

Efficient conformal block evaluation with a novel software implementation

J. Chryssanthacopoulos,^{1,*} V. Niarchos^{2,†} C. Papageorgakis^{1,‡} and A. G. Stapleton^{1,§}

¹*Centre for Theoretical Physics and Astronomy, Department of Physics and Astronomy
Queen Mary University of London, London E1 4NS, United Kingdom*

²*Department of Physics, ITCP & CCTP University of Crete, 71003 Heraklion, Greece*



(Received 28 April 2026; accepted 25 June 2026; published 15 July 2026)

Conformal blocks in odd spacetime dimensions are not known in closed analytic form. To facilitate efficient computations in the conformal bootstrap, we introduce GOBLOCKS: a novel conformal-block generator implemented in the Go programming language, designed for rapid, on-the-fly, parallel evaluation in general spacetime dimensions using recursive relations. The package supports both multipoint and derivative-based bootstrap approaches and allows flexible control over accuracy and performance. We benchmark GOBLOCKS against the SCALAR_BLOCKS package, finding significant speed improvements in applications where computational speed and moderate accuracy are critical, but ultrahigh precision is not essential. As an illustration, we apply GOBLOCKS to the mixed-correlator bootstrap of the three-dimensional Ising model, formulated as a nonconvex optimization problem in a suitable truncation scheme. We simultaneously optimize over external scaling dimensions and OPE CFT data. In addition, we discuss how the approach scales as we increase the number of mixed correlators in more general $O(N)$ vector models.

DOI: [10.1103/p4sb-x44g](https://doi.org/10.1103/p4sb-x44g)

I. INTRODUCTION AND SUMMARY

The numerical conformal bootstrap has proven to be a powerful nonperturbative tool for rigorously constraining the space of conformal field theories (CFTs) [1]. When multiple correlators are bootstrapped simultaneously, it can determine CFT data with remarkable accuracy. The most celebrated success of this approach is for the four-point bootstrap of the three-dimensional Ising model [2–4], which proceeds by scanning over spaces of linear functionals (dual formulation), rather than directly solving the crossing equations (primal formulation). While successful, the linear-functional bootstrap relies crucially on certain positivity assumptions. However, these assumptions are not present in several interesting CFT contexts, e.g., in the presence of defects [5], boundaries [6], at finite temperature [7], and for the five-point bootstrap [8]. This has motivated a revisit of the primal approach through truncations [9–15] and other approximate, deep-learning-based methods [16].

At a technical level, both formulations require an efficient computation of conformal blocks. In the linear-functional approach, one fixes the dimensions of external operators at each step and employs linear or semidefinite programming tools such as SDPB [17]. These typically require expanding the blocks around the crossing-symmetric point—where the OPE converges rapidly—in terms of rational approximations to very high precision and evaluating derivatives up to some order. For the bootstrap of four-point scalar correlators, the *de facto* standard is SCALAR_BLOCKS [18], while `blocks_3d` [19] provides blocks for correlators involving operators in arbitrary Lorentz representations. These blocks are then fed into semidefinite programming tools, like SDPB, which typically require high numerical precision to avoid numerical instabilities. We note that there are also related multipoint approaches where the crossing equations are evaluated on a discrete grid of points in the conformal cross-ratio plane, rather than through derivatives [20,21].

In primal truncation schemes, one needs to optimise over a high-dimensional space of CFT data that may include the external scaling dimensions of the correlators as dynamical variables. In even dimensions, where conformal blocks are known in closed form, the analytic expressions can be used directly for fast, on-the-fly evaluation. However, in odd dimensions—such as the physically interesting case of 3D—the SCALAR_BLOCKS package is prohibitively slow for this purpose and alternative computational tools are

*Contact author: j.chryssanthacopoulos@qmul.ac.uk

†Contact author: niarchos@physics.uoc.gr

‡Contact author: c.papageorgakis@qmul.ac.uk

§Contact author: a.g.stapleton@qmul.ac.uk

Published by the American Physical Society under the terms of the [Creative Commons Attribution 4.0 International](https://creativecommons.org/licenses/by/4.0/) license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI. Funded by SCOAP³.

needed.¹ Often in these schemes, high numerical accuracy in the conformal-block evaluation is not the primary concern, as there are already other sources of systematic and stochastic errors that need to be addressed. Despite these shortcomings, there are examples where truncation methods achieve high-accuracy predictions comparable to state-of-the-art results, see, e.g., [22,23].

