} \in \mathcal{L}} F^{s_{i_1},\ldots,s_{i_n}}\, s_{i_1} \otimes \ldots \otimes s_{i_n}$$



For example

$$\mathscr{E}_{1\mathrm{L}}(u,v,w) = 2\left[\mathrm{Li}_2\left(1-\frac{1}{u}\right)+\mathrm{Li}_2\left(1-\frac{1}{v}\right)+\mathrm{Li}_2\left(1-\frac{1}{w}\right)\right]$$

$$\mathrm{Li}_2(x) = \int_0^x \frac{dt}{t}\int_0^t \frac{dt'}{1-t'} \quad \text{has symbol } \mathcal{S}[\mathrm{Li}_2(x)] = -(1-x)\otimes x$$

$$\mathcal{S}[\mathscr{E}_{1L}] = 2[u\otimes(1-u) - u\otimes u + v\otimes(1-v) - v\otimes v + w\otimes(1-w) - w\otimes w]$$

$$\mathcal{S}[\mathscr{E}_{1L}] = -2[b\otimes d + c\otimes e + a\otimes f + b\otimes f + c\otimes d + a\otimes e]$$

Alphabets

$$\mathscr{L}_u = \{u, v, w, 1-u, 1-v, 1-w\}$$

$$\mathscr{L}_a = \{a, b, c, d, e, f\}$$

$$a = \sqrt{\frac{u}{vw}} \quad b = \sqrt{\frac{v}{wu}} \quad c = \sqrt{\frac{w}{uv}}$$

$$d = \frac{1-u}{u} \quad e = \frac{1-v}{v} \quad f = \frac{1-w}{w}$$

Physical and mathematical constraints

1. Dihedral symmetry: Symmetric under any exchange of the 3 gluons

   Cycle: $\{a,b,c,d,e,f\} \to \{b,c,a,e,f,d\}$

   Flip: $\{a,b,c,d,e,f\} \to \{b,a,c,e,d,f\}$

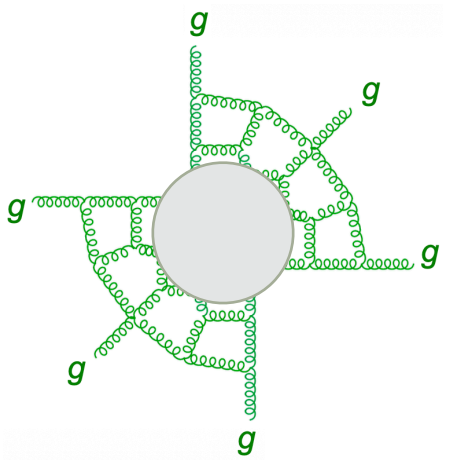
2. Integrability: $\dfrac{\partial^2 F}{\partial u \partial v} = \dfrac{\partial^2 F}{\partial v \partial u}$

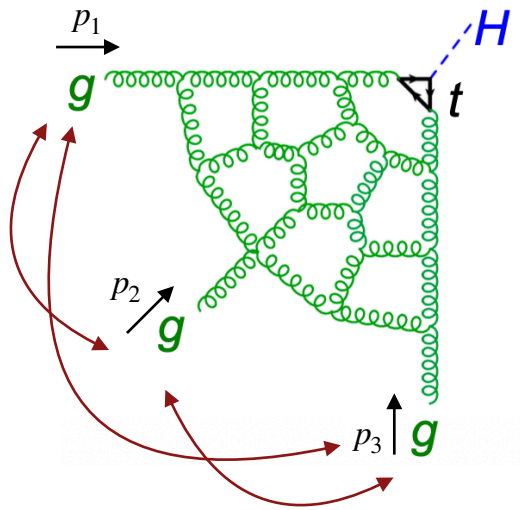
3. Antipodal duality with 6-gluon amplitude

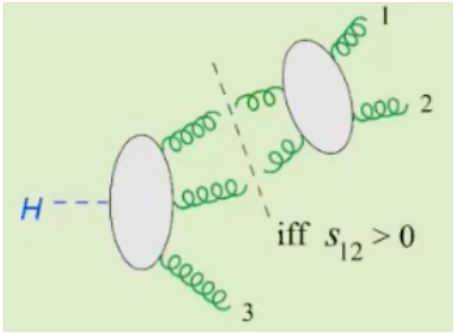
4. Branch-cut conditions: only at physical thresholds

5. Collinear and near-collinear limit (FFOPE) of the remainder R known

   - Discontinuity: L^{th} discontinuity vanishes in R
   - Resummation of $L - 1^{\text{st}}$ discontinuity for 6-gluon R [Z. Li, to appear]







