Bootstrapping the 3d Ising stress tensor

Cyuan-Han Chang ⬩,[a,b] **Vasiliy Dommes** ⬩,[a] **Rajeev S. Erramilli** ⬩,[c]
Alexandre Homrich ⬩,[d,e] **Petr Kravchuk** ⬩,[f] **Aike Liu** ⬩,[a] **Matthew S. Mitchell** ⬩,[g]
David Poland ⬩[g] **and David Simmons-Duffin** ⬩[a]

[a] *Walter Burke Institute for Theoretical Physics, Caltech,*
1200 East California Blvd, Pasadena, CA 91125, U.S.A.

[b] *Kadanoff Center for Theoretical Physics & Enrico Fermi Institute, University of Chicago,*
933 East 56th Street, Chicago, IL 60637, U.S.A.

[c] *Institut des Hautes Études Scientifiques,*
35 Route de Chartres, Bures-sur-Yvette 91440, France

[d] *Perimeter Institute for Theoretical Physics,*
31 Caroline Street North, Waterloo, Ontario N2L 2Y5, Canada

[e] *Kavli Institute for Theoretical Physics, University of California,*
552 University Road, Santa Barbara, CA 93106, U.S.A.

[f] *Department of Mathematics, King's College London,*
Strand, London WC2R 2LS, U.K.

[g] *Department of Physics, Yale University,*
217 Prospect St, New Haven, CT 06520, U.S.A.

E-mail: cchang10@uchicago.edu, vdommes@caltech.edu, erramilli@ihes.fr,
ahomrich@kitp.ucsb.edu, petr.kravchuk@kcl.ac.uk, aliu7@caltech.edu,
matthew.mitchell@yale.edu, david.poland@yale.edu, dsd@caltech.edu

ABSTRACT: We compute observables of the critical 3d Ising model to high precision by applying the numerical conformal bootstrap to mixed correlators of the leading scalar operators σ and ϵ, and the stress tensor $T_{\mu\nu}$. We obtain new precise determinations of scaling dimensions $(\Delta_\sigma, \Delta_\epsilon) = (0.518148806(24), 1.41262528(29))$ as well as OPE coefficients involving σ, ϵ, and $T_{\mu\nu}$. We also describe several improvements made along the way to algorithms and software tools for the numerical bootstrap.

Contents

1 Introduction

The dream of the conformal bootstrap [1, 2] is to use consistency conditions to solve and classify the scale-invariant structures underlying critical phenomena. A tantalizing target for

this program is the critical 3d Ising model, which describes the critical point of water, uniaxial magnets, ϕ^4 theory at strong coupling, and more. This theory has been studied for decades using the ϵ-expansion [3], high/low-temperature expansions, and Monte Carlo simulations [4]. The most precise results for its critical exponents have come from numerical conformal bootstrap methods [5–9] applied to 4-point correlation functions of the leading $\mathbb{Z}_2$-odd and $\mathbb{Z}_2$-even scalar operators σ and ϵ [10, 11]. This approach has led to determinations of scaling dimensions $\Delta_\sigma = 1/2 + \eta/2 = 0.5181489(10)$ and $\Delta_\epsilon = 3 - 1/\nu = 1.412625(10)$, along with determinations of operator product expansion (OPE) coefficients, and clear reconstructions of leading Regge trajectories [12].

However, there are several reasons why we would like to go beyond these results. Firstly, it is not clear that the allowed region of [11] can be made systematically smaller by increasing the derivative order of bootstrap functionals. Relatedly, there are certain low-lying Regge trajectories of operators that are expected theoretically [13–15] but not present in the extremal spectra studied in [12]. These include the $[\sigma T]$, $[\epsilon T]$, and $[TT]$ double-twist trajectories built out of the stress-energy tensor $T^{\mu\nu}$ (which can also be interpreted as triple-twist or quadruple-twist trajectories built out of σ). Furthermore, the spectra of [12] give no information about $\mathbb{Z}_2$-even operators of odd spin, parity-odd operators, or operator product expansion (OPE) coefficients involving multiple spinning operators (e.g. those appearing in $\langle TT\epsilon \rangle$ or $\langle TTT \rangle$).

Meanwhile, there has been recent progress exploring the bootstrap for current [16–18] and stress-tensor 4-point functions [19]. The latter study revealed contours of a global map of the space of CFTs, parameterized by parity-even and parity-odd scalar gaps (Δ_+, Δ_-). This map included, among other features, a sharp discontinuity for Δ_+ near the 3d Ising value of Δ_ϵ. This was a surprise, as it was not a priori clear that the stress-tensor bootstrap should know anything about the 3d Ising CFT. This structure was leveraged in [19] to yield rough estimates of the 3d Ising $\langle TTT \rangle$ 3-point function coefficients.

A natural continuation of these works is to consider a mixed-correlator system involving all nonvanishing 4-point functions of σ, ϵ, and $T^{\mu\nu}$. While such a large system comes with technical challenges, they are not insurmountable and we will see that combining these constraints allows us to make precise predictions for CFT data involving the stress tensor, opening up a new era of precision in the numerical study of the 3d Ising CFT. Along the way we have significantly improved several software tools and algorithms used for the numerical bootstrap. For the impatient reader, a concise summary of our new results for scaling dimensions and OPE coefficients is provided in table 1. The corresponding bounds on the critical exponents η, ν are shown in table 2. We also provide a plot of the allowed island in $(\Delta_\sigma, \Delta_\epsilon)$-space (which is about 50 times smaller than the previous state of the art) in figure 1.

Experience has shown that bootstrap bounds tend to converge as one increases the derivative order, but these converged bounds are *not* exactly saturated by physical theories (though they may be nearly saturated). In other words, there is a limit to the strength of bounds extracted from a fixed, finite set of crossing equations. In the context of the hyperbolic bootstrap [28–30], there are essentially rigorous arguments that this non-saturation occurs [31]. Thus, if the bootstrap is to become a practical tool for solving a theory, we *must* explore larger and larger systems of crossing equations. We view our results as more evidence (along with [10, 32–34]) that this is a fruitful thing to do!

	$T\sigma\epsilon$ (this work)	prev. bootstrap	monte carlo	fuzzy sphere
Δ_σ	0.518148806(**24**)	0.5181489(10),* [11]	0.518142(20) [20]	0.524 [21], 0.519 [22]
Δ_ϵ	1.41262528(**29**)	1.412625(10),* [11]	1.41265(13) [20]	1.414 [21], 1.403 [22]
$\lambda_{\sigma\sigma\epsilon}$	1.05185373(11)	1.0518537(41) [11]	1.051(1) [23]	1.0539(18) [24]
$\lambda_{\epsilon\epsilon\epsilon}$	1.53244304(58)	1.532435(19) [11]	1.533(5) [23]	1.5441(23) [24]
c_T/c_B	0.946538675(42)	0.946534(11) [25]	0.952(29) [26]	0.955(21) [24]
$\lambda_{TT\epsilon}$	0.95331513(42)	0.958(7) [27]	—	0.9162(73) [24]
n_B	0.933444559(75)	0.933(4),** [19]	—	—
n_F	0.013094116(33)	0.014(4),** [19]	—	—

*The errors in [11] have been slightly underestimated, see footnote 1.

**The results for (n_B, n_F) in [19] assume a parity-odd scalar gap $\Delta_- \geq 10$. The weaker assumption $\Delta_- \geq 3$ gave $(n_B, n_F) = (0.919(18), 0.027(18))$.

Table 1. Bounds on scaling dimensions and OPE coefficients in the 3d Ising CFT, using the numerical bootstrap for correlators of σ, ϵ, and the stress tensor T. Here, c_T/c_B is the coefficient of the stress-tensor two-point function (the "central charge") normalized by the free boson value, and n_B and n_F are OPE coefficients in the stress-tensor three-point function. The OPE coefficient $\lambda_{TT\epsilon}$ is defined in section 2.1. Our bounds are computed using derivative order $\Lambda = 43$. Bold-face errors are rigorous. We also include comparisons to previous bootstrap calculations and the best determinations we are aware of from other numerical methods.

	η	ν
$T\sigma\epsilon$ (this work)	0.036297612(**48**)	0.62997097(**12**)
$\sigma\epsilon$ bootstrap [11]	0.0362978(20)	0.6299710(40)
monte carlo [20]	0.036284(40)	0.62998(5)

Table 2. Bounds on the critical exponents, derived from the scaling dimensions in table 1 for the reader's convenience. Bold-face errors are rigorous.

This paper is organized as follows. In section 2 we describe the basic setup of the bootstrap problem, including the analysis of the 3-point structures, Ward identities and conservation constraints, and crossing equations. In section 3 we describe our numerical implementation of the bootstrap problem, along with various improvements to software tools that we have made. In section 4 we describe a new rigorous approach to certifying disallowed points in the context of scanning over OPE coefficients. Section 5 describes the results of our computations, including new bounds from the $\langle TTTT \rangle$ single-correlator system and precise computations of 3d Ising data using the mixed correlator $\{\sigma, \epsilon, T\}$ system. We conclude in section 6 and give details of our setup and implementation in the appendices.

2 Setup

The non-trivial four-point functions in the $\{\sigma, \epsilon, T\}$ system are

$$\{\langle\sigma\sigma\sigma\sigma\rangle, \langle\sigma\sigma\epsilon\epsilon\rangle, \langle\epsilon\epsilon\epsilon\epsilon\rangle, \langle\sigma\sigma\epsilon T\rangle, \langle\epsilon\epsilon\epsilon T\rangle, \langle\sigma\sigma TT\rangle, \langle\epsilon\epsilon TT\rangle, \langle\epsilon TTT\rangle, \langle TTTT\rangle\}. \quad (2.1)$$
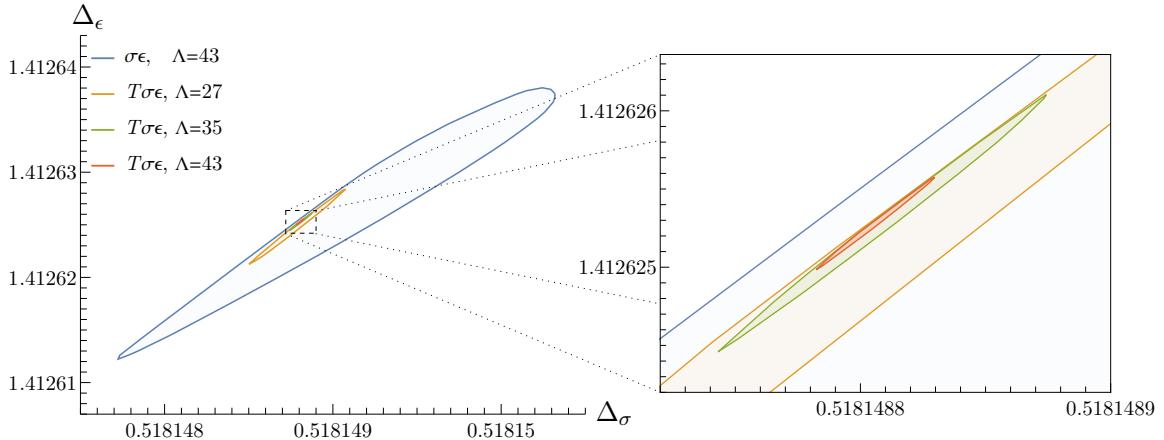
Figure 1. Allowed regions in $(\Delta_\sigma, \Delta_\epsilon)$-space determined from the $T\sigma\epsilon$ system of correlators at $\Lambda = 27, 35, 43$ (orange, green, and red), together with the allowed region determined from the $\sigma\epsilon$ system at $\Lambda = 43$ (blue) [11].[1] On the left, we show a wider view where all the regions are visible. On the right, we zoom in on the $T\sigma\epsilon$ regions at $\Lambda = 35$ and $\Lambda = 43$. At $\Lambda = 43$, the $T\sigma\epsilon$ allowed region is roughly 40–50× smaller than the $\sigma\epsilon$ allowed region.

They are constrained by conformal, parity, and permutation symmetries. Moreover, the correlators involving the stress tensor must be conserved and satisfy Ward identities. In this section, we describe how these constraints are imposed and determine a linearly-independent set of crossing equations. We also study the three-point functions relevant for constructing the conformal block expansions of the above four-point functions. We generally follow the formalism developed and described in [19, 34–38], with a few additional considerations necessary for numerical stability.

2.1 Three-point structures

We have studied all the OPEs between σ, ϵ, T and computed their q-basis [35] three-point structures satisfying conformal invariance, invariance under space parity and permutations, and conservation of T. With the exception of the latter, these constraints can be straightforwardly implemented using the formalism described in [35]. The conservation of T can be imposed by converting to embedding space bases [39, 40]; we, however, chose to impose it directly in the q-basis, which can be done following the discussion in section 4.3 of [41]. Since these manipulations are standard, we do not describe them in detail and refer the interested reader to the above references.

The results for the representations of primary operators $\mathcal{O}$ operators appearing in various OPEs and the respective numbers of tensor structures are summarized in table 3. Note that the counting of conserved structures generally splits into two cases, generic $\mathcal{O}$ and special $\mathcal{O}$. This is why the σ operator in the $\sigma \times T$ OPE, ϵ in the $\epsilon \times T$ OPE, and T in the $T \times T$

---

[1]The $\sigma\epsilon$ allowed region shown in this figure was re-computed for this work and is a more accurate version of the allowed region from [11]. Compared to [11], we have benefitted from the OPE cutting surface algorithm [32], improvements in computational efficiency, and more powerful computational resources. Our $\sigma\epsilon$ allowed region is slightly larger than the one in [11] but this does not significantly affect any of their results. We thank Slava Rychkov and Ning Su for discussions that motivated us to perform this calculation.

OPE	spin	$\mathbb{Z}_2$	#structures$^{\text{parity}}$
$\sigma \times \sigma$	even	even	1^+
$\epsilon \times \epsilon$	even	even	1^+
$\sigma \times \epsilon$	$j \geq 0$	odd	1^+
$\sigma \times T$	σ		1^+
	$j \geq 2$	odd	$1^+ + 1^-$
$\epsilon \times T$	ϵ		1^+
	$j \geq 2$	even	$1^+ + 1^-$
$T \times T$	$j = 0$	even	$1^+ + 1^-$
	$j = 2$	even	$1^+ + 1^-$
	T		2^+
	$j \geq 4$ even	even	$2^+ + 1^-$
	$j \geq 5$ odd	even	1^-

Table 3. The allowed operators and number of tensor structures in each OPE between σ, ϵ, and T. The number of structures is labeled by parity: n^+ stands for n parity-even structures, while n^- stands for n parity-odd structures.

OPE are treated separately. For example, conservation and Ward identities imply that σ is the only scalar that can appear in the $\sigma \times T$ OPE.

The last step is to convert the tensor structures from the q-basis to the $\mathrm{SO}(3)_r$ basis used by `blocks_3d` [37, 38], a program that can compute arbitrary four-point conformal blocks in 3d CFTs. This can be done by writing each q-basis structure as a linear combination of the $\mathrm{SO}(3)_r$ basis structures with the coefficients given by products of the Clebsch-Gordan coefficients; we refer the reader to [37] for more details. The explicit expressions of all the three-point structures in the $\mathrm{SO}(3)_r$ basis are given in the supplementary material `three_point_structures.m`. We have also checked that our structures agree with previous literature.[2] The full list of three-point structures corresponding to the possible OPE coefficients in the $T\sigma\epsilon$ system are listed in appendix A.

Our definition of the OPE coefficient $\lambda_{TT\epsilon}$ is slightly modified in order to match the conventions of [42], namely:

$$\langle \widehat{T}(X_1, Z_1)\widehat{T}(X_2, Z_2)\epsilon(X_3)\rangle$$

$$= \lambda_{TT\epsilon} \frac{1}{X_{12}^{\frac{6-\Delta_\epsilon}{2}+2} X_{23}^{\frac{\Delta_\epsilon}{2}} X_{31}^{\frac{\Delta_\epsilon}{2}}} \left(V_1^2 V_2^2 + \frac{4(\Delta_\epsilon - 4)}{\Delta_\epsilon + 2} H_{12} V_1 V_2 + \frac{2(\Delta_\epsilon^2 - 6\Delta_\epsilon + 6)}{\Delta_\epsilon(\Delta_\epsilon + 2)} H_{12}^2 \right)$$

$$= -\lambda_{TT\epsilon} \frac{16\Delta_\epsilon}{2 + \Delta_\epsilon} \text{ ScalarParityEven}_{\Delta_\epsilon}, \tag{2.2}$$

---

[2]In particular, the $\langle TT\mathcal{O} \rangle$ structures are linear combinations of the differential basis expressions used in the code of [19]. The actual expressions listed in appendix A of [19] for the $\langle TTT \rangle$ and parity-odd $\langle TT\mathcal{O} \rangle$ structures contain typos. See the supplementary material `three_point_structures.m` for the corrected explicit expressions.

where V_i and H_{ij} are standard embedding space structures [39], and $\widehat{T}$ is the stress-tensor with *unit-normalized* two-point function, defined in section 2.2 below. In the $SO(3)_r$ basis, the structure $\texttt{ScalarParityEven}_\Delta$ is defined as

$$\texttt{ScalarParityEven}_\Delta = -\frac{(\Delta - 5)(\Delta - 3)}{4\sqrt{5}\Delta^2}|0, 0\rangle + \frac{(5 - \Delta)}{2\sqrt{14}\Delta}|2, 2\rangle - \frac{(\Delta + 2)}{4\sqrt{70}\Delta}|4, 4\rangle, \quad (2.3)$$

where $|j_{12}, j_{123}\rangle$ are $SO(3)_r$ basis elements. ($\texttt{ScalarParityEven}_\Delta$ is also given in a machine-readable format in the supplementary material.) The expressions for (2.2) under permutations of the 3 operators are given in (A.1).

Linear independence of structures. We emphasize that it is important to ensure that three-point structures for a given channel are linearly-independent for *all* allowed values of Δ.[3] Without care, it can happen that the space spanned by a naïve choice of conserved structures is degenerate at specific values of Δ. This occurs, for example, if one solves the conservation equations in $\texttt{Mathematica}$ in a generic basis.

These comments include the special case where structures have different asymptotic behaviors in the limit of $\Delta \to \infty$. When this happens, the subleading structures effectively become zero, so we have a linear dependence at $\Delta = \infty$. To address this issue, we choose to normalize our structures so that they all grow as $O(1)$ as $\Delta \to \infty$.

When we first set up the $\langle TTTT \rangle$ system, we hadn't appreciated the significance of this requirement. We found to our frustration that SDPB never terminated with $\texttt{dual feasible}$ $\texttt{jump detected}$. We eventually realized that any finite linear combination of functionals in our chosen basis would always lead to a degenerate matrix at some Δ (including $\Delta = \infty$), so the solver could never find a solution where all matrices are strictly positive definite, a prerequisite for a dual feasible jump. Dual jumps are essential for efficient checkpointing [43], and consequently for efficient use of the OPE cutting surface algorithm [32]. Thus, ensuring linear independence of three-point structures is a requirement for this kind of numerical study.

2.2 Ward identities

Some three-point functions involving the stress tensor satisfy Ward identities, which can be seen by integrating T over a surface to obtain conformal generators. For example, the dilatation operator on a surface Σ is given by

$$D(\Sigma) = -\int_\Sigma dS_\mu x_\nu T^{\mu\nu}. \quad (2.4)$$

For a three-point function $\langle \mathcal{O}\mathcal{O}T \rangle$, choosing Σ to surround one of the $\mathcal{O}$'s then gives a Ward identity of the form

$$-\int_\Sigma dS_\mu x_\nu \langle \mathcal{O}\mathcal{O}T^{\mu\nu} \rangle = \Delta_\mathcal{O} \langle \mathcal{O}\mathcal{O} \rangle, \quad (2.5)$$

which relates the OPE coefficients $\lambda_{\mathcal{O}\mathcal{O}T}$ to $\Delta_\mathcal{O}$.

---

[3]We observed empirically that one only needs to ensure linear independence for the values of Δ allowed by unitarity and the gap assumptions. Presumably this happens because in SDPB the conformal blocks are eventually sampled at a large number of allowed values of Δ and polynomially interpolated. This interpolation will very accurately reproduce any degeneracies in the allowed region of Δ, but will likely break any degeneracies outside of this region where the polynomial interpolation is of poor quality.

The two-point function of stress tensors can be parametrized as

$$\langle TT \rangle = c_T \frac{H_{12}^2}{X_{12}^5}, \tag{2.6}$$

where H_{12} and X_{12} are standard embedding-space structures [39], and c_T is called the "central charge".[4] The three-point function of stress tensors takes the form

$$\langle TTT \rangle = n_B \langle TTT \rangle_B + n_F \langle TTT \rangle_F, \tag{2.7}$$

where $\langle TTT \rangle_B$ and $\langle TTT \rangle_F$ denote three-point functions in the theories of a free real scalar and a free Majorana fermion, respectively. The Ward identity (2.5) then implies that $c_T = c_B n_B + c_F n_F = c_B(n_B + n_F)$, where

$$c_B = c_F = \frac{3}{2(4\pi)^2} \tag{2.8}$$

are the central charges of the free scalar and free Majorana fermion theories. The average null energy condition constrains n_B and n_F to be nonnegative: $n_B \geq 0$ and $n_F \geq 0$, with saturation if and only if the theory is free [44–47].