In this work, we develop efficient methods for conformal-block evaluation with generic parameters that include the scaling dimensions of external operators. We present GOBLOCKS: a fast, parallel, and flexible conformal-block generator implemented in the Go programming language.² We demonstrate its use in both the multipoint and derivative-based bootstrap of scalar correlation functions in D dimensions.

As an application, we demonstrate how GOBLOCKS performs in a nonconvex optimization treatment of multiple mixed correlators in the 3D Ising model. Optimization is performed using the companion package BOOTSTOP-MULTI-CORRELATOR, a multiobjective implementation of the bootstrap stochastic optimizer, BOOTSTOP [12,13,23]. GOBLOCKS includes a simple Python interface which, in principle, can be used with any optimization tool independent of BOOTSTOP. We also analyze the relative performance, stability, and computational trade-offs of the multipoint and derivative strategies.

The rest of this paper is organized as follows. Section II provides an overview of GOBLOCKS, including its multipoint and derivative formulations. It also discusses benchmarking against SCALAR_BLOCKS in terms of accuracy and runtime. Section III then summarizes the results of an implementation of the package to the bootstrap of the conformal data of the 3D Ising model, and remarks on the scalability to other $O(N)$ vector models. Section IV concludes with a summary and avenues for future research.

II. OVERVIEW OF GOBLOCKS

The crossing equations that give rise to the conformal bootstrap program for four-point correlators depend on the conformal cross-ratio plane, parametrized by $(z, \bar{z})$. There are two main approaches for the numerical implementation of the bootstrap in the literature. The most popular method involves computing derivatives of the crossing equations at the crossing-symmetric point, which we call the derivative approach. An alternative method involves working with the crossing equations at specific points in the cross-ratio plane, which we term the multipoint approach [20,21]. The numerical optimization of CFT data in both approaches relies on the accurate evaluation of conformal blocks.

¹It should be emphasized that SCALAR_BLOCKS is a very efficient tool for the dual formulation of the numerical conformal bootstrap, where its output can be used directly. For the primal bootstrap, additional processing must be performed on the output, as described below.

²The package can be accessed from GitHub at <https://github.com/xand-stapleton/goblocks> [24].

The state-of-the-art method for the calculation of scalar conformal blocks is SCALAR_BLOCKS, an efficient C++ implementation of the Zamolodchikov recursion relations capable of producing conformal blocks with exceptional precision [18]. This package computes derivatives of conformal blocks at the crossing-symmetric point $z = \bar{z} = 1/2$

$$\partial_z^m \partial_{\bar{z}}^n g_{\Delta, \ell}^{\Delta_{ij}, \Delta_{kl}}(z, \bar{z})|_{(z, \bar{z})=\frac{1}{2}} \approx \frac{r_0^\Delta}{Q_{\kappa, \ell}(\Delta)} P_{\ell, \kappa, N; m, n}^{\Delta_{ij}, \Delta_{kl}}(x), \quad (2.1)$$

where $\Delta_{ij} := \Delta_i - \Delta_j$ and $\Delta_{kl} := \Delta_k - \Delta_l$ denote differences in scaling dimensions of scalar operators in the four-point function $\langle \mathcal{O}_i \mathcal{O}_j \mathcal{O}_k \mathcal{O}_l \rangle$. The subscripts N and κ denote the order of the r expansion and pole order, while ℓ and Δ represent the spin and dimension of the exchanged operator. The functions $Q(\Delta)$ and $P(x)$ are polynomials, with $x := \Delta - (\ell + D - 2)$, and $r_0 := 3 - 2\sqrt{2}$ the crossing-symmetric point in radial coordinates. This rational form allows evaluation at arbitrary Δ without rerunning the recursion.

Three important parameters in SCALAR_BLOCKS are `max-derivs`, `order`, and `poles`. The parameter `max-derivs` encodes $n_{\max}$, defined such that $m + n \leq 2n_{\max} - 1$. The parameter `order`, given by N above, sets the maximum order of the r expansion along the $z = \bar{z}$ diagonal. Similarly, the parameter `poles` specifies how many poles to retain among those whose residues start with r^n , $n \leq \kappa$. There is also a `precision` parameter, which controls the numerical precision of the output and calculations internal to SCALAR_BLOCKS. In this work, `precision` is set to 64 unless otherwise stated. The package returns the expansion coefficients of P and the poles Δ_i , which can be used to construct $P(x)$ and $Q(\Delta)$, rather than the derivatives of g directly.