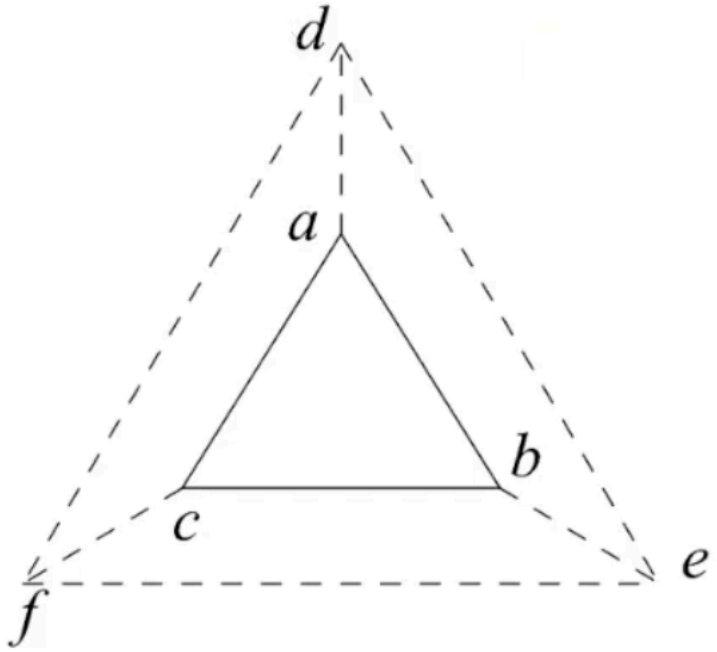
Linear relations

Homogeneous relations:

1. Pair (+triple) relations
   - $ad = da = de = 0$ + dihedral images
   - e.g. $ab - ba + ac - ca = 0$
2. (Multiple) initial-entry relations
   - must start with a, b or c
3. (Multiple) final-entry relations
   - must end with d, e or f

Inhomogeneous relations:

- e.g. $c_3(bddddd) + 2\, c_3(bdddbd) = -64$



- - - forbidden pairs

The Symbol to 9 loops?

Until loop 8, relations fully (over)constrain the symbol

$\mathcal{S}[\mathscr{E}_{1L}] = -2b \otimes d +$ dihedral

$\mathcal{S}[\mathscr{E}_{2L}] = 8\, b \otimes d \otimes d \otimes d + 16\, b \otimes b \otimes b \otimes d\ +$ dihedral

	$L=1$	$L=2$	$L=3$	$L=4$	$L=5$	$L=6$	$L=7$	$L=8$	$L=9$
Total (6^{2L})	36	1296	46,656	$1.7 \cdot 10^6$	$6.0 \cdot 10^7$	$2.2 \cdot 10^9$	$7.8 \cdot 10^{10}$	$2.8 \cdot 10^{12}$	$1.0 \cdot 10^{14}$
Adjacency-allowed	6	102	1830	32,838	589,254	$1.1 \cdot 10^7$	$1.9 \cdot 10^8$	$3.4 \cdot 10^9$	$6.1 \cdot 10^{10}$
Nonzero	6	12	636	11,208	263,880	$4.9 \cdot 10^6$	$9.3 \cdot 10^7$	$1.67 \cdot 10^9$	?

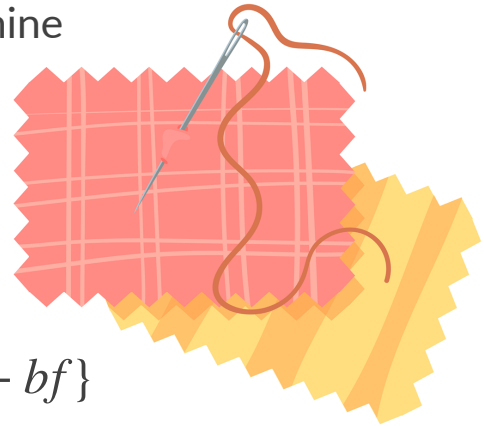
Need some compression from loop 6-7 to manage big number of coefficients.

Can we guess 9 loops from lower-loop data? What format should be used?