In our numerical setup, it is useful to work in conventions where the two-point function of the stress tensor is unit-normalized. Let us therefore define the normalized stress tensor

$$\widehat{T} \equiv \frac{T}{\sqrt{c_T}}, \qquad \langle \widehat{T}\widehat{T} \rangle = \frac{H_{12}^2}{X_{12}^5}. \tag{2.9}$$

Three-point functions of $\widehat{T}$ are given by

$$\langle \widehat{T}\widehat{T}\widehat{T} \rangle = \lambda_B \langle \widehat{T}\widehat{T}\widehat{T} \rangle_B + \lambda_F \langle \widehat{T}\widehat{T}\widehat{T} \rangle_F, \tag{2.10}$$

where $\langle \widehat{T}\widehat{T}\widehat{T} \rangle_{B,F}$ are the corresponding 3-point functions in free theories, and

$$\lambda_B \equiv \frac{n_B}{(c_T/c_B)^{3/2}} = \frac{n_B}{(n_B + n_F)^{3/2}}, \qquad \lambda_F \equiv \frac{n_F}{(c_T/c_F)^{3/2}} = \frac{n_F}{(n_B + n_F)^{3/2}}. \tag{2.11}$$

Note that these normalized coefficients satisfy

$$\lambda_B + \lambda_F = \frac{1}{(n_B + n_F)^{1/2}} = \frac{1}{\sqrt{c_T/c_B}}. \tag{2.12}$$

Three-point correlators involving two scalars and a stress tensor are also constrained by Ward identities. For any scalar ϕ, we have

$$\langle \phi\phi\widehat{T} \rangle = -\frac{\Delta_\phi}{\sqrt{c_T/c_B}} |0,2\rangle = -\Delta_\phi(\lambda_F + \lambda_B)|0,2\rangle, \tag{2.13}$$

where $|0,2\rangle$ denotes an $SO(3)_r$ structure [37, 38]. On the right-hand side, we used the identity (2.12). The fact that λ_B, λ_F appear linearly in Ward identities is convenient for our implementation: we can treat them like any other independent OPE coefficients, with the subtlety that they appear in multiple three-point correlators.

For the rest of the paper we implicitly assume unit-normalized stress tensors and therefore dispense with the hat notation.

---

[4]This is standard but imprecise terminology, since unlike in 2d, c_T is not actually the central charge of a symmetry algebra.

2.3 Four-point structures and crossing equations

Next, we must establish a complete set of independent crossing equations for the four-point correlators in the $T\sigma\epsilon$ system.

Four-point tensor structures. To describe four-point tensor structures we use the q-basis of [35], as it does not suffer from redundancies which plague embedding-space bases [36, 39]. A q-basis four-point tensor structure is denoted by $[q_1 q_2 q_3 q_4]^{\pm}$, where $q_i \in \{-\ell_i, -\ell_i+1, \dots, +\ell_i\}$ and ℓ_i is the spin of the operator at position i. The sign $\pm$ indicates the charge of the tensor structure under exchange of the cross-ratios $z \leftrightarrow \overline{z}$.

A general four-point function can be expanded as

$$\langle \mathcal{O}_1 \mathcal{O}_2 \mathcal{O}_3 \mathcal{O}_4 \rangle = \sum_{\{q_i\}, \pm} [q_1 q_2 q_3 q_4]^{\pm} g_{\mathcal{O}_1 \mathcal{O}_2 \mathcal{O}_3 \mathcal{O}_4}^{[q_1 q_2 q_3 q_4]^{\pm}}(z, \overline{z}). \tag{2.14}$$

The functions $g_{\mathcal{O}_1 \mathcal{O}_2 \mathcal{O}_3 \mathcal{O}_4}^{[q_1 q_2 q_3 q_4]^{\pm}}(z, \overline{z})$ satisfy $g_{\mathcal{O}_1 \mathcal{O}_2 \mathcal{O}_3 \mathcal{O}_4}^{[q_1 q_2 q_3 q_4]^{\pm}}(\overline{z}, z) = \pm g_{\mathcal{O}_1 \mathcal{O}_2 \mathcal{O}_3 \mathcal{O}_4}^{[q_1 q_2 q_3 q_4]^{\pm}}(z, \overline{z})$. We refer the reader to [35] for details. We also define coordinates x, t by[5]

$$z = x + it, \quad \overline{z} = x - it. \tag{2.15}$$

Permutation symmetries of the four-point functions split into crossing transformations and kinematic permutations. The latter are permutations that preserve the cross-ratios, see [35] for details. The group Π of kinematic permutations is given by

$$\Pi_{\langle TTTT \rangle} = \{\text{id}, (12)(34), (13)(24), (14)(23)\}, \tag{2.16}$$

$$\Pi_{\langle TT\sigma\sigma \rangle} = \Pi_{\langle TT\epsilon\epsilon \rangle} = \{\text{id}, (12)(34)\}, \tag{2.17}$$

$$\Pi_{\langle T\epsilon\sigma\sigma \rangle} = \Pi_{\langle TTT\epsilon \rangle} = \Pi_{\langle T\epsilon\epsilon\epsilon \rangle} = \{\text{id}\}. \tag{2.18}$$

with analogous expressions for other orderings. Any q-basis structure can be symmetrized under Π, and we denote symmetrization of $[q_1 q_2 q_3 q_4]^{\pm}$ by $\langle q_1 q_2 q_3 q_4 \rangle^{\pm}$. Four-point functions can now be written as

$$\langle \mathcal{O}_1 \mathcal{O}_2 \mathcal{O}_3 \mathcal{O}_4 \rangle = \sum_{\{q_i\}/\Pi, \pm} \langle q_1 q_2 q_3 q_4 \rangle^{\pm} g_{\mathcal{O}_1 \mathcal{O}_2 \mathcal{O}_3 \mathcal{O}_4}^{\langle q_1 q_2 q_3 q_4 \rangle^{\pm}}(z, \overline{z}). \tag{2.19}$$

Finally, we can account for space parity by restricting the sum to those q for which the structure $\langle q_1 q_2 q_3 q_4 \rangle^{\pm}$ has the desired parity. The parity is always even in our setup, which restricts the sum $\sum_i q_i$ to be even.

We do not provide the explicit lists of the more than one hundred four-point tensor structures. Instead, we provide some counting data in table 4.

Conservation. The four-point functions must satisfy the conservation equations for T,

$$\langle \partial_\mu T^{\mu\nu} \cdots \rangle = 0. \tag{2.20}$$

These are differential equations for each four-point function involving T, which have many components due to the index ν and the spin indices of other insertions. These equations

---

[5]Note that these are different from the x, t-coordinates defined by `blocks_3d` [38].

– 8 –

	conformal	parity	permutation	bulk	line	point
$\langle TTTT\rangle$	625	313	97	5	9	8
$\langle TTT\epsilon\rangle$	125	63	63	4	3	3
$\langle TT\epsilon\epsilon\rangle$	25	13	9	2	1	0
$\langle TT\sigma\sigma\rangle$	25	13	9	2	1	0
$\langle T\epsilon\epsilon\epsilon\rangle$	5	3	3	1	0	0
$\langle T\epsilon\sigma\sigma\rangle$	5	3	3	1	0	0

Table 4. The number of tensor structures contributing to spinning correlators of the $T\sigma\epsilon$ system as conformal symmetry, parity invariance, permutation invariance, and stress-tensor conservation are imposed. For the latter, we count bulk, line, and point degrees of freedom separately.

heavily constrain the functional degrees of freedom of our four-point functions. As we explain below, it is necessary to properly account for these functional degrees when choosing which crossing equations to include in a bootstrap computation. Due to the conformal invariance of the conservation operator $\partial_\mu T^{\mu\nu}$, the left-hand side can be expanded in a basis of conformally-invariant four-point tensor structures for a correlator in which T is replaced by a vector operator.

Using the expansion (2.19) the conservation equations can then be converted to systems of first-order partial differential equations for the functions $g_{\mathcal{O}_1\mathcal{O}_2\mathcal{O}_3\mathcal{O}_4}^{\langle q_1 q_2 q_3 q_4\rangle^\pm}$.[6] Denoting the full set of such coefficient functions as a vector $\vec{g}_{\mathcal{O}_1\mathcal{O}_2\mathcal{O}_3\mathcal{O}_4}$, one can express the collection of differential equations as

$$(M_1\partial_t + M_2\partial_x + M_3)\cdot \vec{g}_{\mathcal{O}_1\mathcal{O}_2\mathcal{O}_3\mathcal{O}_4} = 0, \tag{2.21}$$

where M_i are x, t-dependent matrices. If M_1 had a left inverse, it would be possible to solve this equation given some initial data on the $t = 0$ slice. Instead, M_i has a non-trivial kernel, so in general we must specify some number of components of $\vec{g}_{\mathcal{O}_1\mathcal{O}_2\mathcal{O}_3}$ in the bulk (i.e. over all x, t) in order to solve for the remaining components in terms of $t = 0$ initial data. We refer to the former as "bulk" degrees of freedom — these are two-variable degrees of freedom in the four point function which are not constrained by the conservation equations, and which will therefore enter our bootstrap setup.

The initial data at $t = 0$ must satisfy further constraints. Let N be a matrix whose rows span the left null space of M_1. Multiplying (2.21) by N from the left, we find

$$(NM_2\partial_x + NM_3)\cdot \vec{g}_{\mathcal{O}_1\mathcal{O}_2\mathcal{O}_3\mathcal{O}_4} = 0, \tag{2.22}$$

which at $t = 0$ can be seen as a constraint that analogously reduces some of the initial data from "line" degrees of freedom to "point" degrees of freedom.

Yet further but nonetheless important constraints arise at $t = 0$ as this is a locus of enhanced symmetry: when $t = 0$ the four operators lie all on one circle. It can be shown that this forces some low-order t-derivatives of $\vec{g}_{\mathcal{O}_1\mathcal{O}_2\mathcal{O}_3\mathcal{O}_4}$ to vanish at $t = 0$: see appendix A of [35].

---

[6]Since the q-basis is defined using a choice of conformal frame, one again needs to use conformal invariance as described in section 4.3 of [41].

The complete enumeration of bulk, line, and point degrees of freedom is precisely the set of functional degrees of freedom on which we impose our crossing equations. In practice, the bulk degrees of freedom enter more or less as normal with the usual list of derivatives in our expansion, while the line and point degrees of freedom enter with just the derivatives along $z = \bar{z}$ and without any derivatives, respectively. A summary of the various degrees of freedom in each of the four-point functions can be found in table 4. The complete summary of crossing equations is provided in appendix B.

The above discussion of conservation has been quite schematic as our goal was merely to highlight the key issues. For details, we refer the reader to [19]; the analysis here is a straightforward if somewhat tedious generalization.

Linear independence of crossing equations. We must emphasize that it is strictly necessary to reduce the set of crossing equations to the minimal set of independent equations. Naïvely, one might expect that imposing additional crossing equations might at most slow down numerical computations, but ultimately would lead to equivalent and correct bounds on CFT data. In practice, we found that the presence of spurious crossing equations lead to nearly-flat directions in the SDP, begetting numerical instabilities which ultimately can lead to erroneously ruling out part of theory space. In fact, with typical parameters, it is sufficient to mistakenly impose "line equations" on the $\langle TTT\epsilon \rangle$ point degrees of freedom (i.e., introduce a handful of extra derivatives to our functional) to have the appearance of excluding the full parameter space of the Ising CFT.

3 Implementation

3.1 Generating SDPs

Semidefinite programs (SDPs) for bootstrapping correlators of σ, ϵ, T are somewhat complex, and we needed to devote significant engineering effort to generating them efficiently. Two ideas are particularly important in our implementation:

- Parallelism. Recall that SDPB scales well to several nodes and a large number of cores. (For example, each of our $\Lambda = 43$ results were computed on 16 nodes with 128 cores each, for a total of 2048 cores.) In order to ensure that *generating* SDPs did not become a bottleneck, we needed to achieve similar scaling for computing conformal blocks and assembling them into SDPB input. One way to improve parallelism is to, whenever possible, break computations into independent "chunks" that can be handled separately on different cores. However, one must also be careful to control the memory usage of each chunk, since otherwise the number of simultaneous chunks could become memory limited, degrading overall performance.

- Memoization.[7] We take care to recompute data only when necessary. For example, conformal blocks appearing in the four-point function of stress tensors $\langle TTTT \rangle$ remain unchanged when parameters like Δ_σ and Δ_ϵ are varied. Thus, we pre-compute them and store them in a central location for re-use. As another example, during an OPE scan,

---

[7]This is not a typo, see https://en.wikipedia.org/wiki/Memoization.
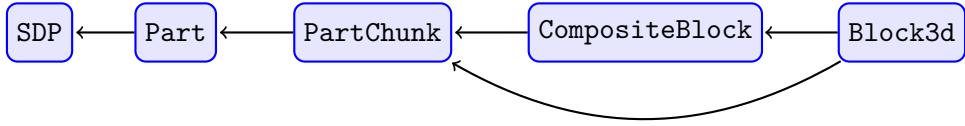
Figure 2. Elements of an SDP. Arrows indicate reverse dependencies — e.g. `Block3d`'s are needed to compute a `CompositeBlock`, while both `Block3d`'s and `CompositeBlock`'s are needed to compute a `PartChunk`.

only OPE coefficients of external operators are changing — the remaining conformal blocks and crossing equations are unchanged, so we take care to only re-compute the contribution to an SDP from the external operator channel.

To achieve these aims of parallelism and memoization, we broke the process of building the SDP into several elements, illustrated in figure 2:

- `Block3d`'s: these are the tables of conformal blocks computed by the program `blocks_3d`. Each run of `blocks_3d` is characterized by a unique `BlockTableKey`, which includes information about the representations of the external operators, a choice of four-point structure, choices of three-point quantum numbers j_{12}, j_{43}, together with settings like a choice of coordinates, recursion order, number of derivatives Λ, and precision. Different runs of `blocks_3d` can be performed in parallel. However, each run can require substantial computational resources. We describe our improvements to `blocks_3d` to take advantage of multiple cores and reduce memory usage in section 3.2.

- `CompositeBlock`'s: a `CompositeBlock` is a linear combination of `Block3d`'s with co-efficients that are rational functions of Δ. Each `CompositeBlock` is computed in a single-threaded Haskell process that reads the necessary `Block3d` files, computes the appropriate linear combination, and saves the result to a file in binary format. Be-cause these processes are single-threaded, we can run many of them at once. However, this requires keeping the memory usage of each process small. To achieve this, we implemented a streaming `Block3d` parser that parses `Block3d` data on the fly without reading whole files into memory. Also, when building linear combinations of blocks, we read a `Block3d` from a file, add it to an accumulator (with the appropriate coefficient), and then free its memory before proceeding to the next `Block3d` file.[8]

- `SDP PartChunk`'s: a `PartChunk` is the contribution to a single crossing equation from a single channel. It is a symmetric matrix whose entries are vectors of derivatives of confor-mal blocks. Each `PartChunk` can depend on several `Block3d`'s and `CompositeBlock`'s. Each `PartChunk` is computed by a single-threaded Haskell process that reads the appro-priate blocks, assembles them into a matrix, places them over a common denominator in Δ (if necessary), and writes the result to a binary file.

- `SDP Part`'s: a `Part` is the contribution of a single channel to an SDP (it is either a positivity condition, a normalization condition for the SDP, or the objective for the

---

[8]More precisely, we force evaluation of the accumulator with `deepseq`, which ensures that unneeded data will be garbage collected.

SDP). It is a symmetric matrix whose entries are concatenated vectors of derivatives of conformal blocks. Each `Part` is computed by a single-threaded Haskell process that reads the appropriate `PartChunk`'s, concatenates their matrix entries, places the whole expression over a common denominator in Δ, and writes the result to a JSON file in the appropriate format for `pmp2sdp`.

Before building an SDP, we construct a tree of required elements and their dependencies.[9] We then check whether any elements have already been built by checking for the existence of its corresponding file.[10] If an element has already been built, we prune that part of the tree, removing the element and its dependencies. This achieves the goal of "memoization" mentioned above. For example, if all of the elements of an SDP already exist, the tree becomes trivial. Finally, we convert the tree of elements to a tree of tasks. (For example, multiple `Block3d`'s corresponding to different spins `j_internal` can be combined into a single task — i.e. a single run of `blocks_3d`.)

The tasks are distributed to available cores by a custom scheduler, which ensures that we stay within the limits of memory and available CPUs, while respecting the dependency order of tasks, and attempting to minimize the total time for building the SDP. (Since `blocks_3d` is multi-threaded, the scheduler must also decide how many cores to allocate to each `blocks_3d` process.) To make the scheduler as efficient as possible, we collect time and memory data for each task, and re-use those statistics when building subsequent SDPs. Our scheduler uses an ad-hoc greedy algorithm for distributing resources to tasks, described in appendix D. This could be improved in the future by taking advantage of more sophisticated scheduling algorithms and/or open-source tools.

We give examples of the number of tasks needed to build an SDP, together with the total build time, in table 5.

3.2 Optimizing `blocks_3d`

A major contribution to SDP generation time comes from computing conformal blocks. We perform these computations using the program `blocks_3d` [37, 38], which is capable of constructing numerical approximations to arbitrary conformal blocks in $d = 3$ CFTs.

Prior to this work, `blocks_3d` has been used in [18, 34, 38, 48, 49] in bootstrap problems involving spinors and spin-1 currents, for which its performance proved to be adequate. In our setup, however, we found it necessary to implement significant improvements to the design of `blocks_3d`. This was partly due to the need to frequently generate very complex conformal blocks (for instance, $\langle \epsilon TTT \rangle$ blocks need to be generated anew for every value of Δ_ϵ), and partly due to the memory constraints of the clusters available to us.

---

[9]We give details on how we determine these dependencies in appendix C.

[10]In order for this to be robust to crashes, it is important that the creation of a file be atomic. As an example of the type of situation we would like to avoid, suppose that a node crashes in the middle of saving a `CompositeBlock` to a file, leading to a corrupted file. When we restart the computation, we would notice the existence of this file and conclude that we don't have to rebuild the corresponding `CompositeBlock`. The whole program would then crash later when trying to incorporate the corrupted `CompositeBlock` into a `PartChunk`. To avoid this situation, we always write data to a temporary file, and then `mv` the temporary file to its permanent location afterwards. The filesystem ensures that `mv` is atomic.

	fresh build	using pre-built $\langle TTTT \rangle$ blocks	during OPE scan
`Block3d`	629	79	0
`CompositeBlock`	10538	4576	0
`PartChunk`	7926	7926	53
`Part`	363	363	1
total time	4.8 hours	1.4 hours	13 seconds

Table 5. Number of each type of task needed to build an SDP with $\Lambda = 43$, together with the total build time on 16 Expanse nodes with 128 cores/node. (Note that a single `Block3d` task produces multiple block tables, corresponding to different `j_internal`'s.) In a "fresh" build (column 1), nothing is pre-built, and the total build time is 4.8 hours. During a fresh build, we store $\langle TTTT \rangle$ blocks in a central location so that they can be re-used. A build making use of these pre-built $\langle TTTT \rangle$ blocks has fewer `Block3d`'s and `CompositeBlock`'s to build, and takes 1.4 hours (column 2). Finally, during an OPE scan (after the initial SDP has already been built), we repeatedly change the OPE coefficients of external operators, but no other parts of the SDP change. Thus, no `Block3d`'s or `CompositeBlock`'s need to be re-built, and only the `Part` corresponding to the external operators (and its corresponding `PartChunk`'s) needs to be constructed again. The build time is negligible in this case.

In order to explain these improvements, we must give a brief overview of the function of `blocks_3d`. A single run of `blocks_3d` computes multiple conformal blocks, which all share the same external operator representations, four-point structure label, as well as the j_{12} and j_{43} parts of the three-point function labels. The program outputs conformal blocks with all possible values of j_{120} and j_{430}, where the index 0 refers to the exchanged operator, as well as all values of the exchanged spin j up to a cutoff $j_{\max}$.

For each conformal block, an array of derivatives is returned, with each derivative approximated by a rational function of the exchanged scaling dimension Δ. These approximations are computed using Zamolodchikov-like recursion relations derived in [37], see [10, 50–52] for earlier work. A starting point for the recursion is the function $h_{\infty,j}(z,\overline{z})$, given by a known analytic formula [37, 53], which controls the $\Delta \to \infty$ limit of the conformal block.

We denote the conformal block schematically by g. Prior to this work, the algorithm of `blocks_3d` that computes g could be described roughly as follows.

- In the first stage, the derivatives $\partial_r^m \partial_\lambda^n g$ are computed.[11] For this, the following steps are taken.

  - The derivatives $\partial_r^m \partial_\lambda^n h_{\infty,j}$ are evaluated.

  - The derivatives $\partial_r^m \partial_\lambda^n h_{\infty,j}$ are used as a starting point for the recursive computation of the derivatives $\partial_r^m \partial_\lambda^n g$. The recursions for different values of n are completely independent, and are performed in parallel.

  - For some conformal blocks, only the $\partial_r^m \partial_\lambda^0 g$ derivatives are required. In this case, only one recursion needs to be performed.