In many applications, one needs to vary dynamically over the unknown scaling dimensions of the external operators, entering as extra parameters in the blocks. This requires direct, repeated application of SCALAR_BLOCKS, which is typically prohibitively slow. Since SCALAR_BLOCKS returns the ingredients to assemble the conformal blocks, additional processing needs to be performed to recover the blocks themselves, introducing computational overhead.³

To circumvent this difficulty, a potential alternative approximation of the blocks could use multilinear or polynomial interpolation. These approaches precompute block values on a fixed grid of scaling dimensions and interpolate to estimate values at arbitrary points. While interpolation is computationally efficient, its accuracy for highly nonlinear functions, such as the conformal blocks, is limited by the grid resolution, and increasing the resolution to improve accuracy substantially increases memory requirements. For a quantitative assessment of multilinear and polynomial interpolation accuracy, see Appendix A.

³The package BOOTSTOP-MULTI-CORRELATOR includes an efficient JAX implementation of the necessary processing steps.

TABLE I. Configuration parameters of GOBLOCKS using the multipoint and derivative approaches. The derivative approach shares all the same parameters as the points approach, but has additional parameters controlling the number of derivatives computed and returned.

Parameter	Approach	Description
k_1_max	Points, derivatives	Maximum order of Type I poles
k_2_max	Points, derivatives	Maximum order of Type II poles
ell_min	Points, derivatives	Minimum spin in the recursion
ell_max	Points, derivatives	Maximum spin in the recursion
max_iterations	Points, derivatives	Maximum Taylor order (r -series)
tol	Points	Tolerance for early termination
n_max	Derivatives	Maximum derivative order
num_derivs_to_keep	Derivatives	Number of derivatives to return

Fortunately, recursive relations for evaluating conformal blocks can be iterated efficiently provided they are implemented in a sufficiently performant language. To that end, we developed GOBLOCKS, an efficient, parallel, multithreaded block generator written in Go. The current implementation of GOBLOCKS supports both multipoint and derivative approaches, outlined in Secs. II A and II B, respectively.

A. Multipoint approach

For the implementation of the multipoint approach, we work in radial coordinates (r, η) , which are related to the cross-ratios by

$$re^{i\theta} = \frac{z}{(1 + \sqrt{1 - z})^2}, \quad \eta = \cos \theta. \quad (2.2)$$

The conformal blocks in these coordinates can be written as

$$h_{\Delta, \ell}^{\Delta_{ij}, \Delta_{kl}}(r, \eta) = r^{-\Delta} g_{\Delta, \ell}^{\Delta_{ij}, \Delta_{kl}}(r, \eta) = \tilde{h}_{\ell}^{\Delta_{ij}, \Delta_{kl}}(r, \eta) + \sum_i \frac{c_i^{\Delta_{ij}, \Delta_{kl}}}{\Delta - \Delta_i} r^{n_i} h_{\Delta_i + n_i, \ell_i}^{\Delta_{ij}, \Delta_{kl}}(r, \eta), \quad (2.3)$$

where Δ_i are the positions of the poles with associated data n_i , ℓ_i , and $c_i^{\Delta_{ij}, \Delta_{kl}}$. There are three types of poles, with the corresponding data summarized in Table 1 of [3]. The function $\tilde{h}$ seeds the recursion with initial data, and can be written in terms of a Gegenbauer polynomial. See Eq. (4.6) of [3] for a complete expression. To solve Eq. (2.3), we expand h in a Taylor series in r around $r = 0$ and compute the coefficients order by order. At each order p , the contribution from a pole with shift n_i uses only coefficients at order $p - n_i < p$, which have already been determined exactly.⁴

⁴Alternatively, the recursion relation (2.3) can be solved naively using a fixed-point scheme in which all r -orders are iterated over simultaneously at each step. This technique was employed in the original version of this work, but it produced divergences at large $|\Delta_{ij}|$, $|\Delta_{kl}|$. Computing the coefficients order-by-order in the r -series expansion instead determines each coefficient exactly from previously fixed lower-order data, removing the divergences. This also improves GOBLOCKS accuracy across the parameter space tested and yields speed gains. We thank S. Rychkov for communication on this point.