The sewing matrix

Build up symbol from front and from back simultaneously, leaves the sewing matrix to determine

$$\mathcal{S}[\mathscr{E}^{(L)}] = \sum_{i,f} F_i^{(\omega)} M_{ij}^{(\omega,\omega')} B_j^{(\omega')}$$



For example,

$F^{(1)} = \{a, b, c\}$ $\quad F^{(2)} = \{aa, bb, cc, bc + cb, ca + ac, ab + ba, bd + cd, ce + ae, af + bf\}$

$B^{(1)} = \{d, e, f\}$ $\quad B^{(2)} = \{dd, ee, ff, ae + af, bd + bf, cd + ce\}$

$$M^{(0,2)} = (0 \quad 0 \quad 0 \quad -2 \quad -2 \quad -2) \qquad M^{(1,1)} = \begin{pmatrix} 0 & -2 & -2 \\ -2 & 0 & -2 \\ -2 & -2 & 0 \end{pmatrix} \qquad M^{(2,0)} = (0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad -2 \quad -2 \quad -2)^T$$

Permissive vs Restrictive format:

$F_P^{(2)} = \{aa, bb, cc, bc + cb, ca + ac, ab + ba, bd + cd, ce + ae, af + bf\}$

$F_R^{(2)} = \{aa, bb, cc, bc, ca, ab, bd, ce, af\}$

Building up front and back space

Reduced spaced constrained by branch cut conditions, antipodal duality and dihedral symmetry:

weight n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$L=1$	1	3	1														
$L=2$	1	3	6	3	1												
$L=3$	1	3	9	12	6	3	1										
$L=4$	1	3	9	21	24	12	6	3	1								
$L=5$	1	3	9	21	46	45	24	12	6	3	1						
$L=6$	1	3	9	21	48	99	85	45	24	12	6	3	1				
$L=7$	1	3	9	21	48	108	236	155	85	45	24	12	6	3	1		
$L=8$	1	3	9	21	48	108	242	466	279	155	85	45	24	12	6	3	1

Words and sentences

$$\mathcal{S}[\mathscr{E}^{(L)}] = \sum_i c_i \, w_i^{(2L)}$$

Amplitude - AI dictionary:

- Symbol $\mathcal{S}[\mathscr{E}^{(L)}]$ = sentence
- Iterative differential representations of transcendental functions of cross ratios = words
- Pair coefficient / word = data point

with

- A 6 letters alphabet: $\{a, b, c, d, e, f\}$
- Words of length $2L$ at L loops: $w_i^{(2L)} = \{\underbrace{aab\ldots fff}_{2L}, \ldots\}$
- Integer coefficients: $c_i \in \mathbb{Z}$

Trillions of data at 8 loops and above make classical linear solver computationally impractical.

↪ Ask AI to find candidate solutions, which can be quickly and exactly verified or discarded.

2

Learning Amplitudes with AI

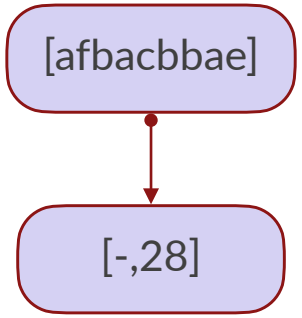
Phase 1: Transforming the bootstrap

A translation problem: words → coefficients [Cai, Charton, Cranmer, Dixon, Merz, Nolte, Wilhelm, 2405.06107] ⇒ Supervised learning

Token embedding:

- Input: word = list of tokens; 1 letter = 1 token,
- Output:
  - Sign token + or -
  - Coefficient = sequence of digits in base 1000

  e.g. 70080 → [+,70,80]



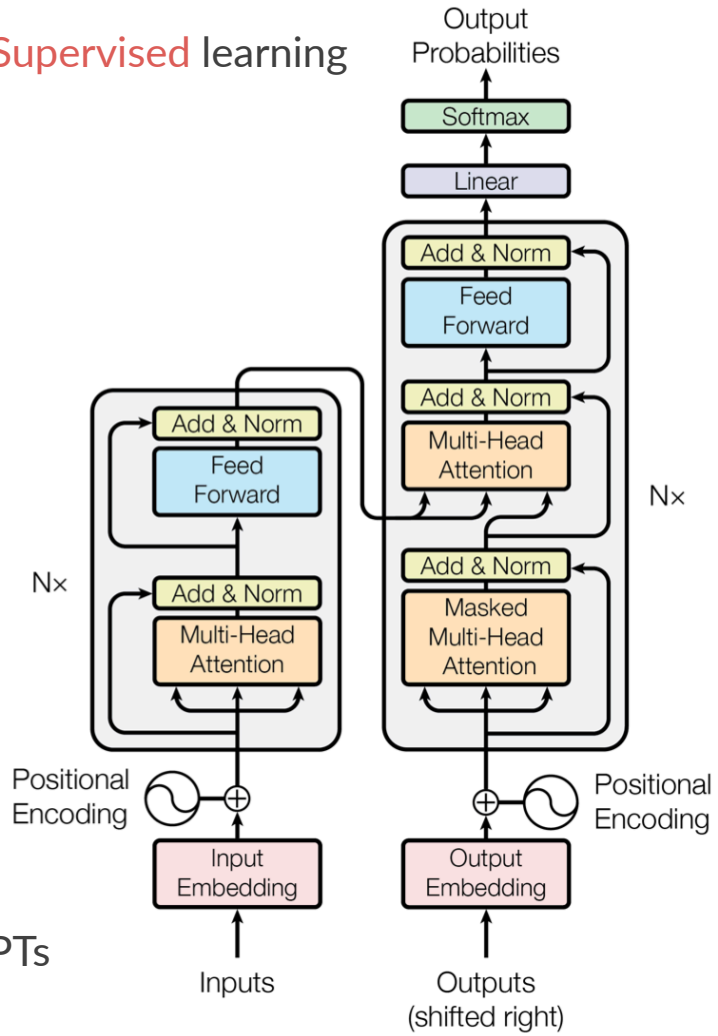
Classification task: minimize token-wise cross-entropy