---

[11]Here, $r = \sqrt{\rho\overline{\rho}}$, $\lambda = \frac{1}{2}\log(\rho/\overline{\rho})$, and $\rho = z/(1 + \sqrt{1-z})^2$.

- In the second stage, the derivatives $\partial_r^m \partial_\lambda^n g$ are converted to the desired basis of derivatives (x, t-derivatives in our case).

Upon analyzing the performance of this algorithm as implemented in `blocks_3d`, we discovered several issues.

- The evaluation of the derivatives $\partial_r^m \partial_\lambda^n h_{\infty,j}$ (the base case) was taking up the majority of the computation time.

- The parallelization by different values of n had significant drawbacks. Firstly, the number of different values of n for derivative order Λ is roughly $\Lambda/2$, which is much smaller than the typical number of cores available on a compute node (compare $\Lambda = 43$ to a typical 128-core machine).

- The recursion part of the code used a significant amount of memory. Running $\Lambda/2$ recursions simultaneously means that the memory consumption is multiplied by a factor of around 20 for $\Lambda = 43$. This meant that such high-Λ calculations could not fit into the memory of a single machine.

- An implementation error led to all derivatives $\partial_r^m \partial_\lambda^n g$ being computed in all cases, even when only $n = 0$ was requested.[12]

To address these issues, we implemented the following optimizations.

- We significantly optimized the code that evaluates $\partial_r^m \partial_\lambda^n h_{\infty,j}$ by adding memoization of some intermediate expressions (in addition to the memoization that was already in place).

- We revamped the structure of parallelism in `blocks_3d`. Specifically, different values of n are now processed sequentially, and we parallelize the recursion step instead. This leads to much better scalability and dramatically improves memory usage by avoiding memory-intensive tasks running in parallel. This new structure also allowed us to parallelize the computation of $h_{\infty,j}$, leading to further performance gains.

- As the restructuring required a significant refactoring of the codebase, we took the opportunity to implement many further minor optimizations. In particular, some memory-intensive memoizations have been removed without any negative impact on performance.

- We fixed the implementation errors in the case when only $n = 0$ derivatives were requested.

These optimizations were implemented over a period of time of continuous development, and we were using the software the entire time. As a result, we have not performed a systematic comparison of the original version to the final version used in this project. However, in comparisons between intermediate versions we observed, for reasonably representative runs,

---

[12]This was a minor performance loss in previous studies, but it makes a significant difference in our current study.

a more than $10\times$ improvement in memory usage and a similar improvement in computation (wall) time. Note that the improvements in memory usage also mean that more `blocks_3d` instances can run on a single machine. This is parallelism with nearly perfect scaling, unlike the parallelization within a single `blocks_3d` instance,[13] further reducing the time needed to generate all conformal blocks.

3.3 Improvements to SDPB

Our work benefited from major performance improvements in SDPB 3.0.[14] Here, we give a brief summary of the main changes. One of the most expensive global operations in each SDPB iteration is computing the matrix $Q \equiv P^T P$, where

$$P \equiv \begin{pmatrix} P_1 \\ P_2 \\ \vdots \\ P_J \end{pmatrix} \tag{3.1}$$

is a dense tall matrix of multiprecision numbers, and each submatrix P_i is calculated independently from a single positivity constraint. In this section, we refer to a positivity constraint as a "block", since it usually comes from the contribution of a particular channel to a set of crossing equations.

To compute $Q = P^T P$, one naïve approach is to compute the local contribution $Q_i \equiv P_i^T P_i$ from each block independently and then call reduce-scatter among cores to aggregate the results into the global distributed matrix $Q = \sum_i Q_i$. This algorithm was originally implemented in SDPB 2.0 and worked well for smaller problems. However, it turned out to be a bottleneck as we proceeded to larger problems: reduce-scatter could take about 50% of total execution time and thus ruined performance scaling for multi-node computations (see figure 3).

In SDPB 3.0, two of us (VD and DSD) implemented a new matrix multiplication algorithm utilizing BLAS, the Chinese Remainder Theorem and MPI shared memory windows. Firstly, we now compute the contribution to Q from the blocks on each node, and then perform reduce-scatter among nodes. This requires much less communication than reduce-scatter among cores and results in much better scaling. Secondly, we made matrix multiplication on a node several times faster by utilizing highly efficient BLAS routines, which was not previously possible with multiprecision matrices. To this aim, we split each multiprecision matrix P' into several double-precision matrices $P'_{(k)}$, where P' is the part of the matrix P stored on a given node. The matrices $P'_{(k)}$ are stored in a shared memory window, so that all cores on a node work together to calculate $Q'_{(k)} = P'^T_{(k)} P'_{(k)}$ using BLAS. Then we restore the multiprecision matrix $Q' = P'^T P'$ from $Q'_{(k)}$ using the Chinese Remainder Theorem and call reduce-scatter to synchronize the result between nodes.

Figure 3 shows performance scaling of SDPB for a large-scale computation for the Ising model with stress tensors at $\Lambda = 35$. (This computation requires at least 4 Expanse nodes to fit in memory.) One can see that SDPB 3.0 is more than $2.5\times$ faster than the previous version at 4 nodes, and scales well even for 10 nodes, achieving a factor of 6 speedup in that

---

[13]This does not contradict the previously stated importance of this internal parallelization.

[14]https://github.com/davidsd/sdpb/releases/tag/3.0.0.
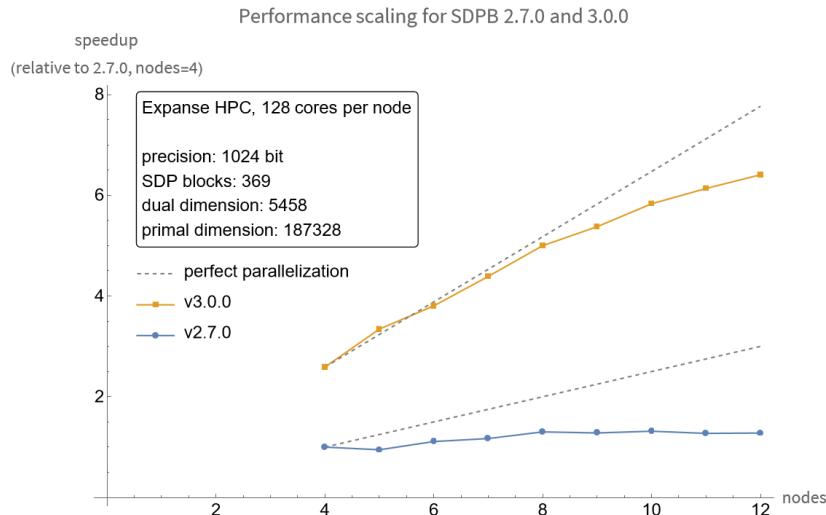
Figure 3. SDPB 2.7.0 and SDPB 3.0.0 speedup on Expanse for the Ising model with stress tensors, $\Lambda = 35$. The speedup is calculated relative to the smallest run for SDPB 2.7.0 (4 nodes). Each node has 128 cores.

case. A detailed description of the new matrix multiplication algorithm can be found at https://github.com/davidsd/sdpb/blob/3.0.0/src/sdp_solve/SDP_Solver/run/bigint_syrk/Readme.md.

3.4 Approximating conformal blocks with interpolation

In the course of this project we also found and implemented an improved approach to approximating conformal blocks using polynomials, leading to better-conditioned and more numerically stable SDPs. A commonly-used approach in the numerical bootstrap has been the following procedure [51]:

1. Truncate the radial coordinate expansion of a conformal block at some order, giving at the crossing-symmetric point an approximation $g_\Delta \approx r^\Delta \frac{P(\Delta)}{\Pi_p(\Delta-\Delta_p)}$ with a polynomial numerator $P(\Delta)$ and set of poles Δ_p.

2. Choose a subset of poles according to a parameter κ, and perform "pole-shifting", where we approximate $R(\Delta) = \frac{P(\Delta)}{\Pi_p(\Delta-\Delta_p)}$ in terms of a lower-degree rational function $S(\Delta)$ with only the subset of poles. This is usually done by imposing that some number of derivatives of $S(\Delta)$ around $\Delta = \Delta_{\min}$ and $\Delta = \infty$ should match the corresponding derivatives of $R(\Delta)$.

We found that this procedure worked poorly in practice for our SDPs at high derivative order. The "pole-shifting" approximation was not accurate enough. In order to make it more accurate, we had to significantly increase κ, but this led to larger degree polynomials, worse performance, and less numerical stability (forcing us to increase precision further, degrading performance even more).

In our new approach, we approximate conformal blocks using interpolating polynomials, with a choice of sample points specially adapted to the class of functionals we wish to describe. In practice, this leads to much more accurate approximations with much lower

degree polynomials, improving both numerical stability and performance. We will describe the details of our approach in a separate publication [54].[15]

3.5 Navigator and skydiving

The navigator function [55] and skydiving algorithm [56] are approaches to optimizing SDPs that depend on parameters $\vec{x}$ (e.g. dimensions and OPE coefficients of "external" operators). The navigator function $\mathcal{N}(\vec{x})$ is positive for disallowed points and negative for allowed points. Thus, one can then use gradient information about $\mathcal{N}(\vec{x})$ to guide the search for allowed points. The skydiving algorithm is a way to simultaneously solve for $\mathcal{N}(\vec{x})$ while moving in $\vec{x}$-space, potentially allowing one to find allowed points with a single `sdpb` run. Both ideas can potentially significantly reduce the number of `sdpb` iterations needed to map out an allowed space. However, a tradeoff is that one must typically build a very large number of conformal blocks and SDPs, since the external parameters $\vec{x}$ change many times.

In this work, we explored using skydiving with the generalized free field (GFF) navigator of [55]. Our external parameters are:

$$\vec{x} = \left(\Delta_\sigma, \Delta_\epsilon, \frac{\lambda_B}{\lambda_{TT\epsilon}}, \frac{\lambda_F}{\lambda_{TT\epsilon}}, \frac{\lambda_{\sigma\sigma\epsilon}}{\lambda_{TT\epsilon}}, \frac{\lambda_{\epsilon\epsilon\epsilon}}{\lambda_{TT\epsilon}} \right). \tag{3.2}$$

We ran the skydiving algorithm with three different objectives:

1. Minimize $\mathcal{N}(\vec{x})$ to find a point deep in a bounded feasible region.

2. Maximize Δ_σ to bound the island from the right.

3. Minimize Δ_σ to bound the island from the left.

In practice, we found that skydiving was effective at lower derivative orders $\Lambda = 11, 19$ (and partially at $\Lambda = 27, 35$). However, at higher derivative orders, the cost of generating many SDPs became prohibitive. It was more efficient to spend computational time in `sdpb` than in generating many new SDPs. Moreover, the gradient descent cannot be easily parallelized. Hence, at higher derivative orders we preferred the "cutting surface" algorithm and Delaunay triangulation approach of [32].

3.6 Code availability and supplementary material

Our code is open-source and available online. This includes the programs listed in appendix A of [34], as well as the central repository for this work:

- https://gitlab.com/davidsd/stress-tensors-3d
  commit hash: `87d8076ee36373848d5cc3f4f5ac9c99856b9ac3`.
  This repository includes setup code for constructing SDPs from T, σ, ϵ three-point structures and crossing equations, an implementation of the "scheduler" described in appendix D, versions of the OPE cutting surface and skydiving algorithms specialized to use the scheduler, as well as all the programs used to generate bounds for this work.

---

[15]The new interpolation scheme is already implemented and publicly available in the tool `pmp2sdp` that is bundled with `sdpb`.

In addition, we include the following supplementary material with this paper:

- `three_point_structures.m`: the three-point structures used in this work, in *Mathematica* format.

- `structures_3d.m`: a *Mathematica* package for manipulating three-point structures, including performing conversions between embedding-space structures, the q-basis, the $SO(3)_r$ basis, and more.

4 OPE scan and certifying disallowed points

4.1 Review: cutting surface algorithm

The crossing equations for T, σ, ϵ system, when expanded in conformal blocks, contain contributions from the external operators T, σ, ϵ themselves. Schematically, this can be written as

$$\vec{\lambda}^T \mathcal{G}_{\text{ext}}^I \vec{\lambda} + \cdots = 0, \tag{4.1}$$

where

$$\vec{\lambda} = (\lambda_B, \lambda_F, \lambda_{\sigma\sigma\epsilon}, \lambda_{\epsilon\epsilon\epsilon}, \lambda_{TT\epsilon})^T \tag{4.2}$$

is the vector of OPE coefficients involving the external operators and $\mathcal{G}_{\text{ext}}^I$ is a symmetric 5×5 matrix constructed out of derivatives of conformal blocks, and I labels different components of the crossing equations.

The basic approach in numerical bootstrap involves searching for a linear functional α_I such that, among other constraints, the matrix $\alpha_I \mathcal{G}_{\text{ext}}^I$ is positive semi-definite,

$$\alpha_I \mathcal{G}_{\text{ext}}^I \succeq 0. \tag{4.3}$$

Acting with such an α_I on the crossing equations we get

$$\vec{\lambda}^T (\alpha_I \mathcal{G}_{\text{ext}}^I) \vec{\lambda} + \cdots = 0. \tag{4.4}$$

The constraint (4.3) ensures that the first term is non-negative. The other constraints on α_I ensure that all the remaining terms are non-negative and that there is at least one positive contribution, leading to a contradiction. This idea allows us to exclude the values in the parameter space for which a suitable α_I can be found.

It has been observed in [11, 57] that the constraint (4.3) is in fact too strong. It also excludes solutions to crossing equations where there are several distinct contributions of the same form in the crossing equations,

$$\sum_i \vec{\lambda}_i^T \mathcal{G}_{\text{ext}}^I \vec{\lambda}_i + \cdots = 0. \tag{4.5}$$

However, we know that in Ising CFT there are unique operators σ, ϵ, T with the respective quantum numbers, leading to single term as in (4.1).

A better strategy is to treat the vector $\vec{\lambda}$ as a parameter similar to the scaling dimensions $\Delta_\sigma, \Delta_\epsilon$, and to try to exclude a given value of $\vec{\lambda}$ by searching for α_I that instead of (4.3) satisfies

$$\vec{\lambda}^T(\alpha_I \mathcal{G}_{\text{ext}}^I)\vec{\lambda} \geq 0. \tag{4.6}$$

Importantly, this constraint is unchanged if we rescale $\vec{\lambda}$ by a non-zero real number. Therefore, it only depends on the equivalence class $[\vec{\lambda}]$ of $\vec{\lambda}$ in $\mathbb{RP}^4$ and we should treat $[\vec{\lambda}]$ as the new parameter. If we are able to exclude all possible values of $[\vec{\lambda}]$ for a fixed $(\Delta_\sigma, \Delta_\epsilon)$, then the point $(\Delta_\sigma, \Delta_\epsilon)$ is excluded in the dimension space. As shown in [11], this procedure yields a significantly smaller allowed region for $(\Delta_\sigma, \Delta_\epsilon)$ than using (4.3).

How can one exclude all possible values of $[\vec{\lambda}]$ in a finite amount of time? The approach used in [11] was to discretize $\mathbb{RP}^1$ (their $\vec{\lambda}$ had only two components) using a finite grid. If all points in the grid are excluded, one can be reasonably confident that there are no allowed points in $\mathbb{RP}^1$ if the grid is fine enough. This approach worked well in [51] because their parameter space $\mathbb{RP}^1$ was one-dimensional. In our case the parameter space is $\mathbb{RP}^4$ and discretization appears computationally infeasible.

This problem was solved in [32] with the "cutting surface" algorithm. The key observation is simply that a finite set of functionals $\alpha_I^{(k)}$, $k = \{1, 2, \ldots, N\}$, can be enough to exclude the full parameter space $\mathbb{RP}^4$. Indeed, existence of a functional $\alpha_I^{(k)}$ implies that all $[\vec{\lambda}]$ which satisfy

$$\vec{\lambda}^T(\alpha_I^{(k)}\mathcal{G}_{\text{ext}})\vec{\lambda} \geq 0 \tag{4.7}$$

are excluded. This condition defines a closed set E_k, and it might very well be that the union of all E_k covers $\mathbb{RP}^4$,

$$\bigcup_{k=1}^N E_k = \mathbb{RP}^4. \tag{4.8}$$

If (4.8) holds, then all $[\vec{\lambda}]$ are excluded. If (4.8) doesn't hold, then we can take a $[\vec{\lambda}_{N+1}]$ that is not in $\bigcup_{k=1}^N E_k$, i.e.

$$\vec{\lambda}_{N+1}^T(\alpha_I^{(k)}\mathcal{G}_{\text{ext}})\vec{\lambda}_{N+1} < 0, \quad \forall k \in \{1, 2, \ldots, N\}, \tag{4.9}$$

and try to construct a linear functional $\vec{\alpha}_I^{(N+1)}$ such that (along with all the other positivity constraints for other contribution to the crossing equations)

$$\vec{\lambda}_{N+1}^T(\alpha_I^{(N+1)}\mathcal{G}_{\text{ext}})\vec{\lambda}_{N+1} \geq 0. \tag{4.10}$$

If such a functional $\alpha_I^{(N+1)}$ is found, we can add it to the list, increment N, and check (4.8) again. If a functional cannot be found, it means that $[\vec{\lambda}_{N+1}]$ is allowed and so are the current values of $(\Delta_\sigma, \Delta_\epsilon)$. Typically, this process converges in a number of steps which is much smaller than the number of points in a fine enough grid on $\mathbb{RP}^4$. It also has the benefit of not relying on a discretization.

The technical content of the cutting surface algorithm [32] is in the procedure for finding a solution $[\vec{\lambda}_{N+1}]$ to (4.9) or determining that no solution exists.[16] Abstractly, this can be formulated as the following problem: given quadratic forms $Q_0, Q_1, \ldots, Q_N$, determine whether there exists a solution $\vec{\lambda} \in \mathbb{R}^{n+1}$ to the equations

$$\vec{\lambda}^T Q_k \vec{\lambda} < 0, \quad k \in \{0, 1, 2, \ldots, N\}. \tag{4.11}$$

This is a quadratically-constrained quadratic problem; this class of problems is in general NP-hard. For this reason, the cutting-surface algorithm of [32] uses a number of methods to try to solve (4.11). If no solution is found by these methods, the algorithm proceeds assuming that no solution exists, but this decision is based on heuristics and is not rigorous.

In this work we developed an efficient algorithm that can rigorously certify that no solutions to (4.11) exist. This does not contradict NP-hardness of the problem since our algorithm is not guaranteed to construct a certificate. However, in all cases when the cutting-surface algorithm of [32] terminated with "no solution found", our algorithm was able to certify that no solutions exist. Therefore, all disallowed points in our plots have been ruled out rigorously.

Before we describe the algorithm, we note that in practice while the quadratic forms $Q_1, \ldots, Q_N$ are given by the functionals $\alpha_I^{(k)}$ as

$$Q_k = \alpha_I^{(k)} \mathcal{G}_{\text{ext}}^I, \tag{4.12}$$

the quadratic form Q_0 is specified manually to restrict the search to an ellipse in $\mathbb{RP}^{n+1}$. This bounding ellipse is chosen conservatively to include the region where the allowed points $[\vec{\lambda}]$ are expected to be. See section 5.2 for details. In essence, one of our assumptions is that if any $[\vec{\lambda}]$ is allowed at all, it is deep inside the region defined by $\vec{\lambda}^T Q_0 \vec{\lambda} < 0$. The technical meaning of this will be clarified below.

4.2 Certification algorithm

In what follows we will use the notation $Q(\lambda) = \lambda^T Q \lambda$ for quadratic forms. Since Q_0 is specified by us and is chosen so that $Q_0(\lambda) < 0$ represents an ellipsoid, it has signature $(1, n)$. By a change of basis, we can assume that

$$Q_0(\lambda) = -t^2 + y^2, \tag{4.13}$$

where $\lambda = (t, y)$ with $t \in \mathbb{R}$ and $y \in \mathbb{R}^n$.

Recall that we are looking for a solution $\lambda \in \mathbb{R}^{n+1}$ to

$$Q_k(\lambda) < 0, \quad k = 0, 1, \ldots N. \tag{4.14}$$

Any solution satisfies $Q_0(\lambda) = -t^2 + y^2 < 0$. Therefore, $t \neq 0$ and by rescaling x we can always set $t = 1$. Therefore, it is enough to find a solution $y \in \mathbb{R}^n$ to

$$Q_k((1, y)) < 0, \quad k = 1, \ldots, N, \tag{4.15}$$

such that $|y| < 1$ or prove that none exists.

---

[16]When several solutions to (4.9) can be found, the cutting surface algorithm will try to select a solution that is the furthest away from the sets E_k in order to speed up the convergence of the whole process.

We can write

$$Q_k((1, y)) = c_k + \beta_k(y) + w_k(y), \tag{4.16}$$

where $c_k \in \mathbb{R}$, β_k is a linear functional and w_k is a quadratic form. It was observed in [32] that in bootstrap applications the surfaces $Q_k((1, y)) = 0$ are usually approximately flat inside the ball $|y| < 1$. In other words, the coefficients in the quadratic form w_k are much smaller than c_k and b_k, and for $|y| < 1$ the contribution of $w_k(y)$ to $Q_k((1, y))$ is subleading. Neglecting $w_k(y)$ would be useful since that would turn (4.15) into linear inequalities.