The blocks are returned in terms of the following linear combinations, which appear directly in the crossing equations

$$F_{\pm, \Delta, \ell}^{ij, kl}(u, v) := v^{\frac{\Delta_j + \Delta_k}{2}} g_{\Delta, \ell}^{\Delta_{ij}, \Delta_{kl}}(u, v) \pm u^{\frac{\Delta_j + \Delta_k}{2}} g_{\Delta, \ell}^{\Delta_{ij}, \Delta_{kl}}(v, u), \quad (2.4)$$

where $u := z\bar{z}$ and $v := (1 - z)(1 - \bar{z})$. These functions depend on four scaling dimensions: Δ_{ij} , Δ_{kl} , $\bar{\Delta}_{jk} := (\Delta_j + \Delta_k)/2$, and Δ .

The multipoint algorithm consists of three steps: recursion, block evaluation, and construction of $F_{\pm}$. These steps are depicted in Fig. 5 of Appendix B. In the recursion step, the blocks $h_{\Delta, \ell}^{\Delta_{ij}, \Delta_{kl}}$ are computed at a set of pole locations given the external operator dimensions Δ_{ij} and Δ_{kl} . This is the most computationally intensive step. The evaluation step then accepts a list of spins and scaling dimensions of exchanged operators, and uses the recurse-step results to compute the value of g for each operator. In the construction step, the $F_{\pm}$ are constructed from Eq. (2.4). Since Eq. (2.4) requires both $g(u, v)$ and $g(v, u)$, the recursion step needs to be run twice with different arguments. The algorithm returns $F_{\pm}$ at the set of points in the z plane that were requested.

Table I summarizes the parameters used by GOBLOCKS in the multipoint approach. The parameters k_1_max and k_2_max set the maximum number of Type I and Type II poles for recursion, analogous to the poles parameter in SCALAR_BLOCKS. The max_iterations parameter sets the maximum Taylor order in the r -series expansion,⁵ playing a role analogous to the order parameter in SCALAR_BLOCKS. The tol parameter enables early termination: if the maximum fractional contribution of the latest Taylor term falls below tol, the series is considered converged. To ensure robust convergence, the early termination check is deferred until a sufficient number of terms have been computed. The parameter ell_max sets the maximum spin included in the recursion; for optimal

⁵More specifically, the order is determined by $\max(2 \times k_2_max, \text{max_iterations})$, which ensures all pole types have contributed.

TABLE II. Percent errors of the GOBLOCKS multipoint approach relative to SCALAR_BLOCKS for different parameters, defined in Table I. Statistics are aggregated across all spins up to six and for 1,000 points in the $(\Delta_{ij}, \Delta_{kl}, \bar{\Delta}_{jk}, \Delta)$ plane with $|\Delta_{ij}|, |\Delta_{kl}| \leq 3.5$.

Parameters		Error (%)				
$k_{1,2}^{\max}$	$\ell_{\max}$	25%	50%	75%	90%	Mean
10	10	5.517×10^{-6}	1.875×10^{-4}	9.039×10^{-3}	1.241×10^{-1}	8.157×10^{-2}
10	16	1.220×10^{-8}	4.839×10^{-8}	1.159×10^{-6}	1.837×10^{-5}	2.077×10^{-5}
10	20	4.588×10^{-9}	1.165×10^{-8}	2.744×10^{-8}	6.375×10^{-8}	4.190×10^{-8}
16	10	5.517×10^{-6}	1.875×10^{-4}	9.039×10^{-3}	1.241×10^{-1}	8.157×10^{-2}
16	20	1.687×10^{-12}	1.520×10^{-11}	1.174×10^{-9}	1.914×10^{-8}	2.539×10^{-8}
20	20	1.551×10^{-12}	1.505×10^{-11}	1.174×10^{-9}	1.914×10^{-8}	2.539×10^{-8}

accuracy, it should be set larger than the highest spin of interest, since poles of spin ℓ reference spins up to $\ell + k_{1,2}^{\max}$.