Architecture: transformer with encoder and decoder

- Up to 8 encoder layers + 8 decoder layers
- 8 or 16 attention heads
- Hidden dimensions $d = 256, 512$ or 1024

⇒ between 4.5 and 245 million trainable parameters ≪ ~ billions from GPTs

Training: on single V100 (or A100) GPU



[Vaswani, Jones, Shazeer, Gomez, Parmar, Uszkoreit, Kaiser, Polosukhin, 1706.03762]

Transforming the bootstrap: coefficients from words

One epoch = pass over 300 000 key coefficients

Loop 6 data: split into train set (4,816,466 nonzero-coefficients terms)

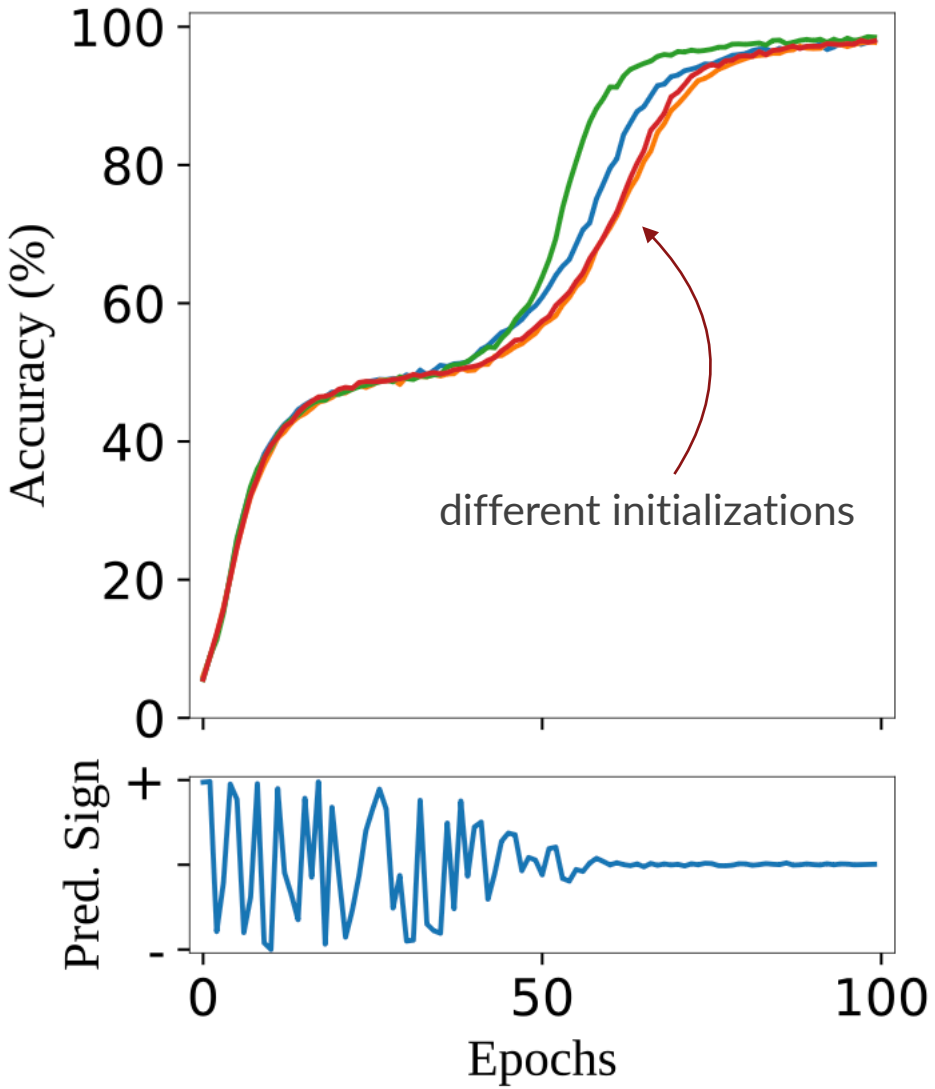
and test set (100,000 terms)