Since our goal is to construct a rigorous proof that no solutions to (4.15) exist, we cannot simply neglect $w_k(y)$. Instead, we will search for a rigorous linear lower bound on Q_k of the form

$$Q_k((1, y)) \geq c_k + \delta c_k + \beta_k(y). \tag{4.17}$$

Let

$$\mathcal{A} = \{y \in \mathbb{R}^n \,|\, |y| < 1 \text{ and } Q_k((1, y)) < 0, \quad k = 1, \ldots, N\} \tag{4.18}$$

be the set of solutions to (4.15) (our goal is to show that $\mathcal{A} = \emptyset$). Suppose we can show that

$$w_k(y) \geq \delta c_k, \quad \forall y \in \mathcal{A}. \tag{4.19}$$

Then (4.17) holds on $\mathcal{A}$ and therefore

$$\mathcal{A} \subseteq \mathcal{B}[\delta c] = \{y \in \mathbb{R}^n \,|\, c_k + \delta c_k + \beta_k(y) < 0, \quad k = 1, \ldots, N\}. \tag{4.20}$$

The set $\mathcal{B}[\delta c]$ is defined using linear inequalities, and therefore can be studied using linear programming. In particular, it is easy to check whether $\mathcal{B}[\delta c] = \emptyset$. If we can show that $\mathcal{B}[\delta c] = \emptyset$, this would constitute a proof of $\mathcal{A} = \emptyset$. Our strategy will therefore be to construct various sets of coefficients δc_k for which (4.19) holds. Our notation $\mathcal{B}[\delta c]$ stresses the fact that $\mathcal{B}[\delta c]$ depends on the choice of the δc_k.

A simple choice is to set

$$\delta c_k = \delta c_k^0 \equiv \inf_{|y| < 1} w_k(y) = \min\{\lambda_{k,\min}, 0\}, \tag{4.21}$$

where $\lambda_{k,\min}$ the minimal eigenvalue of the matrix that defines w_k. With this choice (4.19) is obviously true, and therefore $\mathcal{A} \subseteq \mathcal{B}[\delta c^0]$. Typically, we find that the set $\mathcal{B}[\delta c^0]$ is not empty. It is, however, usually much smaller than the unit ball $|y| < 1$. This allows us to improve our choice of δc_k.

We construct an improved set of constants δc_k^1 as follows. Let $\lambda_{k,i}$ and $e_{k,i}$ be the eigenvalues and the eigenvectors of w_k. We can then write

$$w_k(y) = \sum_i \lambda_{k,i}(e_{k,i} \cdot y)^2 \geq \delta c_k^1, \tag{4.22}$$

$$\delta c_k^1 \equiv \sum_{i:\lambda_i > 0} \lambda_{k,i} \inf_y (e_{k,i} \cdot y)^2 + \sum_{i:\lambda_i < 0} \lambda_{k,i} \sup_y (e_{k,i} \cdot y)^2. \tag{4.23}$$

We would like this to hold on $\mathcal{A}$. Since $\mathcal{A} \subseteq \mathcal{B}[\delta c^0]$, this can be achieved by taking the sup and the inf over $\mathcal{B}[\delta c^0]$. Note that $(e_{k,i} \cdot y)$ is a linear function on the convex set $\mathcal{B}[\delta c^0]$, so its image is an interval which can be computed using linear programming. From this, $\inf_{y \in \mathcal{B}[\delta c^0]}(e_{k,i} \cdot y)^2$ and $\sup_{y \in \mathcal{B}[\delta c^0]}(e_{k,i} \cdot y)^2$ can be easily obtained.[17]

In the majority of cases when the cutting surface algorithm of [32] does not find a solution, we observe that the set $\mathcal{B}[\delta c^1]$ is empty. Since $\mathcal{A} \subseteq \mathcal{B}[\delta c^1]$, this proves that $\mathcal{A} = \emptyset$ and certifies the absence of solutions. When the set $\mathcal{B}[\delta c^1]$ is non-empty, we can construct δc^2 from $\mathcal{B}[\delta c^1]$ just as δc^1 is constructed from $\mathcal{B}[\delta c^0]$. If $\mathcal{B}[\delta c^2]$ is non-empty, we can continue to construct δc^3 and so on. With rare exceptions discussed below, we found in practice that this process terminates with an empty set $\mathcal{B}[\delta c^m]$ for some $m > 0$. The largest value of m that we observed was $m = 3$.

Out of the 38 disallowed points in $(\Delta_\sigma, \Delta_\epsilon)$ space that we computed at $\Lambda = 43$, 30 points were certified by $\mathcal{B}[\delta c^1] = \emptyset$, 1 point by $\mathcal{B}[\delta c^2] = \emptyset$, and 2 points by $\mathcal{B}[\delta c^3] = \emptyset$. This produces certificates for 33 out of 38 points.

In these cases the sets $\mathcal{B}[\delta c^m]$ are typically quite small and deep in the interior of the unit ball $|y| < 1$. However, for the 5 remaining points $\mathcal{B}[\delta c^0]$ is an unbounded set, and δc^1 cannot be constructed. It appears that this happens when the cutting surface algorithm quickly excluded the majority of the unit ball $|y| < 1$, and then spent some time ruling out a boundary region close to $|y| = 1$.

To deal with such situations, we use convex quadratic optimization[18] to compute

$$r_{\min} \equiv \inf_{y \in \mathcal{B}[\delta c^0]} |y|. \tag{4.24}$$

We found that in all 5 cases at $\Lambda = 43$ when $\mathcal{B}[\delta c^m]$ iterations fail, $r_{\min}$ is at least 0.99. Since $\mathcal{A} \subseteq \mathcal{B}[\delta c^0]$, this shows that any solution to (4.15) satisfies $|y| > 0.99$.

Recall that the ball $|y| < 1$ is the bounding ellipse that we choose conservatively to restrict the search region in the OPE space. In other words, we are assuming that all solutions, if any, lie deep inside the unit ball $|y| < 1$. In particular, we assume that no solution can satisfy $|y| > 0.99$. Therefore, the statement that all solutions satisfy $|y| > 0.99$ is a proof that no solutions exist under our assumptions.

5 Results

5.1 The $\langle TTTT \rangle$ system

We first present new results in the system of only correlators of T, i.e. constraints on $\langle TTTT \rangle$. This was previously studied in [19], where the authors found universal bounds for all local unitary 3d CFTs on the spectrum of low-lying operators as well as the surprising presence of features seemingly linked to the 3d Ising CFT in particular. As we have a completely new implementation of this system (which is a subset of our full T, σ, ϵ system) and have

---

[17]We also implemented two minor improvements on the value of δc^1. Firstly, it is guaranteed that $(e_{k,i} \cdot y)^2 \le 1$ on $\mathcal{A}$, so we can replace $\sup_{y \in \mathcal{B}[\delta c^0]}(e_{k,i} \cdot y)^2$ by $\min\{\sup_{y \in \mathcal{B}[\delta c^0]}(e_{k,i} \cdot y)^2, 1\}$. Secondly, one can replace $\sum_{i:\lambda_i < 0} \lambda_{k,i} \sup_y(e_{k,i} \cdot y)^2$ by $\max\left\{\sum_{i:\lambda_i < 0} \lambda_{k,i} \sup_y(e_{k,i} \cdot y)^2, \delta c_k^0\right\}$.

[18]We used the function `QuadraticOptimization` in `Mathematica`.
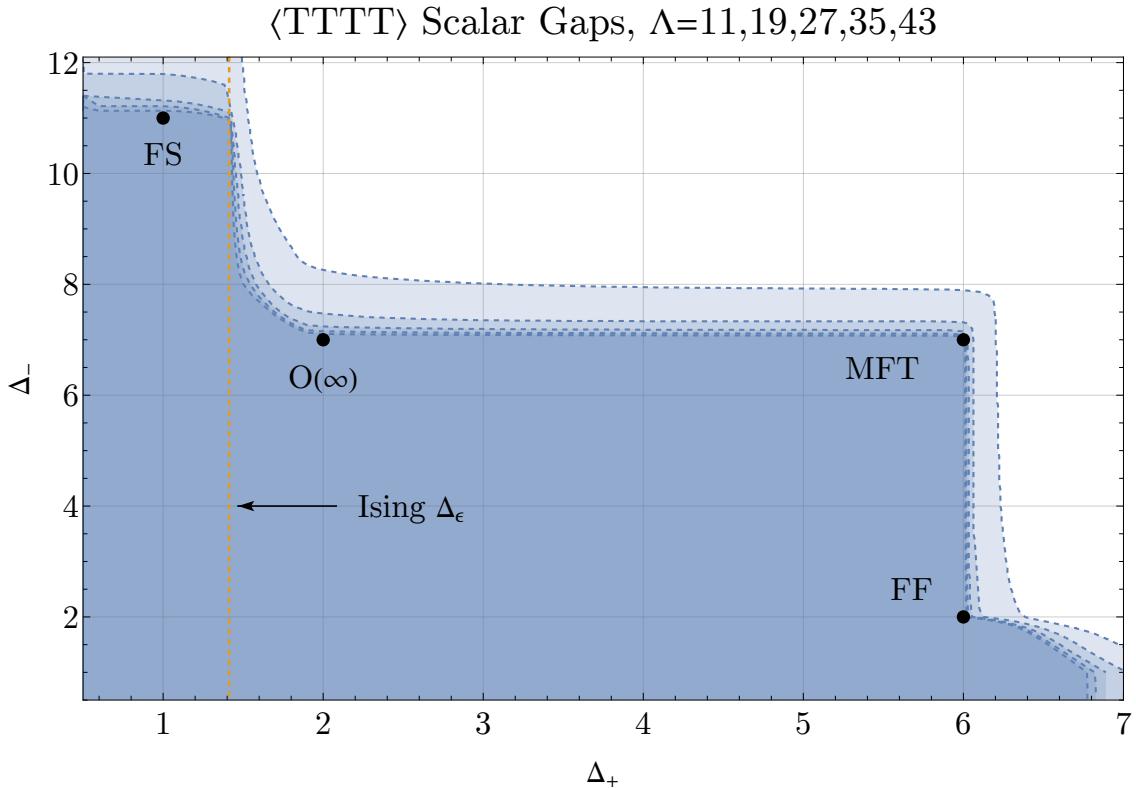
Figure 4. Bound on the allowed gaps for parity-even and -odd scalars in the single-correlator $\langle TTTT \rangle$ system. Successively interior regions correspond to higher derivative order Λ. We can see that by $\Lambda = 43$ the bounds are fairly well converged and therefore are hard to distinguish visually at this scale. The known values for these gaps in the free scalar (FS) theory, Majorana free fermion (FF) theory, stress tensor mean-field theory (MFT), $O(N)$ CFT in the large-N limit ($O(\infty)$), and Ising CFT are indicated.

the target of the 3d Ising CFT, it's therefore worthwhile to dwell briefly on this T-only system. The results serve to validate the correctness of our implementation, to improve and refine the universal numerical results of [19], and to reveal what more these universal bounds seem to know about the 3d Ising CFT.

5.1.1 Bounds on scalar gaps

The headline result of [19] was the discovery of universal upper bounds on the dimensions of the leading scalars, both parity even and odd, for local unitary 3d CFTs. In particular, [19] found a contour in the space of (Δ_+, Δ_-) ($\pm$ referring to the respective space parities) up to the derivative order of $\Lambda = 19$. We reproduce and improve on this result up to $\Lambda = 43$ in figure 4. This contour features sharp discontinuities at various notable CFTs such as the free Majorana fermion CFT and the Ising CFT, as well as the stress tensor mean-field theory. We refer to [19] for further details.

We obtained the result in figure 4 with two grids of binary searches, namely by fixing Δ_+ while searching over Δ_- and vice versa. The fixed values sample a range from $\frac{1}{2} + 10^{-6}$ (just
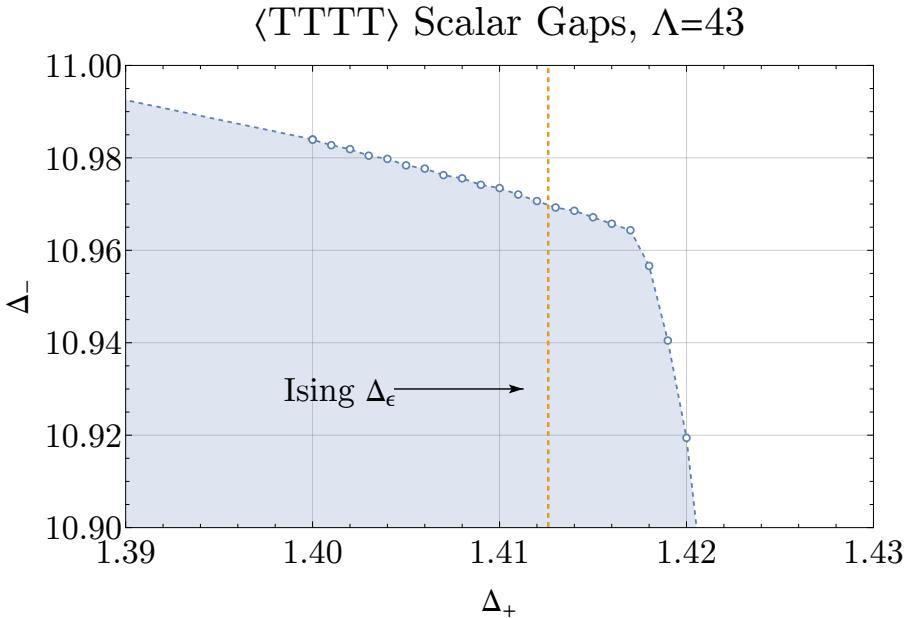
Figure 5. Inset around the Ising CFT data of the $\Lambda = 43$ bound on the allowed gaps for parity-even and -odd scalars in the single-correlator $\langle TTTT \rangle$ system; see figure 4 for the full bound. The known value for the Δ_+ gap in the Ising CFT is indicated. The markers indicate data points.

above the unitarity bound) to where there are no longer any allowed points. The only gap assumptions for this bound are Δ_+ and Δ_- in parity-even and parity-odd scalar channels.

We see that our results both agree with and improve upon the prior result of [19]. Notably, the discontinuity near the Ising CFT persists to $\Lambda = 43$, and by fixing $\Delta_+ = 1.41262528$, i.e. our central value of Δ_ϵ in the Ising CFT, we find the upper bound $\Delta_- \lesssim 10.97$. We show a zoomed view of this bound in figure 5. This is close to expectations based on the free scalar value 11, and it's tempting to speculate that the Ising CFT nearly saturates this universal bound on Δ_-. We will have more to say in the next section.

In [19] the authors also noted an intriguing feature that appears at $(\Delta_+, \Delta_-) = (7, 1)$ which seemed to not be any known CFT. Even at $\Lambda = 27$, which is our first step beyond the previous work's numerical order, the bound gets stronger and leaves $(7, 1)$ disallowed. We therefore conclude that this neighborhood of $\Delta_+ > 6$ is likely to not contain any physical CFTs and that the presence of the feature in the bound likely amounts to some low-derivative order numerical artifact.[19]

5.1.2 Bounds on OPE coefficients

Given the presence and persistence of an Ising kink in these universal bounds of local unitary 3d CFTs, one should wonder how much more the pure $\langle TTTT \rangle$ system can tell us about the Ising CFT. For example, can we see the Ising CFT in bounds of OPE coefficients? Is

---

[19]Indeed, the authors of the previous work note that this region is disallowed if one assumes that the same leading parity even scalar also appears in the OPE of $\mathcal{O}_- \times \mathcal{O}_-$, i.e. the leading odd scalar with itself. So for it to be disallowed even without such an inoffensive assumption would lend credence to this region not having much physical interest.
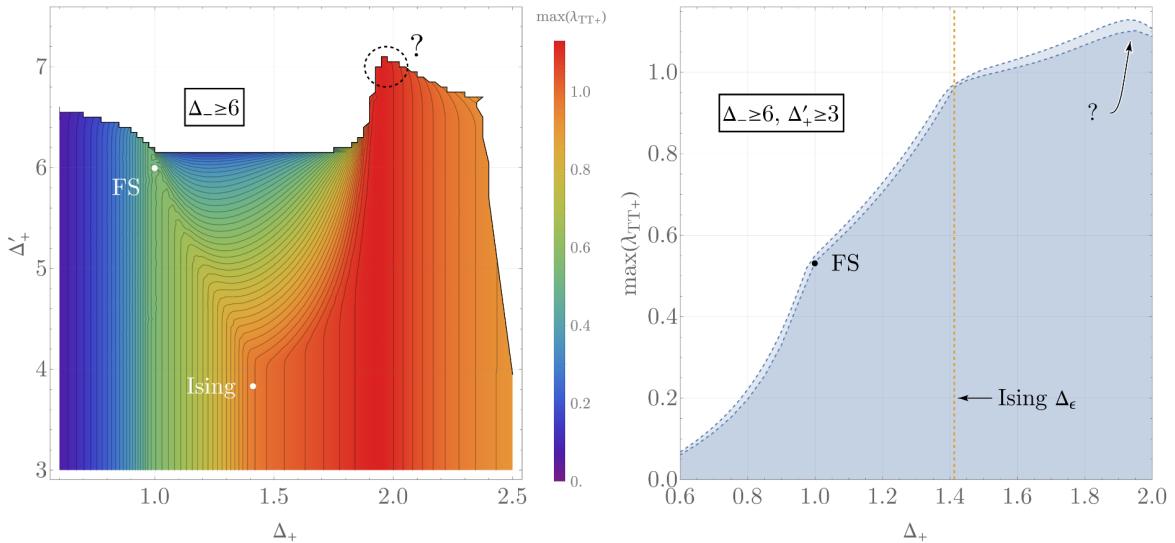
Figure 6. Results from OPE maximization of the coefficient λ_{TT+}, corresponding to the leading parity-even scalar, as a function of its dimension Δ_+ and the gap to the next parity-even scalar Δ'_+, subject to the assumption that the leading parity-odd scalar has dimension $\Delta_- \geq 6$. The left-hand plot presents the results for $\Lambda = 19$ as a contour plot, and the right-hand plot presents the results for $\Lambda = 19, 27$ as a line plot for the slice of $\Delta'_+ \geq 3$. In both, we highlight the known solution of the free scalar (FS) theory,[21] the best-known numerical bootstrap results for the Ising CFT, and a third mysterious feature.

Ising extremal in the space of CFTs, and how much so? To answer this, we computed upper bounds on the OPE coefficient of the leading even scalar λ_{TT+},[20] corresponding to $\lambda_{TT\epsilon}$ for the Ising CFT. This coefficient was recently studied non-rigorously by both the fuzzy sphere [24] and the 5-point bootstrap [27, 42], but to our knowledge had not been studied rigorously by the 4-point bootstrap before.

We computed the upper bound on λ_{TT+} as a function of Δ_+ and Δ'_+, where Δ'_+ is the gap to the next-to-leading parity-even scalar. We made a conservatively Ising-compatible assumption of $\Delta_- \geq 6$. Our results are presented in figure 6.

Let us focus on the slice with the assumption $\Delta'_+ \geq 3$. Remarkably, we see that even with these few assumptions the upper bounds on λ_{TT+} show sharp and distinct features. We first see a discontinuity at $\Delta_+ = 1$ which exactly corresponds to the free scalar theory including the known value of the OPE coefficient; so the free scalar saturates these bounds.[21] Then, for $\Delta_+ \sim 1.41$ we have another sharp feature, evidently corresponding to the Ising CFT. The upper bound at this Δ_+ is consistent with previous estimates of $\lambda_{TT\epsilon}$ in the Ising CFT in [24, 27, 42]. As we will see in the next section, the bound is in fact nearly saturated by the high-precision value of $\lambda_{TT\epsilon}$ in the Ising CFT!

---

[20]See equation (2.2) for the explicit statement of the conventions.

[21]We note that this kink is present as a vertical line on the full (Δ_+, Δ'_+) plot all the way to a gap assumption of $\Delta'_+ > 6$. The careful reader may worry that this seems to be inconsistent with the free scalar theory which has an operator ϕ^4 of dimension 2. However, such an operator does not and cannot appear in the OPE of $T \times T$; equivalently $\langle TT\phi^4 \rangle = 0$. Indeed, Wick's theorem factorizes $\langle TT\phi^4 \rangle = 0$ into two point functions $\langle T\phi^2 \rangle^2$, which then must vanish by conformal symmetry. The leading scalar that appears in this OPE is therefore of the form T^2, of dimension 6, which is exactly in line with our bound.

We can see more evidence to this effect even without the exact value of this OPE coefficient. Returning to the full (Δ_+, Δ'_+) plot, note that this putative Ising OPE kink disappears roughly around $\Delta'_+ \sim 3.9$, which is just above where the known Ising CFT would lie, with $\Delta_{\epsilon'} = 3.82951(61)$ [58]. This further underlines that these features correspond to the Ising CFT, and that it appears to be extremal — that is, special enough to be found even without assumptions of symmetries in this universal system of purely stress tensors, in multiple ways. One might wonder if this is a feature of the Ising CFT itself, or if there are other CFTs which are similarly extremal.