Accuracy: % of correct predictions in test set

$\Rightarrow$ $> 98\,\%$ after 88 epochs, $99.3\,\%$ after 200 epochs

Two learning phases:

- First magnitudes
- Then signs



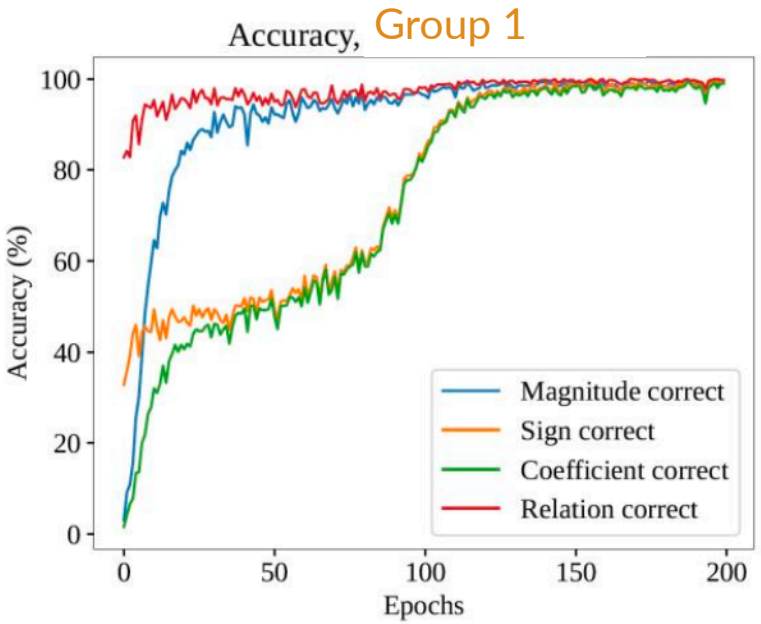
Transforming the bootstrap: relations learned

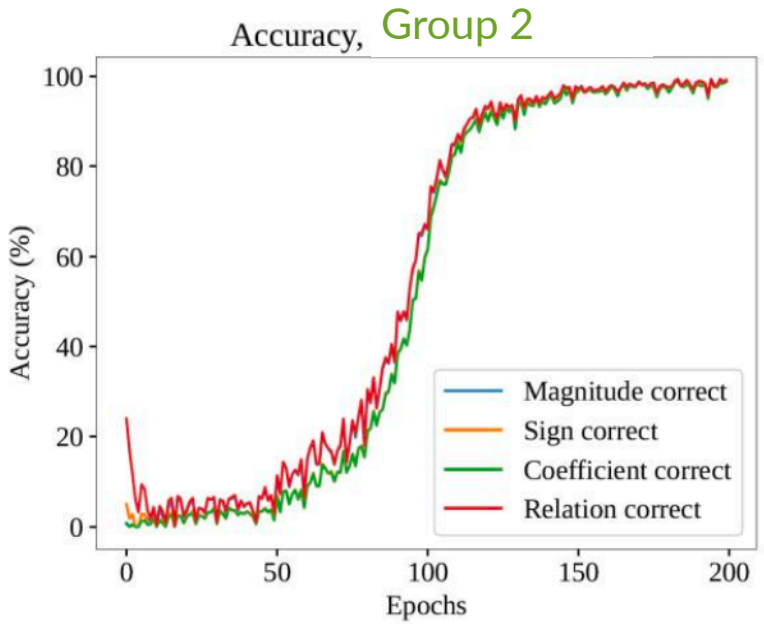
Linear relations learned at different rates.

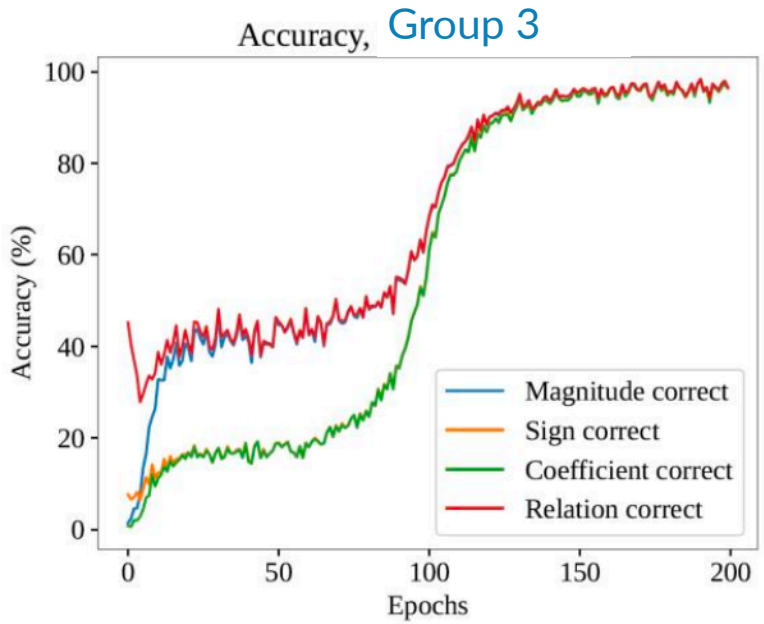
Group 1: pair of equal terms, e.g. $bf - bd = 0$ → learned first

Group 2: at least two terms with opposite signs, e.g. $aab + abb + acb = 0$ → learned last

Group 3: mixed, e.g. $ab + ac - ba - ca = 0$







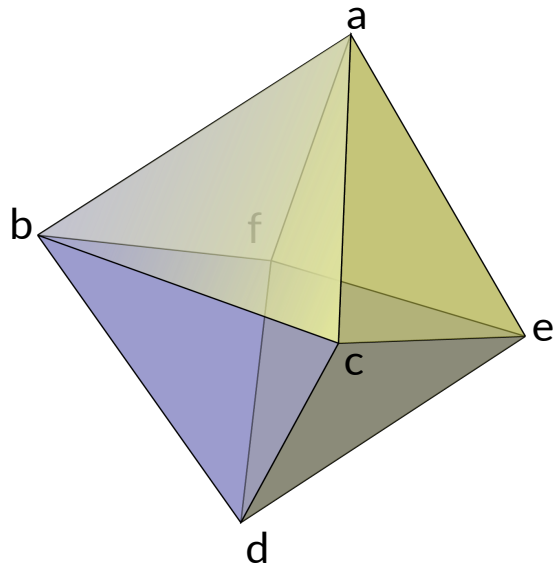
Transforming the bootstrap: dihedral symmetry learned

Dihedral symmetry learned by embedding.

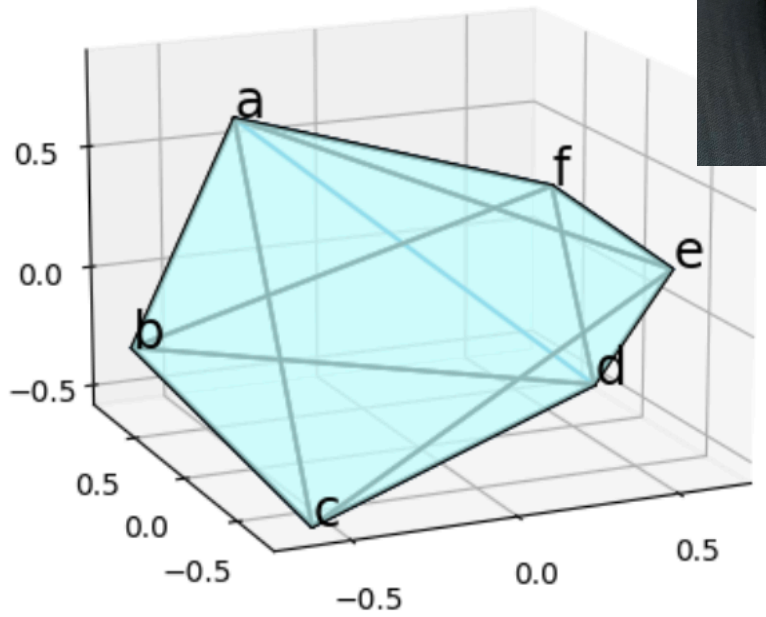
Angles between learned embedding vectors of a 2-layers transformer with $d = 512$ and 8 attention heads trained on $L = 6$ data for 200 epochs:

Δabc and Δdef within 1^o of $60^o \rightarrow$ cycle symmetry

Δabf, Δbcd, Δace within 1^o of $60^o \rightarrow$ flip symmetry



Principal component analysis (PCA) to plot embeddings of letter tokens



Transforming the bootstrap: strike-out experiments

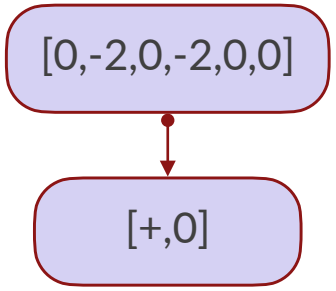
Can AI predict coefficients at loop L from related "parent" coefficients at loop $L-1$?