On this topic, we note that there appears to be a third such feature in the OPE coefficient bound, denoted by "?". It appears to correspond to roughly $(\Delta_+, \Delta'_+) \sim (1.9, 7)$, which isn't obviously related to any known CFT. Some guesses are that it is related to the $\mathcal{N} = 2$ super-Ising CFT, an $N = 4$ Gross-Neveu-Yukawa CFT, or an $O(N)$ model at some large value of N, which have Δ_+ near 1.9 [34, 59]; however these theories do not appear consistent with the Δ'_+ gap of 7 [49, 60]. It could also be possible that this feature is a numerical effect which lacks any physical correspondence. Regardless, this mysterious feature warrants further investigation.

5.2 Mixed correlators of T, σ, ϵ

Now we will move on to the main focus of this work, which is to study the system of T, σ, ϵ correlators, computing bounds on the operator dimensions $(\Delta_\sigma, \Delta_\epsilon)$ and external OPE coefficients $\vec{\lambda} = (\lambda_B, \lambda_F, \lambda_{\sigma\sigma\epsilon}, \lambda_{\epsilon\epsilon\epsilon}, \lambda_{TT\epsilon})$. We make the (very conservative) gap assumptions listed in table 6.

At lower derivative orders we computed bounds following the navigator and skydiving approach described above. We found the minimum of the navigator $\mathcal{N}_{\min}$ and maximum and minimum values of Δ_σ across the space of external data. For example, at $\Lambda = 11, 19$ we obtained the following results for the $\vec{x}$ vector defined in (3.2):

$\Lambda = 11$:

$$\mathcal{N}_{\min} : (0.5181484162, 1.412527721, 1.063394763, 0.01494714711, 1.103578160, 1.607303854),$$
$$\max \Delta_\sigma : (0.5182851826, 1.413917985, 1.061046759, 0.01527714250, 1.100931019, 1.606494761),$$
$$\min \Delta_\sigma : (0.5181429340, 1.412553256, 1.063390426, 0.01490640017, 1.103509587, 1.607512121).$$

$\Lambda = 19$:

$$\mathcal{N}_{\min} : (0.5181487291, 1.412622963, 1.063275748, 0.01491576190, 1.103369155, 1.607484793),$$
$$\max \Delta_\sigma : (0.5181534635, 1.412679013, 1.063183417, 0.01492340043, 1.103256933, 1.607464832),$$
$$\min \Delta_\sigma : (0.5181429340, 1.412553256, 1.063390426, 0.01490640017, 1.103509587, 1.607512121).$$

To compute bounds at higher derivative order, we followed a similar procedure to the computations of bootstrap islands in the $O(2)$ model [32], $O(3)$ model [33], and GNY models [34]. For each point in $(\Delta_\sigma, \Delta_\epsilon)$-space, we use the OPE "cutting surface" algorithm of [32] to find an allowed direction for the vector $\vec{\lambda}$. The output of this algorithm is either an equivalence class $[\vec{\lambda}] \in \mathbb{RP}^4$ for which the corresponding SDP is primal feasible, or evidence

$\mathbb{Z}_2$	parity	ℓ	gap assumption	3d Ising
$+$	$+$	0	$\Delta \geq 3$	$\Delta = 3.82951(61)$ [58]
$-$	$+$	0	$\Delta \geq 3$	$\Delta = 5.262(89)$ [58]
$+$	$-$	0	$\Delta \geq 3$	$\Delta \gtrsim 10$ [21]
$+$	$+$	2	$\Delta \geq 4$	$\Delta = 5.499(17)$ [58]
otherwise			$\tau \geq 1 + 10^{-6}$	$\tau \gtrsim 1.0226$ [12]

Table 6. Our gap assumptions for operators other than σ, ϵ, T, used for computing the allowed island in the 3d Ising CFT. Essentially, we assume that all scalars (other than σ, ϵ) are irrelevant, and that there is a gap of 1 above the stress tensor. We also assume a small twist gap in all other sectors ($\tau = \Delta - \ell$). In the right-hand column, we display the expected actual values for these gaps in the 3d Ising CFT. In the last row of the right-hand column, the twist gap comes from the $\mathbb{Z}_2$-even, parity-even, spin-4 sector, which has $\Delta_{C_{\mu\nu\rho\sigma}} = 5.022665(28)$. Clearly our gap assumptions are very conservative.

that no such $[\vec{\lambda}]$ exists (see section 4 for details). In the former case, we say that the point $(\Delta_\sigma, \Delta_\epsilon)$ is "allowed", and in the latter case we say it is "disallowed."

We use the Delaunay triangulation search algorithm of [32] to choose points in $(\Delta_\sigma, \Delta_\epsilon)$-space to test for allowed/disallowed status, until a reasonably clear picture of the allowed island emerges. To assist the Delaunay search, it is helpful to use affine coordinates where the allowed island is expected to be roughly disc-shaped. We determine such affine coordinates by working our way up in derivative order Λ. At a given value of Λ, we find the allowed region and use principal component analysis to find coordinates that make it roughly disc-shaped. We then use these coordinates for the Delaunay search at the next higher value $\Lambda' > \Lambda$. As an example, the affine coordinates (x, y) determined from the $\Lambda = 35$ allowed region (and used in the $\Lambda = 43$ Delaunay search) are defined by

$$\Delta_\sigma = 0.5181488106917789 + 8.496193518377354 \cdot 10^{-8} x - 5.774118868365222 \cdot 10^{-9} y,$$
$$\Delta_\epsilon = 1.412625302544141 + 1.061340020903086 \cdot 10^{-6} x + 4.622272819035116 \cdot 10^{-10} y.$$
$$(5.1)$$

In addition, the Delaunay search requires some initial allowed and disallowed points. We determined these by a combination of guesswork, using results at lower Λ, and in some cases, results from using skydiving to minimize the GFF navigator function (see section 3.5).[22] Our allowed and disallowed points in (x, y)-space for $\Lambda = 35, 43$ are illustrated in the affine coordinates (5.1) in figure 7.

The allowed islands for $\Lambda = 19, 27, 35, 43$ are illustrated in figure 1 in the introduction, where we also compare to the allowed island from the σ-ϵ system at $\Lambda = 43$ [11].

The OPE cutting surface algorithm requires an initial bounding ellipse to work efficiently. We determine it by working our way up in derivative order Λ, with a method similar to our strategy for finding affine coordinates (x, y). At a given value of Λ, we collect together all the

---

[22]As discussed in section 3.5, we chose not to use skydiving to determine the allowed islands at $\Lambda = 35$ and $\Lambda = 43$ for performance reasons. Skydiving requires building many SDPs corresponding to different values of $(\Delta_\sigma, \Delta_\epsilon)$. Furthermore, these cannot be built in parallel since the algorithm is sequential. Given the cost of building a single SDP, we found it was more efficient to use the methods of [32] for our problem.
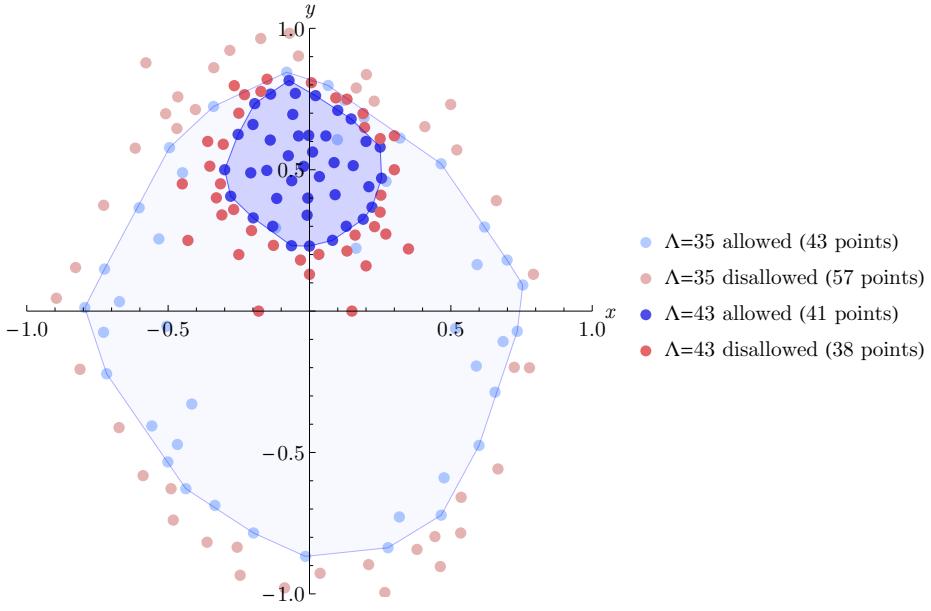
Figure 7. Allowed and disallowed points with gaps given in table 6, in the affine frame (5.1) that makes the allowed islands roughly disc-shaped, computed at $\Lambda = 35, 43$. Note that disallowed points at $\Lambda = 35$ are also disallowed at $\Lambda = 43$. Similarly, all allowed points at $\Lambda = 43$ are allowed at $\Lambda = 35$. To aid visualization, we also display the convex hulls of the allowed points, but note that this is slightly smaller than the true "allowed region.".

allowed points in OPE space $[\vec{\lambda}]$ and use principal component analysis (in an affine patch of $\mathbb{RP}^4$) to find an ellipse that contains them all. Then we expand this ellipse by an $O(1)$ factor, say 4. Finally, we use the expanded ellipse as the bounding ellipse for the next higher value $\Lambda' > \Lambda$. As a consistency check, we observe afterwards that the allowed OPE points $[\vec{\lambda}]$ at the higher Λ' cluster in a small region deep in the interior of the bounding ellipse, indicating that the results are independent of the choice of bounding ellipse. As an example, the bounding ellipse at $\Lambda = 43$ is defined by $\vec{\lambda}^T Q_0 \vec{\lambda} \leq 0$, where Q_0 is given by the matrix

$$
\begin{pmatrix}
0.37440036039553 & 0.37933721733773 & -0.27957642579931 & -0.0020965065873632 & -0.091902746129379 \\
0.37933721733773 & 0.40707588192755 & -0.28259224067795 & 0.0011378561741937 & -0.099437358761553 \\
-0.27957642579931 & -0.28259224067795 & 0.20879627066583 & 0.0016234621598708 & 0.068492801656868 \\
-0.0020965065873632 & 0.0011378561741937 & 0.0016234621598708 & 0.00070008679295711 & -0.00070446545208272 \\
-0.091902746129379 & -0.099437358761553 & 0.068492801656868 & -0.00070446545208272 & 0.024760700718227
\end{pmatrix} .
$$

$$(5.2)$$

In figure 8, we show a plot of the allowed points $[\vec{\lambda}]$ at $\Lambda = 43$, compared to this bounding ellipse (which is a sphere of radius 4 in the coordinates of the plot). As claimed, all the allowed points are deep in the interior.

As mentioned above, each allowed point in $(\Delta_\sigma, \Delta_\epsilon)$-space comes with an equivalence class $[\vec{\lambda}] \in \mathbb{RP}^4$, determined by the OPE cutting surface algorithm. Next, for each allowed point, we compute upper and lower bounds on the norm-squared $|\vec{\lambda}|^2$ using the standard procedure for bounding OPE coefficients [61]. This results in pairs of OPE coefficient vectors $\vec{\lambda}_u, \vec{\lambda}_l$ with
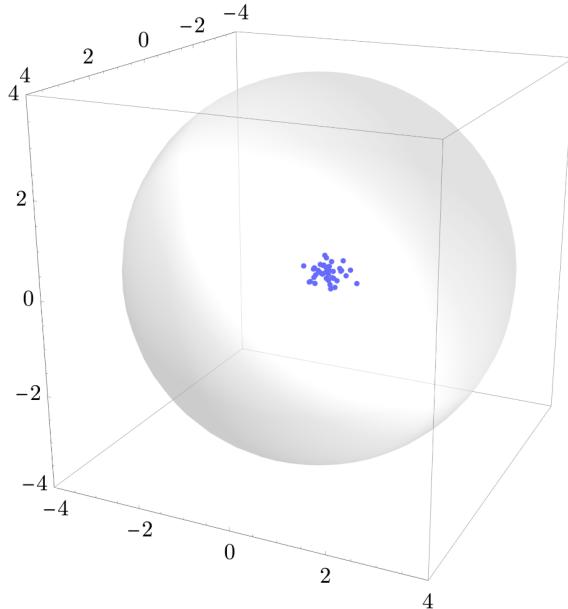
Figure 8. Allowed values of $[\vec{\lambda}]$ found at $\Lambda = 43$, compared to the initial bounding ellipse, in coordinates where the bounding ellipse is a sphere of radius 4. Here, we show just 3 out of 4 coordinates of $[\vec{\lambda}]$ — the other coordinate is similar. All the allowed points are deep inside the bounding ellipse.

norms given by the upper/lower bounds on $|\vec{\lambda}|$. Collecting together the data $(\Delta_\sigma, \Delta_\epsilon, \vec{\lambda}_u)$ and $(\Delta_\sigma, \Delta_\epsilon, \vec{\lambda}_l)$, we obtain a cloud of allowed points in $\mathbb{R}^7$. (We actually have allowed line segments between each pair of $\vec{\lambda}_l$ and $\vec{\lambda}_u$, but in practice, $|\vec{\lambda}_l - \vec{\lambda}_u|$ is tiny compared to the differences between $\vec{\lambda}$ at different points in dimension space, so the distinction is not important.) The bounds on $|\vec{\lambda}|^2$ come with extremal functionals containing interesting information about the spectrum of the 3d Ising CFT, which we will report on in future work.

The shape of the cloud of allowed points in $\mathbb{R}^7$ suggests that we can obtain reasonable bounds on the components $(\Delta_\sigma, \Delta_\epsilon, \vec{\lambda})$ by simply taking their minimum and maximum values over the cloud of allowed points (equivalently, using the smallest box in $\mathbb{R}^7$ that is aligned with the axes and contains all the allowed points).[23] This leads to the bounds reported in table 1. Let us explain this in more detail. First, note that our cloud of allowed points in $\mathbb{R}^7$ is long and thin: the eigenvalues of its covariance matrix have relative sizes

$$(1, 0.0493, 0.0144, 0.00261, 0.00165, 0.000555, 0.000126), \tag{5.3}$$

where we have divided by the largest eigenvalue. This suggests that the allowed region in $\mathbb{R}^7$ can be well-approximated by a low-dimensional shape, say a 2-dimensional shape. Furthermore, by the Delaunay search, the known allowed points approximately cover the projection of this 2-dimensional shape to $(\Delta_\sigma, \Delta_\epsilon)$-space. This suggests that the known allowed points in $\mathbb{R}^7$ give a good approximation of the allowed region. The relative error

---

[23]To make sure that our cloud of allowed points is representative, we sampled a number of points deep in the interior of the $\Lambda = 43$ allowed region in the dimension space, see figure 7.

can be estimated using the 3rd and higher eigenvalues of the covariance matrix, which is $\lesssim 0.3\%$ compared to the size of the island.

5.2.1 The parity-odd sector of the Ising CFT

Because $T \times T$ contains parity-odd tensor structures, the T, σ, ϵ mixed correlator system can be used to probe parity-odd scalars, giving us access to a new sector of operators. The $\mathbb{Z}_2$-even, parity-odd scalar channel has an unusually large gap. For intuition, note that the lightest analogous operator in the free scalar CFT is

$$\mathcal{O}_- = \; :\epsilon^{\mu\nu\rho}\phi(\partial_\alpha\partial_{\beta_1}\partial_{\beta_2}\partial_\mu\phi)(\partial^\alpha\partial_\nu\phi)(\partial^{\beta_1}\partial^{\beta_2}\partial_\rho\phi): \;, \tag{5.4}$$

which has scaling dimension $\Delta_- = 11$. The second-lightest operator in the free scalar theory, which contains an additional factor of ϕ^2, has scaling dimension $\Delta'_- = 12$.

We obtain an upper bound on Δ_- in the 3d Ising CFT by starting from the spectrum assumptions in table 6 and maximizing the $\mathbb{Z}_2$-even, parity-odd scalar gap via a binary search. At each value of the gap, we perform an OPE scan over the bounding ellipse depicted in 3d. Because the maximum gap depends on the operator dimensions $(\Delta_\sigma, \Delta_\epsilon)$, we repeat this process for a few points in the final $\Lambda = 43$ Ising island (figure 7; our search points are indicated on the inset of figure 9). This allows us to incorporate all available information about the dimensions while running the search at a much lower derivative order. At $\Lambda = 27$, we find an upper bound of

$$\Delta_- \leq 10.9293 \tag{5.5}$$

which very nearly coincides with the universal bound of $\Delta_- \lesssim 10.97$ (figure 5). This lends support to the conjecture that the 3d Ising model saturates this universal bound.

It's worth noting that all trial values of $(\Delta_\sigma, \Delta_\epsilon)$ produce identical upper bounds, at least within the binary search stopping tolerance of 10^{-4}. This is expected, as the $\Lambda = 43$ island covers roughly 1% of the area of the $\Lambda = 27$ one, so all points in it should look roughly the same from the perspective of the $\Lambda = 27$ crossing equations.

To compute a *lower* bound on Δ_-, we must impose a gap above this operator, which requires us to make some (physically motivated) assumptions about Δ'_-. It is reasonable to guess that Δ'_- is close to the free theory value of 12. A conservative gap of $\Delta'_- \geq 11$ implies a lower bound of $\Delta_- \geq 10.5432$, and a less conservative gap of $\Delta'_- \geq 11.5$ implies a lower bound of $\Delta_- \geq 10.8790$. An exclusion plot in (Δ_-, Δ'_-)-space is illustrated in figure 9. As in the gap maximization searches, all points in $(\Delta_\sigma, \Delta_\epsilon)$-space return similar lower bounds — the largest discrepancy between bounds from different points is 0.0002.

These lower bounds are considerably higher than fuzzy sphere estimate of $\Delta_- = 10.01(30)$ obtained in [21] (Note that the authors do not quantify the uncertainty of this operator's scaling dimension, but suggest that a 3% relative uncertainty is reasonable based on the uncertainties of other primaries' scaling dimensions). By fixing Δ_- to the fuzzy sphere value and scanning over various values of the Δ'_- gap, we find that it requires $\Delta'_- \leq 10.95$.

6 Conclusions

Our results give very precise determinations of leading scaling dimensions and OPE coefficients of the 3d Ising CFT, including data involving the stress tensor. We expect that one can
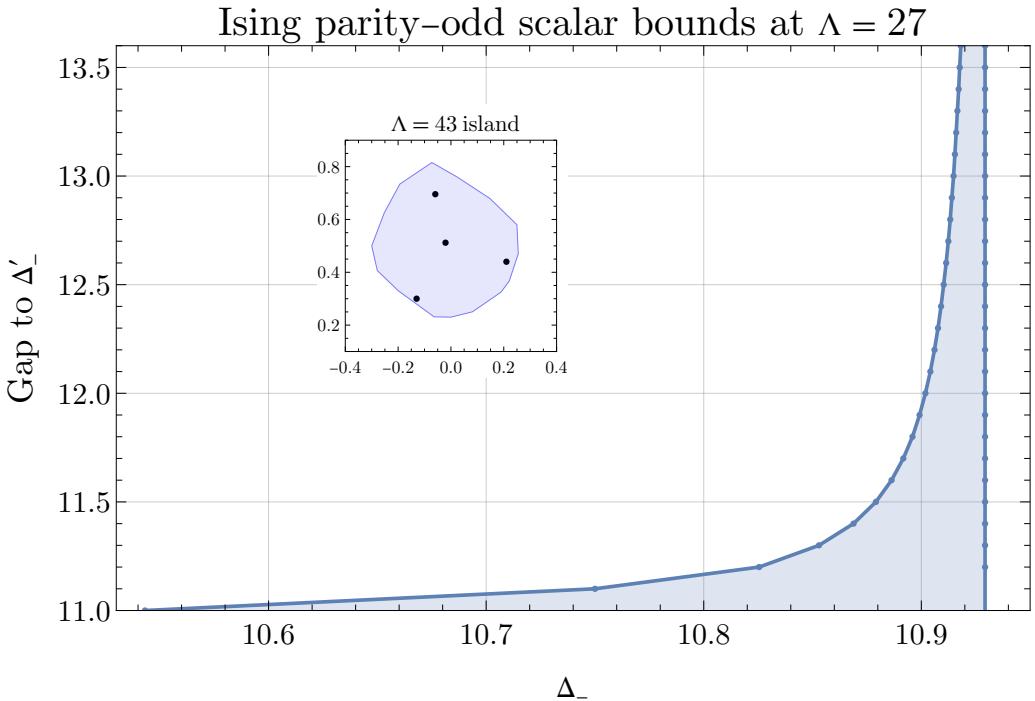
Figure 9. Bounds on scaling dimensions of the two lightest parity-even, $\mathbb{Z}_2$-odd scalars in the Ising model, computed at $\Lambda = 27$. The lower bound on Δ_- is sensitive to the value of Δ'_-, while the upper bound is not. The region shown here is the union of allowed regions for several representative points in the $\Lambda = 43$ $(\Delta_\sigma, \Delta_\epsilon)$ island, as highlighted on the plot inset.

learn about other operators in the spectrum by extracting and analyzing the extremal functionals [62, 63] corresponding to various boundaries of the 3d Ising island. This data should include trajectories of double-twist operators built out of the stress tensor which have so far not been studied in the numerical bootstrap. It will be very interesting to analyze these trajectories, comparing with predictions from the Lorentzian inversion formula [64], in the spirit of [12, 65–67]. This data will also give access to interesting observables associated with the ANEC operator in the 3d Ising CFT, such as energy correlators via the light-ray OPE [44, 68].