Experiment: L-loop word: aacf

$L-1$-loop parent words coefficients:

~~aa~~cf=cf, ~~aa~~cf=af, ~~aa~~cf=ac, a~~a~~cf=af, a~~a~~cf=ac, aa~~cf~~=aa

0 −2 0 −2 0 0



Input: list of L-1 parent coefficients in strikeout order

Output: L-loop target word coefficient

For $L = 5 \to L = 6$, after 500 epochs on a 4-layer transformer with $d = 512$ and 8 heads:

Train set: 757.5k; Test set: 10k	Accuracy	Magnitude accuracy	Sign accuracy
Strike-two, all parents	98.1%	98.4%	99.6%

⇒ one learning phase: magnitudes and signs at the same time

Recurrent features

Analytic discovery of new patterns [Cai, Charton, Cranmer, Dixon, Merz, Nolte, Wilhelm, 2501.05743]

Simple linear relations: account for $> 99.7\,\%$ of zeros above $L = 6$

- Zero-suffix rules: $c_L(Xba\ldots af) = 0$ and $c_L(Xca\ldots af) = 0$
- Zero-prefix rules: $c_L(abdZ) = 0$
- Two-term relations: $c_L(Xfaf^{n-1}) = c_L(Xf^k af^{n-k})$

All-loop sequences: closed-form expressions valid at any loop order

- coefficients of $a\ldots af$ are $-2,\ +16,\ -384,\ +15360,\ -860160,\ +61931520\ldots$

  ↪ found with OEIS: $= (-1)^L 2^{3L-2}(2L-3)!!$

- more general $c_L(X_8 f\ldots f) = p_L(X_8 f\ldots f) \times (-1)^L 2^{2L-8}(2L-9)!!$ where p_L is a polynomial in L

Recursion relations: connecting loop L to loop $L-1$ (or even $L-2$)

- $c_L(abbbdd\ldots) = (36-8L)\,c_{L-1}(abbbdd\ldots) - 16\,c_{L-1}(abbddd\ldots)$ and many more

  ↪ inspired by strike-out experiments

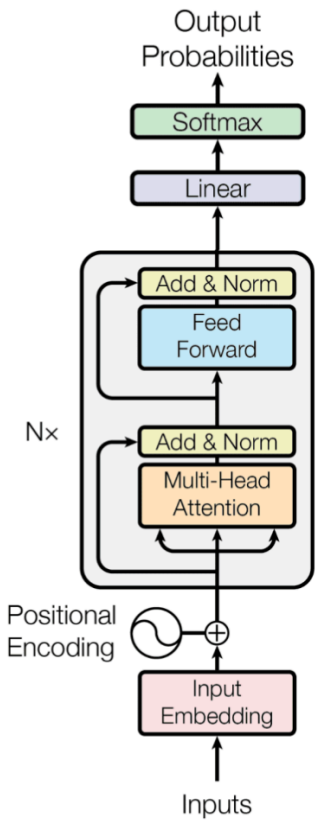
3

Joining forces with AI and HI

Phase 2: towards a foundation model

AI as a “resource of inspiration” and “agent of understanding” dixit Kyle Cranmer

What we are currently trying:



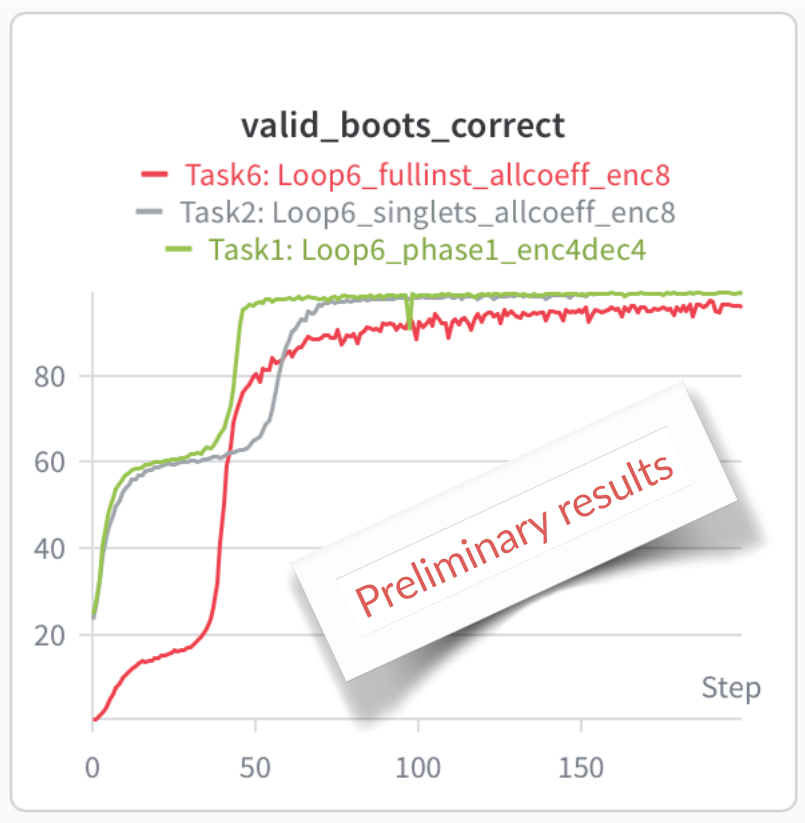
New architecture:

- Encoder-only, like BERT
- with Masked Language modeling

  e.g. [afbacbbaee (-28) + afbacbaaee (-36) + afbacbcaee (64) = 0

New data format:

- full relations instead of individual coefficient/word pair
- compressed symbol with front and/or back space and sewing matrix



Compressing the symbol

Above loop 6, symbol becomes too big to train directly on.

In phase 1, some compression was used on the back space:

- Quadruples: 8 (+dihedral) $\ll$ naive $6^4 = 1296$
- Octuples: 93 (+dihedral) $\ll$ naive $6^8 = 1679616$

which resulted in

- Less coefficients to find
- More subtle relation between them (no dihedral...)

$\hookrightarrow$ required larger models

	$L = 6$	$L = 7$	$L = 8$
None	$4.9 \cdot 10^6$	$9.3 \cdot 10^7$	$1.67 \cdot 10^9$
Quad	$391,570$	$7.3 \cdot 10^6$	
Oct	$16,971$	$312,463$	$5.6 \cdot 10^6$

To reach loop 9, further compression needed.

Streamline the bootstrap using "good bases"

Build up front space and back space with three-index tensors $F_i^{(\omega)} = T_{ijk}\, F_j^{(\omega-1)}\, a_k$ with $a_k = a, b, c, d, e, f$

Example:

- At weight 1 : $B^{(1)} = \{d, e, f\}$
- At weight 2: tag one letter to the left of $B^{(1)}$ from $\{a, b, c, d, e, f\}$ and look at relations
  - $ad, be, cf, de, df, ed, ef, fd, fe$ forbidden
  - dd, ee, ff mandatory
  - Permissive format: $B_P^{(2)} = \{dd, ee, ff, ae + af, bd + bf, cd + ce\}$
  - Restrictive format: $B_R^{(2)} = \{dd, ee, ff, bd, ce, af\}$

In restrictive format, only monomials, but which ones are best? Different possible basis choice.

Define dependents D related to independent I through relations $D_i = M_{ij} I_j$, e.g. $\begin{pmatrix} bf \\ ae \\ cd \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} af \\ bd \\ ce \end{pmatrix}$

Rules of the game:

- Action = switch one dependent with one independent, e.g. take bf in basis instead of bd
- Goal = make M as integer and sparse as possible → less and simpler linear equations to solve

↪ try with reinforcement learning or unscrambling? See David Shih's talk

Towards loop 9

At loop 9, $\sim 3 \cdot 10^{10}$ non-zero coefficient v.s. 1251 (weight 8 front) x 809 (weight 10 back) $\sim 1 \cdot 10^{9}$ in sewing format.

Pushing on front space and back space independently (back space dimension grows more slowly)

$$\mathcal{S}[\mathscr{E}^{(L)}] = \sum_{i,f} F_i^{(\omega)} M_{ij}^{(\omega,\omega')} B_j^{(\omega')}$$

$++\omega$ $\quad$ $++\omega'$

leaves the sewing matrix to determine.

Known relations can be translated to constraints on the sewing matrix elements.

Remaning problem:

Solve

$$A \cdot \mathbf{x} = \mathbf{b}$$

where the matrix A encode the relations on the linearized sewing matrix elements $\mathbf{x}$ and $\mathbf{b}$ contains the inhomogeneous part (elements = 0 for homogeneous relations)

⇒ very sparse system with integer solution → Mixed Integer Linear Programming (MILP) v.s. solving modulo primes?

Conclusion

Conclusion

- Amplitude bootstrap provides a new playground for AI and HI collaboration