One amusing interpretation of our determinations of $\langle TTT \rangle$ OPE coefficients is that the 3d Ising CFT is "$n_F/(n_F + n_B) = 1.3833683(35)\%$ fermionic". We may wonder how this percentage compares against other interacting CFTs, such as the critical $O(N)$ vector models or CFTs built from interacting fermions or gauge fields. In particular, how these OPE coefficients may vary across different families of CFTs could well be nontrivial in light of conjectured bosonization/fermionization dualities [69]; see [70] for an excellent pedagogical review. Bootstrapping other mixed systems involving the stress tensor therefore would be an interesting line of inquiry. We leave these exciting directions to future work.

In section 5.1, we found that even the $\langle TTTT \rangle$ system alone can produce sharp features corresponding to the 3d Ising CFT. This observation leads to further questions: are there more such features in $\langle TTTT \rangle$ bounds, and can they be related to other physical CFTs? (Note that the stress tensor bootstrap automatically excludes non-local CFTs, such as the

long-range Ising model [71, 72].) Concretely, there is a large region of parameter space to the interior of the universal bounds in figure 4 in which any local, unitary CFT in 3d must exist. Is it possible to map out this interior (and the analogous space for global symmetry currents) by extremizing other parameters? Preliminary explorations reveal that there is a rich structure here and we plan to continue studying it in future work.

Given the effectiveness of including $T_{\mu\nu}$ for increasing the precision of bounds on the 3d Ising CFT, it is natural to ask: which operator is next? That is, which additional operator(s) should we include in the crossing equations to further isolate this theory? We do not have a rigorous answer to this question, but intuition from the lightcone bootstrap [12, 73, 74] suggests that low-twist operators could play an important role. Note that σ is the lowest-twist operator in the 3d Ising CFT, and $T_{\mu\nu}$ is the next-lowest. After that, we have the leading spin-4 operator $C_{\mu\nu\rho\sigma}$, with twist approximately $\tau_C \approx 1.022$. However, it is daunting (from the point of view of numerics) to imagine the system of crossing equations we would obtain by studying four-point functions of $C_{\mu\nu\rho\sigma}$ and other operators. As an example, the number of parity-even four-point structures in the $\langle CCCC \rangle$ four-point function alone is 881, which can be contrasted with the 29 "bulk" crossing equations studied in this work. It is interesting to ask whether we can include only the low-twist components of $C_{\mu\nu\rho\sigma}$ in an efficient way. An alternative approach is to consider higher-point crossing equations, which encode an infinite number of four-point crossing equations at once. See [27, 42, 75, 76] for recent work in this direction.

Acknowledgments

A Three-point structures

In this appendix, we record our choice of three-point structures for correlators of σ, ϵ, T in the 3d Ising CFT. We follow the conventions of [34], where we provide a three-point structure for each ordered triple of operators that can appear in a three-point correlator $\langle \mathcal{O}_i \mathcal{O}_j \mathcal{O}_k \rangle$. These structures must be chosen so that the OPE coefficients λ_{ijk}^a are functions of the *unordered* triple (i, j, k).[24] When all three operators are external (i.e. σ, ϵ, or T), we specify structures for all unique ordered triples. When one of the operators is internal, we choose conventions where the internal operator appears last, so we only specify structures for up to two unique orderings.

In the tables below, each section bounded by horizontal lines corresponds to an independent OPE coefficient λ_{ijk}. Within each section, each row corresponds to an ordered triple of operators and its corresponding three-point structure. The notation $|j_{12}, j_{123}\rangle$ denotes a conformal structure in the $SO(3)_r$ basis of [37, 38]. In addition, we also have specialized three-point structures such as `StressTensorB`, `ScalarParityEven`, etc., which are defined in the supplementary material `three_point_structures.m`.

In the tables below, we work in conventions where the two-point function of stress tensors is unit-normalized. Thus, T in this section is the same as $\widehat{T}$ in section 2.2.

External operators. There are five OPE coefficients corresponding to external operators: $(\lambda_B, \lambda_F, \lambda_{\sigma\sigma\epsilon}, \lambda_{\epsilon\epsilon\epsilon}, \lambda_{TT\epsilon})$. Due to the Ward identity (2.13), the OPE coefficients λ_B and λ_F

---

[24]In practice this means that the structures are related by permutations. By providing the functions explicitly, we allow the code that prepares conformal block expansions to be independent of the details of how permutations act on local operators.

appear in three-point structures for triples of multiple different operators.

OPE coefficient	ordered triple $\langle \mathcal{O}_i \mathcal{O}_j \mathcal{O}_j \rangle$	three-point structure
λ_B	$\langle TTT \rangle$	StressTensorB
	$\langle T\sigma\sigma \rangle$	$-\Delta_\sigma |2,2\rangle$
	$\langle \sigma T\sigma \rangle$	$-\Delta_\sigma |2,2\rangle$
	$\langle \sigma\sigma T \rangle$	$-\Delta_\sigma |0,2\rangle$
	$\langle T\epsilon\epsilon \rangle$	$-\Delta_\epsilon |2,2\rangle$
	$\langle \epsilon T\epsilon \rangle$	$-\Delta_\epsilon |2,2\rangle$
	$\langle \epsilon\epsilon T \rangle$	$-\Delta_\epsilon |0,2\rangle$
λ_F	$\langle TTT \rangle$	StressTensorF
	$\langle T\sigma\sigma \rangle$	$-\Delta_\sigma |2,2\rangle$
	$\langle \sigma T\sigma \rangle$	$-\Delta_\sigma |2,2\rangle$
	$\langle \sigma\sigma T \rangle$	$-\Delta_\sigma |0,2\rangle$
	$\langle T\epsilon\epsilon \rangle$	$-\Delta_\epsilon |2,2\rangle$
	$\langle \epsilon T\epsilon \rangle$	$-\Delta_\epsilon |2,2\rangle$
	$\langle \epsilon\epsilon T \rangle$	$-\Delta_\epsilon |0,2\rangle$
$\lambda_{\sigma\sigma\epsilon}$	$\langle \sigma\sigma\epsilon \rangle$	$|0,0\rangle$
	$\langle \sigma\epsilon\sigma \rangle$	$|0,0\rangle$
	$\langle \epsilon\sigma\sigma \rangle$	$|0,0\rangle$
$\lambda_{\epsilon\epsilon\epsilon}$	$\langle \epsilon\epsilon\epsilon \rangle$	$|0,0\rangle$
$\lambda_{TT\epsilon}$	$\langle TT\epsilon \rangle$	$-\frac{16\Delta_\epsilon}{2+\Delta_\epsilon}$ ScalarParityEven
	$\langle T\epsilon T \rangle$	$\frac{6}{\Delta_\epsilon(2+\Delta_\epsilon)}$ TSOParityEven$_2$
	$\langle \epsilon TT \rangle$	$\frac{6}{\Delta_\epsilon(2+\Delta_\epsilon)}$ STOParityEven$_2$

(A.1)

In subsequent tables, we omit the OPE coefficient names (i.e. the left-most column in the table above).

$\mathbb{Z}_2$-even, parity-even, $j = 0$

$\langle TT\mathcal{O} \rangle$	ScalarParityEven
$\langle \sigma\sigma\mathcal{O} \rangle$	$|0,0\rangle$
$\langle \epsilon\epsilon\mathcal{O} \rangle$	$|0,0\rangle$

(A.2)

$\mathbb{Z}_2$-even, parity-even, $j = 2$

$\langle TT\mathcal{O} \rangle$	Spin2ParityEven
$\langle \sigma\sigma\mathcal{O} \rangle$	$|0,2\rangle$
$\langle \epsilon\epsilon\mathcal{O} \rangle$	$|0,2\rangle$
$\langle T\epsilon\mathcal{O} \rangle$	TSOParityEven$_2$
$\langle \epsilon T\mathcal{O} \rangle$	STOParityEven$_2$

(A.3)

$\mathbb{Z}_2$-even, parity-even, $j \geq 3$ odd

$$
\begin{array}{|c|c|}
\hline
\langle T\epsilon\mathcal{O}\rangle & \texttt{TSOParityEven}_j \\
\hline
\langle \epsilon T\mathcal{O}\rangle & \texttt{STOParityEven}_j \\
\hline
\end{array}
\tag{A.4}
$$

$\mathbb{Z}_2$-even, parity-even, $j \geq 4$ even

$$
\begin{array}{|c|c|}
\hline
\langle TT\mathcal{O}\rangle & \texttt{GenericParityEven1}_j \\
\hline
\langle TT\mathcal{O}\rangle & \texttt{GenericParityEven2}_j \\
\hline
\langle \sigma\sigma\mathcal{O}\rangle & |0, j\rangle \\
\hline
\langle \epsilon\epsilon\mathcal{O}\rangle & |0, j\rangle \\
\hline
\langle T\epsilon\mathcal{O}\rangle & \texttt{TSOParityEven}_j \\
\hline
\langle \epsilon T\mathcal{O}\rangle & \texttt{STOParityEven}_j \\
\hline
\end{array}
\tag{A.5}
$$

$\mathbb{Z}_2$-even, parity-odd, $j = 0$

$$
\begin{array}{|c|c|}
\hline
\langle TT\mathcal{O}\rangle & \texttt{ScalarParityOdd} \\
\hline
\end{array}
\tag{A.6}
$$

$\mathbb{Z}_2$-even, parity-odd, $j = 2$

$$
\begin{array}{|c|c|}
\hline
\langle TT\mathcal{O}\rangle & \texttt{Spin2ParityOdd} \\
\hline
\langle T\epsilon\mathcal{O}\rangle & \texttt{TSOParityOdd}_2 \\
\hline
\langle \epsilon T\mathcal{O}\rangle & \texttt{STOParityOdd}_2 \\
\hline
\end{array}
\tag{A.7}
$$

$\mathbb{Z}_2$-even, parity-odd, $j = 3$

$$
\begin{array}{|c|c|}
\hline
\langle T\epsilon\mathcal{O}\rangle & \texttt{TSOParityOdd}_3 \\
\hline
\langle \epsilon T\mathcal{O}\rangle & \texttt{STOParityOdd}_3 \\
\hline
\end{array}
\tag{A.8}
$$

$\mathbb{Z}_2$-even, parity-odd, $j \geq 4$

$$
\begin{array}{|c|c|}
\hline
\langle TT\mathcal{O}\rangle & \texttt{GenericParityOdd}_j \\
\hline
\langle T\epsilon\mathcal{O}\rangle & \texttt{TSOParityOdd}_j \\
\hline
\langle \epsilon T\mathcal{O}\rangle & \texttt{STOParityOdd}_j \\
\hline
\end{array}
\tag{A.9}
$$

$\mathbb{Z}_2$-odd, parity-even, $j = 0$

$$
\begin{array}{|c|c|}
\hline
\langle \sigma\epsilon\mathcal{O}\rangle & |0, 0\rangle \\
\hline
\langle \epsilon\sigma\mathcal{O}\rangle & |0, 0\rangle \\
\hline
\end{array}
\tag{A.10}
$$

$\mathbb{Z}_2$-odd, parity-even, $j = 1$

$$
\begin{array}{|c|c|}
\hline
\langle \sigma\epsilon\mathcal{O}\rangle & |0, 1\rangle \\
\hline
\langle \epsilon\sigma\mathcal{O}\rangle & -|0, 1\rangle \\
\hline
\end{array}
\tag{A.11}
$$

$\mathbb{Z}_2$-odd, parity-even, $j \geq 2$

$$
\begin{array}{c|l}
\langle T\sigma\mathcal{O}\rangle & \texttt{TSOParityEven}_j \\
\hline
\langle \sigma T\mathcal{O}\rangle & \texttt{STOParityEven}_j \\
\hline
\langle \sigma\epsilon\mathcal{O}\rangle & |0,j\rangle \\
\hline
\langle \epsilon\sigma\mathcal{O}\rangle & (-1)^j|0,j\rangle
\end{array}
\tag{A.12}
$$

$\mathbb{Z}_2$-odd, parity-odd, $j \geq 2$

$$
\begin{array}{c|l}
\langle T\sigma\mathcal{O}\rangle & \texttt{TSOParityOdd}_j \\
\hline
\langle \sigma T\mathcal{O}\rangle & \texttt{STOParityOdd}_j
\end{array}
\tag{A.13}
$$

B Crossing equations for the $T\sigma\epsilon$ system

In this appendix, we present the crossing equations for correlators of T, σ, and ϵ. We use the cross-ratios w, s defined by [38, 78]

$$
z = \frac{(1+y)^2}{2(1+y^2)}, \qquad \overline{z} = \frac{(1+\overline{y})^2}{2(1+\overline{y}^2)},
$$

$$
w = \frac{y+\overline{y}}{2}, \qquad s = \left(\frac{y-\overline{y}}{2}\right)^2.
\tag{B.1}
$$

In these coordinates, the crossing symmetric point is $w = s = 0$. We denote evaluation of a function $g(w,s)$ at the crossing symmetric point by $g|_* \equiv g(w=0, s=0)$. The careful reader may worry that with this definition of s we omit structures that are odd under $z \leftrightarrow \overline{z}$; as noted in section 2.3, only the even structures appear. For more general cases where this may not be true, refer to [34].

The number of derivatives $\partial_w^m \partial_s^n$ of each crossing equation is determined by an integer parameter Λ. It is also useful to define $n_{\max} = \left\lfloor \frac{\Lambda+1}{2} \right\rfloor$. We use the following terminology:

- "bulk equations" are equations that involve derivatives in both cross-ratios w and s. In our conventions, we symmetrize/anti-symmetrize each equation with respect to the crossing transformation $w \to -w$. Thus, each bulk equation involves either even or odd derivatives with respect to w. Consequently, each bulk equation has $\frac{n_{\max}(n_{\max}+1)}{2}$ components.

- "line equations" (or "radial equations") are equations that involve derivatives only along the line $s = 0$. Each line equation has $n_{\max}$ components.

- "point equations" involve no derivatives, and are given by evaluating an expression at the crossing-symmetric point. Each point equation has 1 component.

Overall, the $T\sigma\epsilon$ system has 30 bulk equations, 18 line equations, and 5 point equations, for a total of

$$
\dim \vec{\alpha} = 30\frac{n_{\max}(n_{\max}+1)}{2} + 18n_{\max} + 5
\tag{B.2}
$$

functional components, see table 7. The crossing equations are listed below. In each equation $m, n \geq 0$ are nonnegative integers subject to the constraint $m + 2n \leq \Lambda$.

Λ	$n_{\max}$	components
11	6	743
19	10	1835
27	14	3407
35	18	5459
43	22	7991

Table 7. Number of components of the functional acting on crossing equations for the $T\sigma\epsilon$ system, for different derivative orders.

Bulk equations for $\langle TTTT \rangle$

$$0 = \partial_w^m \partial_s^n \left(g_{TTTT}^{\langle 2,2,2,2 \rangle^+} \right)\Big|_*, \qquad\qquad (m \text{ odd})$$

$$0 = \partial_w^m \partial_s^n \left(g_{TTTT}^{\langle 1,1,1,1 \rangle^+} \right)\Big|_*, \qquad\qquad (m \text{ odd})$$

$$0 = \partial_w^m \partial_s^n \left(g_{TTTT}^{\langle 1,1,2,2 \rangle^+} + g_{TTTT}^{\langle 2,1,1,2 \rangle^+} \right)\Big|_*, \qquad\qquad (m \text{ odd})$$

$$0 = \partial_w^m \partial_s^n \left(g_{TTTT}^{\langle 1,1,2,2 \rangle^+} - g_{TTTT}^{\langle 2,1,1,2 \rangle^+} \right)\Big|_*, \qquad\qquad (m \text{ even})$$

$$0 = \partial_w^m \partial_s^n \left(g_{TTTT}^{\langle 1,2,1,2 \rangle^+} \right)\Big|_*. \qquad\qquad (m \text{ odd}) \qquad (\text{B.3})$$

Line equations for $\langle TTTT \rangle$

$$0 = \partial_w^m \left(g_{TTTT}^{\langle 0,0,0,0 \rangle^+} \right)\Big|_*, \qquad\qquad (m \text{ odd})$$

$$0 = \partial_w^m \left(g_{TTTT}^{\langle 0,1,0,1 \rangle^+} \right)\Big|_*, \qquad\qquad (m \text{ odd})$$

$$0 = \partial_w^m \left(g_{TTTT}^{\langle 0,2,0,2 \rangle^+} \right)\Big|_*, \qquad\qquad (m \text{ odd})$$

$$0 = \partial_w^m \left(g_{TTTT}^{\langle 0,1,1,2 \rangle^+} + g_{TTTT}^{\langle 1,1,0,2 \rangle^+} \right)\Big|_*, \qquad\qquad (m \text{ odd})$$

$$0 = \partial_w^m \left(g_{TTTT}^{\langle 0,1,1,2 \rangle^+} - g_{TTTT}^{\langle 1,1,0,2 \rangle^+} \right)\Big|_*, \qquad\qquad (m \text{ even})$$

$$0 = \partial_w^m \left(g_{TTTT}^{\langle 0,0,1,1 \rangle^+} + g_{TTTT}^{\langle 1,0,0,1 \rangle^+} \right)\Big|_*, \qquad\qquad (m \text{ odd})$$

$$0 = \partial_w^m \left(g_{TTTT}^{\langle 0,0,1,1 \rangle^+} - g_{TTTT}^{\langle 1,0,0,1 \rangle^+} \right)\Big|_*, \qquad\qquad (m \text{ even})$$

$$0 = \partial_w^m \left(g_{TTTT}^{\langle -1,0,0,1 \rangle^+} + g_{TTTT}^{\langle 0,0,-1,1 \rangle^+} \right)\Big|_*, \qquad\qquad (m \text{ odd})$$

$$0 = \partial_w^m \left(g_{TTTT}^{\langle -1,0,0,1 \rangle^+} - g_{TTTT}^{\langle 0,0,-1,1 \rangle^+} \right)\Big|_*. \qquad\qquad (m \text{ even}) \qquad (\text{B.4})$$

Point equations for $\langle TTTT \rangle$

$$0 = \left(g_{TTTT}^{\langle 0,0,2,2 \rangle^+} - g_{TTTT}^{\langle 2,0,0,2 \rangle^+} \right)\Big|_*,$$

$$0 = \left(g_{TTTT}^{\langle 0,-1,1,2 \rangle^+} - g_{TTTT}^{\langle 1,-1,0,2 \rangle^+} \right)\Big|_*,$$

$$0 = \left(g_{TTTT}^{\langle -1,-1,1,1 \rangle^+} - g_{TTTT}^{\langle 1,-1,-1,1 \rangle^+} \right)\Big|_*,$$

$$0 = \left(g_{TTTT}^{\langle -1,1,0,2 \rangle^+} - g_{TTTT}^{\langle 0,1,-1,2 \rangle^+} \right)\Big|_*. \qquad (\text{B.5})$$

Bulk equations for $\langle TTT\epsilon \rangle$

$$0 = \partial_w^m \partial_s^n \left(g_{TTT\epsilon}^{\langle 2,2,2,0 \rangle^+} \right)\Big|_* , \qquad (m \text{ odd})$$

$$0 = \partial_w^m \partial_s^n \left(g_{TTT\epsilon}^{\langle 1,2,1,0 \rangle^+} \right)\Big|_* , \qquad (m \text{ odd})$$

$$0 = \partial_w^m \partial_s^n \left(g_{TTT\epsilon}^{\langle 1,1,2,0 \rangle^+} + g_{TTT\epsilon}^{\langle 2,1,1,0 \rangle^+} \right)\Big|_* , \qquad (m \text{ odd})$$

$$0 = \partial_w^m \partial_s^n \left(g_{TTT\epsilon}^{\langle 1,1,2,0 \rangle^+} - g_{TTT\epsilon}^{\langle 2,1,1,0 \rangle^+} \right)\Big|_* . \qquad (m \text{ even}) \qquad (\text{B.6})$$

Line equations for $\langle TTT\epsilon \rangle$

$$0 = \partial_w^m \left(g_{TTT\epsilon}^{\langle 0,0,2,0 \rangle^+} + g_{TTT\epsilon}^{\langle 2,0,0,0 \rangle^+} \right)\Big|_* , \qquad (m \text{ odd})$$

$$0 = \partial_w^m \left(g_{TTT\epsilon}^{\langle 0,0,2,0 \rangle^+} - g_{TTT\epsilon}^{\langle 2,0,0,0 \rangle^+} \right)\Big|_* , \qquad (m \text{ even})$$

$$0 = \partial_w^m \left(g_{TTT\epsilon}^{\langle 0,2,0,0 \rangle^+} \right)\Big|_* . \qquad (m \text{ odd}) \qquad (\text{B.7})$$

Point equations for $\langle TTT\epsilon \rangle$

$$0 = \left(g_{TTT\epsilon}^{\langle -2,2,2,0 \rangle^+} - g_{TTT\epsilon}^{\langle 2,2,-2,0 \rangle^+} \right)\Big|_* . \qquad (\text{B.8})$$

Bulk equations for $\langle TT\sigma\sigma \rangle$

$$0 = \partial_w^m \partial_s^n \left(g_{TT\sigma\sigma}^{\langle 2,2,0,0 \rangle^+} + g_{\sigma TT\sigma}^{\langle 0,2,2,0 \rangle^+} \right)\Big|_* , \qquad (m \text{ odd})$$

$$0 = \partial_w^m \partial_s^n \left(g_{TT\sigma\sigma}^{\langle 2,2,0,0 \rangle^+} - g_{\sigma TT\sigma}^{\langle 0,2,2,0 \rangle^+} \right)\Big|_* , \qquad (m \text{ even})$$

$$0 = \partial_w^m \partial_s^n \left(g_{TT\sigma\sigma}^{\langle 1,1,0,0 \rangle^+} + g_{\sigma TT\sigma}^{\langle 0,1,1,0 \rangle^+} \right)\Big|_* , \qquad (m \text{ odd})$$

$$0 = \partial_w^m \partial_s^n \left(g_{TT\sigma\sigma}^{\langle 1,1,0,0 \rangle^+} - g_{\sigma TT\sigma}^{\langle 0,1,1,0 \rangle^+} \right)\Big|_* , \qquad (m \text{ even})$$

$$0 = \partial_w^m \partial_s^n \left(g_{\sigma T\sigma T}^{\langle 0,2,0,2 \rangle^+} \right)\Big|_* , \qquad (m \text{ odd})$$

$$0 = \partial_w^m \partial_s^n \left(g_{\sigma T\sigma T}^{\langle 0,1,0,1 \rangle^+} \right)\Big|_* . \qquad (m \text{ odd}) \qquad (\text{B.9})$$

Line equations for $\langle TT\sigma\sigma \rangle$

$$0 = \partial_w^m \left(g_{TT\sigma\sigma}^{\langle 0,0,0,0 \rangle^+} + g_{\sigma TT\sigma}^{\langle 0,0,0,0 \rangle^+} \right)\Big|_* , \qquad (m \text{ odd})$$

$$0 = \partial_w^m \left(g_{TT\sigma\sigma}^{\langle 0,0,0,0 \rangle^+} - g_{\sigma TT\sigma}^{\langle 0,0,0,0 \rangle^+} \right)\Big|_* , \qquad (m \text{ even})$$

$$0 = \partial_w^m \left(g_{\sigma T\sigma T}^{\langle 0,0,0,0 \rangle^+} \right)\Big|_* . \qquad (m \text{ odd}) \qquad (\text{B.10})$$

Bulk equations for $\langle TT\epsilon\epsilon \rangle$

$$0 = \partial_w^m \partial_s^n \left(g_{TT\epsilon\epsilon}^{\langle 2,2,0,0 \rangle^+} + g_{\epsilon TT\epsilon}^{\langle 0,2,2,0 \rangle^+} \right)\Big|_* , \qquad (m \text{ odd})$$

$$0 = \partial_w^m \partial_s^n \left(g_{TT\epsilon\epsilon}^{\langle 2,2,0,0 \rangle^+} - g_{\epsilon TT\epsilon}^{\langle 0,2,2,0 \rangle^+} \right)\Big|_* , \qquad (m \text{ even})$$

$$0 = \partial_w^m \partial_s^n \left(g_{TT\epsilon\epsilon}^{\langle 1,1,0,0 \rangle^+} + g_{\epsilon TT\epsilon}^{\langle 0,1,1,0 \rangle^+} \right)\Big|_* , \qquad (m \text{ odd})$$

$$0 = \partial_w^m \partial_s^n \left(g_{TT\epsilon\epsilon}^{\langle 1,1,0,0 \rangle^+} - g_{\epsilon TT\epsilon}^{\langle 0,1,1,0 \rangle^+} \right)\Big|_* , \qquad (m \text{ even})$$

$$0 = \partial_w^m \partial_s^n \left(g_{\epsilon T\epsilon T}^{\langle 0,2,0,2 \rangle^+} \right)\Big|_* , \qquad (m \text{ odd})$$

$$0 = \partial_w^m \partial_s^n \left(g_{\epsilon T\epsilon T}^{\langle 0,1,0,1 \rangle^+} \right)\Big|_* . \qquad (m \text{ odd}) \qquad (\text{B.11})$$

Line equations for $\langle TT\epsilon\epsilon \rangle$

$$0 = \partial_w^m \left(g_{TT\epsilon\epsilon}^{\langle 0,0,0,0 \rangle^+} + g_{\epsilon TT\epsilon}^{\langle 0,0,0,0 \rangle^+} \right)\Big|_*, \qquad (m \text{ odd})$$

$$0 = \partial_w^m \left(g_{TT\epsilon\epsilon}^{\langle 0,0,0,0 \rangle^+} - g_{\epsilon TT\epsilon}^{\langle 0,0,0,0 \rangle^+} \right)\Big|_*, \qquad (m \text{ even})$$

$$0 = \partial_w^m \left(g_{\epsilon T\epsilon T}^{\langle 0,0,0,0 \rangle^+} \right)\Big|_*. \qquad (m \text{ odd}) \qquad (\text{B.12})$$

Bulk equations for $\langle T\epsilon\epsilon\epsilon \rangle$

$$0 = \partial_w^m \partial_s^n \left(g_{\epsilon T\epsilon\epsilon}^{\langle 0,2,0,0 \rangle^+} \right)\Big|_*. \qquad (m \text{ odd}) \qquad (\text{B.13})$$

Bulk equations for $\langle \sigma\sigma\epsilon T \rangle$

$$0 = \partial_w^m \partial_s^n \left(g_{\sigma T\epsilon\sigma}^{\langle 0,2,0,0 \rangle^+} + g_{\epsilon T\sigma\sigma}^{\langle 0,2,0,0 \rangle^+} \right)\Big|_*, \qquad (m \text{ odd})$$

$$0 = \partial_w^m \partial_s^n \left(g_{\sigma T\epsilon\sigma}^{\langle 0,2,0,0 \rangle^+} - g_{\epsilon T\sigma\sigma}^{\langle 0,2,0,0 \rangle^+} \right)\Big|_*, \qquad (m \text{ even})$$

$$0 = \partial_w^m \partial_s^n \left(g_{\sigma T\sigma\epsilon}^{\langle 0,2,0,0 \rangle^+} \right)\Big|_*. \qquad (m \text{ odd}) \qquad (\text{B.14})$$

Bulk equations for the $\sigma\epsilon$ system [10]

$$0 = \partial_w^m \partial_s^n \left(g_{\sigma\sigma\sigma\sigma}^{\langle 0,0,0,0 \rangle^+} \right)\Big|_*, \qquad (m \text{ odd})$$

$$0 = \partial_w^m \partial_s^n \left(g_{\epsilon\epsilon\epsilon\epsilon}^{\langle 0,0,0,0 \rangle^+} \right)\Big|_*, \qquad (m \text{ odd})$$

$$0 = \partial_w^m \partial_s^n \left(g_{\sigma\sigma\epsilon\epsilon}^{\langle 0,0,0,0 \rangle^+} + g_{\epsilon\sigma\sigma\epsilon}^{\langle 0,0,0,0 \rangle^+} \right)\Big|_*, \qquad (m \text{ odd})$$

$$0 = \partial_w^m \partial_s^n \left(g_{\sigma\sigma\epsilon\epsilon}^{\langle 0,0,0,0 \rangle^+} - g_{\epsilon\sigma\sigma\epsilon}^{\langle 0,0,0,0 \rangle^+} \right)\Big|_*, \qquad (m \text{ even})$$

$$0 = \partial_w^m \partial_s^n \left(g_{\sigma\epsilon\sigma\epsilon}^{\langle 0,0,0,0 \rangle^+} \right)\Big|_*. \qquad (m \text{ odd}) \qquad (\text{B.15})$$

C   Computing dependencies

In the build process described in section 3.1, we need to compute dependencies of each element of an SDP so as to minimize redundant work. For example, we must be able to determine the `Block3d`'s that a given `CompositeBlock` depends on, or the `Block3d`'s and `CompositeBlock`'s that a given `PartChunk` depends on. One possible approach is to write and maintain separate routines for constructing an element and for determining its dependencies. However, a downside of this approach is that the two routines must be manually kept in sync with each other, and this is error-prone. It is common to make changes to many parts of a computation when debugging, and it is furthermore common to spin off smaller example computations, such as an SDP for T-σ correlators alone. We prefer not to have to worry about keeping track of dependencies during such changes.

Instead we use a technique from [79] (nicely reviewed in [80]) that leverages Haskell's polymorphism to write a single function that simultaneously describes how to build an element and determine its dependencies. We first started using this method in [34], but to our knowledge it has not yet been described in the bootstrap literature.

A toy example of the technique is shown in listing 1. In this example, the routine `myComputation` takes a function `fetch` that returns an `Int` for each `Key` (here `Key` is `String`)

```
{-# LANGUAGE ApplicativeDo #-}

type Key = String

myComputation :: Applicative f => (Key -> f Int) -> f Int
myComputation fetch = do
x1 <- fetch "Hello"
x2 <- fetch "Dolly"
x3 <- fetch "Yes"
pure (x1 + 2*x2 + 3*x3)

-- > import Data.Functor.Identity
-- > import Data.Functor.Const
-- > runIdentity $ myComputation (Identity . length)
-- 24
-- > getConst $ myComputation (\k -> Const [k])
-- ["Hello", "Dolly", "Yes"]
```

Listing 1. Example of using polymorphism in an `Applicative` to simultaneously encode a computation and its dependencies. The comments show an example interactive session in `ghci` using this code.

in some `Applicative` context `f`. It calls `fetch` on a few `Key`s, and does a computation on the results. When we run `myComputation` with `f` instantiated to `Identity` and `fetch` the length function, it performs the computation on the lengths of the `Key`s "inside" `myComputation`. However, at the same time, we can access the `Key`s themselves by instantiating `f` to `Const [Key]`. In this case, no computation is performed, and only the `Key`s are collected. Thus, because `myComputation` is polymorphic in `f`, it elegantly describes both a computation and the `Key`s that the computation depends on.

In a more realistic example, we could have a function `fetchCompositeBlock` with type `Applicative f => (BlockTableKey -> f Block3d) -> f CompositeBlock`. When the `f` is instantiated to `IO`, `fetchCompositeBlock` can fetch `Block3d`s from files and assemble them into appropriate linear combinations. By contrast, when `f` is instantiated to `Const [BlockTableKey]`, then `fetchCompositeBlock` simply lists the `BlockTableKey`s that are needed without doing any fetching or computation. This technique also composes nicely. We can use `fetchCompositeBlock` as a subroutine in a more complicated function `fetchSDP` with type `Applicative f => (BlockTableKey -> f Block3d) -> f SDP`. This more complicated function is then capable of both building an `SDP` and listing all the `BlockTableKey`s that go into it, depending on how `f` is instantiated.

In this way, we get the best of both worlds: we can pre-compute and memoize our dependencies, but the explicit construction routine is robust and nearly invisible to the end-user once the general `fetch` routines are in place.

D   Scheduler for building SDPs

In this appendix, we describe our ad-hoc scheduler for building SDPs. Recall from section 3.1 that each SDP depends on several elements: some number of `Block3d`'s, `CompositeBlock`'s,

`PartChunk`'s, and `Part`'s. From these elements, we build a tree of tasks, where an edge indicates a dependency: a child task must be completed before the parent task can be started. We label each task with a memory estimate, a runtime estimate function (which maps a number of cores n to a time), and a minimum and maximum number of threads. (For example, our `CompositeBlock, PartChunk, and Part` tasks are not parallelized, so they each have `maxThreads = minThreads = 1`.) We assume that we have access to K nodes, each with C CPUs, and a fixed amount of memory M. Our algorithm has two phases:

D.1 Phase 1: initial task distribution

To begin, we collect the tasks that have no dependencies that need to be built (which are usually `Block3d` tasks), and sort them in reverse order of memory usage. We find a distribution of these initial tasks to the K nodes by traversing the list in order. For a task t, we sort the nodes by how quickly they are expected to complete their current task assignments, and filter out nodes for which adding t would decrease their "score." We assign t to the earliest-completing node whose score will be increased, and such that we will not exceed the memory of the node. When we encounter a t such that there doesn't exist a node satisfying these requirements (for example, because none of the nodes have enough extra memory to accommodate t), the initial task distribution phase is over, and we proceed to the next phase.

The above algorithm depends on two definitions: a *completion time* and a *score* for a node with a given task assignment. We estimate *completion time* by finding a locally optimal allocation of CPUs to the assigned tasks that minimizes the maximum completion time of all assigned tasks (based on the runtime estimate function). We find this allocation in a simple way: we start with an even distribution of CPUs, and then steal CPUs from the fastest task and give them to the slowest task (while taking care not to violate the `minThreads` and `maxThreads` requirements), as long as overall performance increases. The *score* is a heuristic measure that we give to an allocation of CPUs to tasks, based on the idea that the completion times of the individual tasks should be similar.

After obtaining the initial task distribution, we start the corresponding tasks on their assigned nodes.

D.2 Phase 2: continually running tasks

Once tasks are running, we switch to phase 2. We maintain a priority queue of tasks that are ready to run. In practice, we choose the priority to be a tuple (`depth, memory`), where `depth` is the depth of the task in the tree (deeper tasks have higher priority), and `memory` is the task's memory estimate.

When a task finishes, we compute its reverse dependencies. If any of them are ready to be started (because all their child tasks have been completed), we add them to the priority queue. Each node keeps track of its number of free CPUs and amount of free memory and monitors the queue. If a node finds a task at the front of the queue that it can perform, given its free resources, it dequeue's that task. If it has enough resources, a node can dequeue multiple tasks at once, in which case it allocates free CPUs to them using the same algorithm as in subsection D.1. It then begins running the newly dequeued tasks.

By running this algorithm, eventually all tasks will be completed.

D.3 Collecting resource usage data

The algorithm described above depends on a memory estimate and a runtime estimate for each task. When possible, we tried to build an analytical model predicting these resource requirements. This is painstaking because it requires intimate knowledge of how the task is performed, together with good measurements. These analytical models are needed when we perform a fresh build with new values of the parameters ($\ell_{\max}$, `order`, etc. discussed in appendix E) that we have not tried before.

However, during every build, we record the actual resource usage of each task. Once we have this data, we can use it for memory and runtime estimates in subsequent builds. (This is possible because changing parameters like Δ_σ and Δ_ϵ does not affect resource requirements.) Thus, after collecting resource usage data, we have accurate information about resource requirements, and we are no longer reliant on ad-hoc analytical models.

E Parameters

We ran a number of tests changing different parameters in our computation to find the optimal parameters that yield accurate results without wasting too much CPU time. The parameters are $\ell_{\max}$, giving the maximum spin included in our constraints, `order`, giving the order of the radial expansion used in generating the conformal blocks, `poles`, giving the effective number of poles used in the block interpolation described in 3.4, and `prec`, giving the internal binary precision used in our computations.

The results for $\Lambda = 35$ and $\Lambda = 43$ are summarized in tables 8 and 9, respectively. For a given set of spins $0 \ldots \ell_{\max}$, `order`, `poles`, and `prec`, we computed the primal objective defined as the value of the GFF navigator function.

Then, we computed the accuracy δ as the absolute difference of the primal objective and the most accurate primal objective (i.e. computed with the largest $\ell_{\max}$ and reasonable values of other parameters, see the last rows in tables 8 and 9). In a similar way, we computed the accuracy at a given $\ell_{\max}$, denoted as $\delta^{(\ell_{\max})}$, and the accuracy at a given $\ell_{\max}$ and `order`, denoted as $\delta^{(\ell_{\max}, \text{order})}$.

One can make the following observations from the accuracy tests:

- Objective converges slowly with increasing $\ell_{\max}$. This means that, after a certain point, increasing `order` or number of poles for fixed $\ell_{\max}$ does not improve accuracy (see the column δ in tables 8 and 9).

- There is a sharp cutoff at low `order`s: for $\Lambda = 35$, SDPB fails to converge at `order` $= 95$, converges to a completely wrong result for `order` $= 96$ and `order` $= 97$, and behaves well starting from `order` $= 98$ (see figure 10 and the column $\delta^{(\ell_{\max})}$ in table 8). $\Lambda = 43$ exhibits similar behavior and converges to a correct result starting from `order` $= 120$ (see the column $\delta^{(\ell_{\max})}$ in table 9).

- There is a cutoff for precision as well. For example, for $\Lambda = 35$, `prec` $= 576$ leads to SDPB crashing due to numerical errors. Any precision starting from 640 yields the same result.

$\ell_{\max}$	order	poles	prec	primal objective	δ	$\delta^{(\ell_{\max})}$	$\delta^{(\ell_{\max},\texttt{order})}$
40	100	30	1088	— (SDPB failed)	—	—	—
50	100	30	1088	$4.034436016 \times 10^{-8}$	$2. \times 10^{-10}$	0	0
60	100	30	1088	$4.0256247689 \times 10^{-8}$	1.1×10^{-10}	0	0
70	100	30	576	— (SDPB failed)	—	—	—
70	100	30	640	$4.0210005644 \times 10^{-8}$	6.3×10^{-11}	0	0
70	100	30	768	$4.0210005644 \times 10^{-8}$	6.3×10^{-11}	0	0
70	100	30	1088	$4.0210005644 \times 10^{-8}$	6.3×10^{-11}	0	0
80	100	30	1088	$4.018319405 \times 10^{-8}$	3.6×10^{-11}	0	0
88	95	80	1088	— (SDPB failed)	—	—	—
88	96	80	1088	0.050659646253	0.051	0.051	0
88	97	80	1088	0.37298630331	0.37	0.37	0
88	98	80	1088	$4.0176280315 \times 10^{-8}$	2.9×10^{-11}	$6. \times 10^{-12}$	0
88	99	80	1088	$4.0253171472 \times 10^{-8}$	1.1×10^{-10}	8.3×10^{-11}	0
88	100	20	1088	$3.9985768333 \times 10^{-8}$	1.6×10^{-10}	1.8×10^{-10}	1.8×10^{-10}
88	100	25	1088	$4.0169619513 \times 10^{-8}$	2.2×10^{-11}	6.5×10^{-13}	5.2×10^{-23}
88	100	30	1088	$4.0169619513 \times 10^{-8}$	2.2×10^{-11}	6.5×10^{-13}	1.8×10^{-24}
88	100	40	1088	$4.0169619513 \times 10^{-8}$	2.2×10^{-11}	6.5×10^{-13}	3.1×10^{-28}
88	100	60	1088	$4.0169619513 \times 10^{-8}$	2.2×10^{-11}	6.5×10^{-13}	0
88	100	80	1088	$4.0169619513 \times 10^{-8}$	2.2×10^{-11}	6.5×10^{-13}	0
88	101	80	1088	$4.0168942283 \times 10^{-8}$	2.2×10^{-11}	1.3×10^{-12}	0
88	110	80	1088	$4.0170267048 \times 10^{-8}$	2.3×10^{-11}	1.2×10^{-18}	0
88	120	80	1088	$4.0170267049 \times 10^{-8}$	2.3×10^{-11}	0	0
100	100	30	1088	$4.0157281606 \times 10^{-8}$	$1. \times 10^{-11}$	0	0
120	100	30	1088	$4.0147229801 \times 10^{-8}$	0	0	0
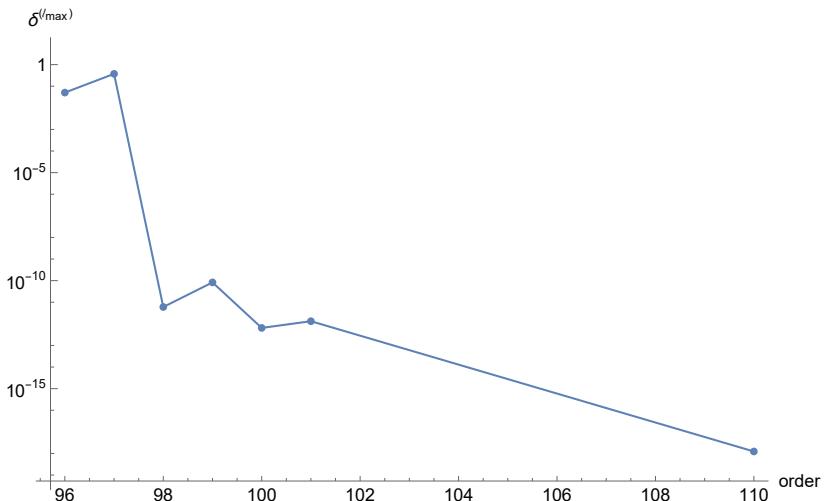
Table 8. Accuracy tests for $\Lambda = 35$ navigator.

- At fixed spins and `order`, results converge rapidly with increasing number of poles. For example, for $\Lambda = 35$, $\ell_{\max} = 88$, `order` $= 100$, increasing the number of poles to any value beyond 25 will change the objective by less than 10^{-22} (see the column $\delta^{(\ell_{\max},\texttt{order})}$ in table 8).

Based on these test results, we chose the following parameters for our computations to ensure that the conformal block approximation and spin truncation induce a relative error which is negligible compared to other sources (e.g. the error due to the Delaunay search):

- $\Lambda = 35$: $\ell_{\max} = 75$, `order` $= 106$, `poles` $= 35$, `prec` $= 832$.

- $\Lambda = 43$: $\ell_{\max} = 90$, `order` $= 130$, `poles` $= 35$, `prec` $= 832$.

$\ell_{\max}$	order	poles	prec	primal objective	δ	$\delta^{(\ell_{\max})}$	$\delta^{(\ell_{\max},\text{order})}$
60	120	50	768	$2.565956605 \times 10^{-8}$	1.6×10^{-10}	0	0
80	120	20	768	$2.5546288291 \times 10^{-8}$	5.1×10^{-11}	1.5×10^{-11}	1.5×10^{-11}
80	120	30	768	$2.5561529711 \times 10^{-8}$	6.6×10^{-11}	9.9×10^{-17}	9.9×10^{-17}
80	120	40	768	$2.5561529613 \times 10^{-8}$	6.6×10^{-11}	5.3×10^{-20}	5.3×10^{-20}
80	120	50	1024	$2.5561529613 \times 10^{-8}$	6.6×10^{-11}	1.9×10^{-24}	1.9×10^{-24}
80	120	60	768	$2.5561529613 \times 10^{-8}$	6.6×10^{-11}	0	0
100	110	30	768	— (SDPB failed)	—	—	—
100	116	30	768	0.8804118909	0.88	0.88	0
100	118	30	768	0.0033975568417	0.0034	0.0034	0
100	120	50	768	$2.5518907348 \times 10^{-8}$	2.3×10^{-11}	$2. \times 10^{-13}$	0
100	120	50	1024	$2.5518907348 \times 10^{-8}$	2.3×10^{-11}	$2. \times 10^{-13}$	0
100	130	50	1024	$2.5519103657 \times 10^{-8}$	2.4×10^{-11}	$8. \times 10^{-18}$	0
100	140	50	1024	$2.5519103665 \times 10^{-8}$	2.4×10^{-11}	9.6×10^{-24}	0
100	150	50	1024	$2.5519103665 \times 10^{-8}$	2.4×10^{-11}	0	0
120	120	50	1024	$2.5495483927 \times 10^{-8}$	0	0	0

Table 9. Accuracy tests for $\Lambda = 43$ navigator.

Figure 10. Primal objective accuracy $\delta^{(\ell_{\max})}$ at fixed $\ell_{\max} = 88$ vs. `order`, $\Lambda = 35$.

Let us emphasize in particular that it is important to systematically increase `order` (the order of the radial expansion of conformal blocks used by `blocks_3d`) as we increase Λ. From our testing, we estimate that $\texttt{order} \sim 5.5\Lambda$ is roughly sufficient to ensure that errors remain small. This might seem surprising, given that the radial expansion for conformal blocks [81] is an expansion in $r = 3 - 2\sqrt{2} \approx 0.17$, which is numerically quite small. However, the coefficient of each power of r is a polynomial in external dimensions, and the growth in

degree of these polynomials plays an important role in determining the error from truncating the series. We encourage users of `blocks_3d` to keep this in mind and be sure to test the effects of `order` on any given bootstrap problem.

Data Availability Statement. This article has no associated data or the data will not be deposited.

Code Availability Statement. This article has code included as electronic supplementary material.

Open Access. This article is distributed under the terms of the Creative Commons Attribution License ([CC-BY4.0](#)), which permits any use, distribution and reproduction in any medium, provided the original author(s) and source are credited.

References

[1] S. Ferrara, A.F. Grillo and R. Gatto, *Tensor representations of conformal algebra and conformally covariant operator product expansion*, *Ann. Phys.* **76** (1973) 161 [INSPIRE].

[2] A.M. Polyakov, *Nonhamiltonian approach to conformal quantum field theory*, *Zh. Eksp. Teor. Fiz.* **66** (1974) 23 [INSPIRE].

[3] K.G. Wilson and M.E. Fisher, *Critical exponents in* 3.99 *dimensions*, *Phys. Rev. Lett.* **28** (1972) 240 [INSPIRE].

[4] A. Pelissetto and E. Vicari, *Critical phenomena and renormalization group theory*, *Phys. Rep.* **368** (2002) 549 [cond-mat/0012164] [INSPIRE].

[5] R. Rattazzi, V.S. Rychkov, E. Tonni and A. Vichi, *Bounding scalar operator dimensions in* 4D *CFT*, *JHEP* **12** (2008) 031 [arXiv:0807.0004] [INSPIRE].

[6] S. El-Showk, M.F. Paulos, D. Poland, S. Rychkov, D. Simmons-Duffin and A. Vichi, *Solving the* 3D *Ising Model with the Conformal Bootstrap*, *Phys. Rev. D* **86** (2012) 025022 [arXiv:1203.6064] [INSPIRE].

[7] D. Poland, S. Rychkov and A. Vichi, *The Conformal Bootstrap: Theory, Numerical Techniques, and Applications*, *Rev. Mod. Phys.* **91** (2019) 015002 [arXiv:1805.04405] [INSPIRE].

[8] D. Poland and D. Simmons-Duffin, *Snowmass White Paper: The Numerical Conformal Bootstrap*, in the proceedings of the *Snowmass 2021*, Seattle, U.S.A., 17–26 July 2022, arXiv:2203.08117 [INSPIRE].

[9] S. Rychkov and N. Su, *New developments in the numerical conformal bootstrap*, *Rev. Mod. Phys.* **96** (2024) 045004 [arXiv:2311.15844] [INSPIRE].

[10] F. Kos, D. Poland and D. Simmons-Duffin, *Bootstrapping Mixed Correlators in the* 3D *Ising Model*, *JHEP* **11** (2014) 109 [arXiv:1406.4858] [INSPIRE].

[11] F. Kos, D. Poland, D. Simmons-Duffin and A. Vichi, *Precision Islands in the Ising and $O(N)$ Models*, *JHEP* **08** (2016) 036 [arXiv:1603.04436] [INSPIRE].

[12] D. Simmons-Duffin, *The Lightcone Bootstrap and the Spectrum of the* 3d *Ising CFT*, *JHEP* **03** (2017) 086 [arXiv:1612.08471] [INSPIRE].

[13] A.L. Fitzpatrick, J. Kaplan, D. Poland and D. Simmons-Duffin, *The Analytic Bootstrap and AdS Superhorizon Locality*, *JHEP* **12** (2013) 004 [arXiv:1212.3616] [INSPIRE].

[14] Z. Komargodski and A. Zhiboedov, *Convexity and Liberation at Large Spin*, *JHEP* **11** (2013) 140 [`arXiv:1212.4103`] [InSPIRE].

[15] A.L. Fitzpatrick, J. Kaplan, M.T. Walters and J. Wang, *Eikonalization of Conformal Blocks*, *JHEP* **09** (2015) 019 [`arXiv:1504.01737`] [InSPIRE].

[16] A. Dymarsky, J. Penedones, E. Trevisani and A. Vichi, *Charting the space of 3D CFTs with a continuous global symmetry*, *JHEP* **05** (2019) 098 [`arXiv:1705.04278`] [InSPIRE].

[17] M. Reehorst, E. Trevisani and A. Vichi, *Mixed Scalar-Current bootstrap in three dimensions*, *JHEP* **12** (2020) 156 [`arXiv:1911.05747`] [InSPIRE].

[18] Y.-C. He, J. Rong, N. Su and A. Vichi, *Non-Abelian currents bootstrap*, *JHEP* **03** (2024) 175 [`arXiv:2302.11585`] [InSPIRE].

[19] A. Dymarsky, F. Kos, P. Kravchuk, D. Poland and D. Simmons-Duffin, *The 3d Stress-Tensor Bootstrap*, *JHEP* **02** (2018) 164 [`arXiv:1708.05718`] [InSPIRE].

[20] M. Hasenbusch, *Restoring isotropy in a three-dimensional lattice model: The Ising universality class*, *Phys. Rev. B* **104** (2021) 014426 [`arXiv:2105.09781`] [InSPIRE].

[21] W. Zhu, C. Han, E. Huffman, J.S. Hofmann and Y.-C. He, *Uncovering Conformal Symmetry in the 3D Ising Transition: State-Operator Correspondence from a Quantum Fuzzy Sphere Regularization*, *Phys. Rev. X* **13** (2023) 021009 [`arXiv:2210.13482`] [InSPIRE].

[22] G. Fardelli, A.L. Fitzpatrick and E. Katz, *Constructing the Infrared Conformal Generators on the Fuzzy Sphere*, `arXiv:2409.02998` [InSPIRE].

[23] M. Hasenbusch, *Two- and three-point functions at criticality: Monte Carlo simulations of the improved three-dimensional Blume-Capel model*, *Phys. Rev. E* **97** (2018) 012119 [`arXiv:1711.10946`] [InSPIRE].

[24] L. Hu, Y.-C. He and W. Zhu, *Operator Product Expansion Coefficients of the 3D Ising Criticality via Quantum Fuzzy Spheres*, *Phys. Rev. Lett.* **131** (2023) 031601 [`arXiv:2303.08844`] [InSPIRE].

[25] S. El-Showk, M.F. Paulos, D. Poland, S. Rychkov, D. Simmons-Duffin and A. Vichi, *Solving the 3d Ising Model with the Conformal Bootstrap II. c-Minimization and Precise Critical Exponents*, *J. Stat. Phys.* **157** (2014) 869 [`arXiv:1403.4545`] [InSPIRE].

[26] V. Ayyar et al., *Operator product expansion for radial lattice quantization of 3D ϕ^4 theory*, *Phys. Rev. D* **109** (2024) 114518 [`arXiv:2311.01100`] [InSPIRE].

[27] D. Poland, V. Prilepina and P. Tadić, *Improving the five-point bootstrap*, *JHEP* **05** (2024) 299 [`arXiv:2312.13344`] [InSPIRE].

[28] J. Bonifacio, *Bootstrap bounds on closed hyperbolic manifolds*, *JHEP* **02** (2022) 025 [`arXiv:2107.09674`] [InSPIRE].

[29] J. Bonifacio, *Bootstrapping closed hyperbolic surfaces*, *JHEP* **03** (2022) 093 [`arXiv:2111.13215`] [InSPIRE].

[30] P. Kravchuk, D. Mazac and S. Pal, *Automorphic spectra and the conformal bootstrap*, *Commun. Am. Math. Soc.* **4** (2024) 1 [`arXiv:2111.12716`] [InSPIRE].

[31] A. Radcliffe, *Non-saturation of Bootstrap Bounds by Hyperbolic Orbifolds*, `arXiv:2404.14479` [InSPIRE].

[32] S.M. Chester et al., *Carving out OPE space and precise O(2) model critical exponents*, *JHEP* **06** (2020) 142 [`arXiv:1912.03324`] [InSPIRE].

[33] S.M. Chester et al., *Bootstrapping Heisenberg magnets and their cubic instability*, *Phys. Rev. D* **104** (2021) 105013 [`arXiv:2011.14647`] [ɪɴSPIRE].

[34] R.S. Erramilli, L.V. Iliesiu, P. Kravchuk, A. Liu, D. Poland and D. Simmons-Duffin, *The Gross-Neveu-Yukawa archipelago*, *JHEP* **02** (2023) 036 [`arXiv:2210.02492`] [ɪɴSPIRE].

[35] P. Kravchuk and D. Simmons-Duffin, *Counting Conformal Correlators*, *JHEP* **02** (2018) 096 [`arXiv:1612.08987`] [ɪɴSPIRE].

[36] A. Dymarsky, *On the four-point function of the stress-energy tensors in a CFT*, *JHEP* **10** (2015) 075 [`arXiv:1311.4546`] [ɪɴSPIRE].

[37] R.S. Erramilli, L.V. Iliesiu and P. Kravchuk, *Recursion relation for general 3d blocks*, *JHEP* **12** (2019) 116 [`arXiv:1907.11247`] [ɪɴSPIRE].

[38] R.S. Erramilli, L.V. Iliesiu, P. Kravchuk, W. Landry, D. Poland and D. Simmons-Duffin, *blocks_3d: software for general 3d conformal blocks*, *JHEP* **11** (2021) 006 [`arXiv:2011.01959`] [ɪɴSPIRE].

[39] M.S. Costa, J. Penedones, D. Poland and S. Rychkov, *Spinning Conformal Correlators*, *JHEP* **11** (2011) 071 [`arXiv:1107.3554`] [ɪɴSPIRE].

[40] M.S. Costa, J. Penedones, D. Poland and S. Rychkov, *Spinning Conformal Blocks*, *JHEP* **11** (2011) 154 [`arXiv:1109.6321`] [ɪɴSPIRE].

[41] G.F. Cuomo, D. Karateev and P. Kravchuk, *General Bootstrap Equations in 4D CFTs*, *JHEP* **01** (2018) 130 [`arXiv:1705.05401`] [ɪɴSPIRE].

[42] D. Poland, V. Prilepina and P. Tadić, *The five-point bootstrap*, *JHEP* **10** (2023) 153 [`arXiv:2305.08914`] [ɪɴSPIRE].

[43] M. Go and Y. Tachikawa, *autoboot: A generator of bootstrap equations with global symmetry*, *JHEP* **06** (2019) 084 [`arXiv:1903.10522`] [ɪɴSPIRE].

[44] D.M. Hofman and J. Maldacena, *Conformal collider physics: Energy and charge correlations*, *JHEP* **05** (2008) 012 [`arXiv:0803.1467`] [ɪɴSPIRE].

[45] D.M. Hofman, D. Li, D. Meltzer, D. Poland and F. Rejon-Barrera, *A Proof of the Conformal Collider Bounds*, *JHEP* **06** (2016) 111 [`arXiv:1603.03771`] [ɪɴSPIRE].

[46] T. Hartman, S. Kundu and A. Tajdini, *Averaged Null Energy Condition from Causality*, *JHEP* **07** (2017) 066 [`arXiv:1610.05308`] [ɪɴSPIRE].

[47] T. Faulkner, R.G. Leigh, O. Parrikar and H. Wang, *Modular Hamiltonians for Deformed Half-Spaces and the Averaged Null Energy Condition*, *JHEP* **09** (2016) 038 [`arXiv:1605.08072`] [ɪɴSPIRE].

[48] S. Bartlett-Tisdall, C.P. Herzog and V. Schaub, *Bootstrapping boundary QED. Part I*, *JHEP* **05** (2024) 235 [`arXiv:2312.07692`] [ɪɴSPIRE].

[49] M.S. Mitchell and D. Poland, *Bounding irrelevant operators in the 3d Gross-Neveu-Yukawa CFTs*, *JHEP* **09** (2024) 134 [`arXiv:2406.12974`] [ɪɴSPIRE].

[50] A.B. Zamolodchikov, *Conformal symmetry in two-dimensional space: Recursion representation of conformal block*, *Theor. Math. Phys.* **73** (1987) 1088 [ɪɴSPIRE].

[51] F. Kos, D. Poland and D. Simmons-Duffin, *Bootstrapping the O(N) vector models*, *JHEP* **06** (2014) 091 [`arXiv:1307.6856`] [ɪɴSPIRE].

[52] J. Penedones, E. Trevisani and M. Yamazaki, *Recursion Relations for Conformal Blocks*, *JHEP* **09** (2016) 070 [`arXiv:1509.00428`] [ɪɴSPIRE].

[53] P. Kravchuk, *Casimir recursion relations for general conformal blocks*, *JHEP* **02** (2018) 011 [arXiv:1709.05347] [INSPIRE].

[54] C.-H. Chang, V. Dommes, P. Kravchuk, D. Poland and D. Simmons-Duffin, *Improved Polynomial Interpolation for Conformal Blocks*, to appear.

[55] M. Reehorst, S. Rychkov, D. Simmons-Duffin, B. Sirois, N. Su and B. van Rees, *Navigator Function for the Conformal Bootstrap*, *SciPost Phys.* **11** (2021) 072 [arXiv:2104.09518] [INSPIRE].

[56] A. Liu, D. Simmons-Duffin, N. Su and B.C. van Rees, *Skydiving to Bootstrap Islands*, arXiv:2307.13046 [INSPIRE].

[57] S. Rychkov, unpublished work.

[58] M. Reehorst, *Rigorous bounds on irrelevant operators in the 3d Ising model CFT*, *JHEP* **09** (2022) 177 [arXiv:2111.12093] [INSPIRE].

[59] N. Bobev, S. El-Showk, D. Mazac and M.F. Paulos, *Bootstrapping the Three-Dimensional Supersymmetric Ising Model*, *Phys. Rev. Lett.* **115** (2015) 051601 [arXiv:1502.04124] [INSPIRE].

[60] J. Henriksson, *The critical $O(N)$ CFT: Methods and conformal data*, *Phys. Rep.* **1002** (2023) 1 [arXiv:2201.09520] [INSPIRE].

[61] F. Caracciolo and V.S. Rychkov, *Rigorous Limits on the Interaction Strength in Quantum Field Theory*, *Phys. Rev. D* **81** (2010) 085037 [arXiv:0912.2726] [INSPIRE].

[62] D. Poland and D. Simmons-Duffin, *Bounds on 4D Conformal and Superconformal Field Theories*, *JHEP* **05** (2011) 017 [arXiv:1009.2087] [INSPIRE].

[63] S. El-Showk and M.F. Paulos, *Bootstrapping Conformal Field Theories with the Extremal Functional Method*, *Phys. Rev. Lett.* **111** (2013) 241601 [arXiv:1211.2810] [INSPIRE].

[64] S. Caron-Huot, *Analyticity in Spin in Conformal Theories*, *JHEP* **09** (2017) 078 [arXiv:1703.00278] [INSPIRE].

[65] S. Caron-Huot, Y. Gobeil and Z. Zahraee, *The leading trajectory in the $2+1D$ Ising CFT*, *JHEP* **02** (2023) 190 [arXiv:2007.11647] [INSPIRE].

[66] J. Liu, D. Meltzer, D. Poland and D. Simmons-Duffin, *The Lorentzian inversion formula and the spectrum of the 3d $O(2)$ CFT*, *JHEP* **09** (2020) 115 [*Erratum ibid.* **01** (2021) 206] [arXiv:2007.07914] [INSPIRE].

[67] A. Atanasov, A. Hillman, D. Poland, J. Rong and N. Su, *Precision bootstrap for the $\mathcal{N}=1$ super-Ising model*, *JHEP* **08** (2022) 136 [arXiv:2201.02206] [INSPIRE].

[68] M. Kologlu, P. Kravchuk, D. Simmons-Duffin and A. Zhiboedov, *The light-ray OPE and conformal colliders*, *JHEP* **01** (2021) 128 [arXiv:1905.01311] [INSPIRE].

[69] N. Seiberg, T. Senthil, C. Wang and E. Witten, *A Duality Web in $2+1$ Dimensions and Condensed Matter Physics*, *Ann. Phys.* **374** (2016) 395 [arXiv:1606.01989] [INSPIRE].

[70] C. Turner, *Dualities in $2+1$ Dimensions*, *PoS* **Modave2018** (2019) 001 [arXiv:1905.12656] [INSPIRE].

[71] C. Behan, L. Rastelli, S. Rychkov and B. Zan, *Long-range critical exponents near the short-range crossover*, *Phys. Rev. Lett.* **118** (2017) 241601 [arXiv:1703.03430] [INSPIRE].

[72] C. Behan, L. Rastelli, S. Rychkov and B. Zan, *A scaling theory for the long-range to short-range crossover and an infrared duality*, *J. Phys. A* **50** (2017) 354002 [arXiv:1703.05325] [INSPIRE].

[73] N. Su, *The Hybrid Bootstrap*, arXiv:2202.07607 [INSPIRE].

[74] W. Li, *Easy bootstrap for the 3D Ising model: a hybrid approach of the lightcone bootstrap and error minimization methods*, *JHEP* **07** (2024) 047 [arXiv:2312.07866] [INSPIRE].

[75] S. Harris, A. Kaviraj, J.A. Mann, L. Quintavalle and V. Schomerus, *Comb channel lightcone bootstrap: triple-twist anomalous dimensions*, *JHEP* **08** (2024) 122 [arXiv:2401.10986] [INSPIRE].

[76] A. Antunes, S. Harris, A. Kaviraj and V. Schomerus, *Lining up a positive semi-definite six-point bootstrap*, *JHEP* **06** (2024) 058 [arXiv:2312.11660] [INSPIRE].

[77] King's College London, *King's Computational Research, Engineering and Technology Environment (CREATE)*, (2024) [DOI:10.18742/rnvf-m076].

[78] D. Mazac, *Analytic bounds and emergence of AdS_2 physics from the conformal bootstrap*, *JHEP* **04** (2017) 146 [arXiv:1611.10060] [INSPIRE].

[79] P. Capriotti and A. Kaposi, *Free Applicative Functors*, in the proceedings of the *5th Workshop on Mathematically Structured Functional Programming, MSFP@ETAPS 2014*, Grenoble, France, 12 April 2014, P.B. Levy and N. Krishnaswami eds., Open Publishing Association (2014) [*Electron. Proc. Theor. Comput. Sci.* **153** (2014) 2].

[80] A. Mokhov, N. Mitchell and S. Peyton Jones, *Build systems à la carte*, *Proc. ACM Program. Lang.* **2** (2018) 1.

[81] M. Hogervorst and S. Rychkov, *Radial Coordinates for Conformal Blocks*, *Phys. Rev. D* **87** (2013) 106004 [arXiv:1303.1111] [INSPIRE].