B. Derivative approach

An alternative to the multipoint approach is to use derivatives of the conformal blocks at the crossing-symmetric point, as in SCALAR_BLOCKS. Appendix C derives a recursive formula for block derivatives with respect to (r, η) at the crossing-symmetric point. Converting to derivatives with respect to $(z, \bar{z})$ requires derivatives of r and η with respect to $(z, \bar{z})$, which are derived in Appendix D. The derivative approach in GOBLOCKS returns derivatives of $F_{\pm}$. From Eq. (2.4), these derivatives can be written in terms of derivatives of g as

$$\partial_z^m \partial_{\bar{z}}^n F_{\pm, \Delta, \ell}^{ij, kl}(z, \bar{z}) = (1 \pm (-1)^{m+n}) \sum_{p=0}^m \sum_{q=0}^n \binom{m}{p} \times \binom{n}{q} c_{pq}^{\bar{\Delta}_{jk}} \partial_z^{m-p} \partial_{\bar{z}}^{n-q} g_{\Delta, \ell}^{\Delta_{ij}, \Delta_{kl}}(z, \bar{z}), \quad (2.5)$$

where $c_{ij}^{\alpha} := (-1)^{i+j} 2^{i+j-2\alpha} (1+\alpha-i)_i (1+\alpha-j)_j$, with $(x)_n$ the rising factorial. For the same number of constraints (i.e., points and derivatives), the runtimes of the multipoint and derivative approaches are comparable when run with identical shared hyperparameters (e.g., $\ell_{\max}, k_{1,2}^{\max}$).⁶

The algorithmic flow of the derivative approach is depicted in Fig. 6 of Appendix B. The recursion and evaluation steps are very similar to the multipoint approach. In the construction of the $F_{\pm}$ derivatives, there is an additional step to convert from derivatives of g with respect to (r, η) to derivatives with respect to $(z, \bar{z})$. These steps involve combinatorial sums whose cost grows steeply with $n_{\max}$ and introduce the dominant computational cost in the derivative approach. To mitigate this, the required combinatorial factors are

⁶Empirically, for the same number of constraints, running on x64 architectures appears to give a slight (approximately 25%) performance advantage to the multipoint method over the derivative approach. This has not been observed on ARM architectures running MacOS.

precomputed and cached, so that subsequent evaluations at different operator dimensions are fast. GOBLOCKS also supports GPU acceleration via CUDA for the matrix operations in the conversion step, which typically becomes advantageous when $n_{\max}$ is large.

Table I summarizes the parameters in the derivative approach. This method shares all parameters with the multipoint approach but introduces two new controls: the maximum derivative order, $n_{\max}$, and the number of derivatives retained, `num_derivs_to_keep`. The maximum derivative order, $n_{\max}$, is consistent with its usage in SCALAR_BLOCKS. The parameter `num_derivs_to_keep` governs the subset of computed derivatives that are returned, directly impacting numerical accuracy. By selecting a smaller subset for a fixed $n_{\max}$, the retained derivatives exhibit higher precision compared to retaining a larger set.

C. Accuracy

We next present a comparison of the accuracy of GOBLOCKS and SCALAR_BLOCKS over a range of scaling dimensions. The output from SCALAR_BLOCKS, computed with large parameter values, serves as the reference for evaluating the accuracy of GOBLOCKS. The parameters used for SCALAR_BLOCKS were `order` = 90, `precision` = 90, `poles` = 20, and `max-derivs` = 14. In practice, it is preferable to compare the output of GOBLOCKS and SCALAR_BLOCKS using $F_{\pm, \Delta, \ell}$ rather than $g_{\Delta, \ell}$ directly. This approach reduces noise in the estimates, especially when g is small, and provides smoother results for F and its derivatives.

Multipoint tests: Table II presents the percent errors of the GOBLOCKS multipoint approach relative to SCALAR_BLOCKS for various parameters. The comparison is performed at the crossing-symmetric point $z = \bar{z} = 1/2$, the only nonderivative term returned by SCALAR_BLOCKS, corresponding to $m = n = 0$. The results focus on $F_{+, \Delta, \ell}$ from both methods. The relative error is defined as the difference between the two values, normalized by the value of $F_{+, \Delta, \ell}$ from SCALAR_BLOCKS. For each spin up to six, 1,000 points in the $(\Delta_{ij}, \Delta_{kl}, \bar{\Delta}_{jk}, \Delta)$ space were evaluated with $|\Delta_{ij}|, |\Delta_{kl}| \leq 3.5$. The table reports the error percentiles and the mean. Across all parameters tested, the errors remain well below a tenth of a percent. Increasing $\ell_{\max}$ significantly

TABLE III. Percent errors of the GOBLOCKS derivative approach relative to SCALAR_BLOCKS for different parameters, defined in Table I. The statistics are aggregated across all spins up to six and for 1,000 points in the $(\Delta_{ij}, \Delta_{kl}, \bar{\Delta}_{jk}, \Delta)$ plane with $|\Delta_{ij}|, |\Delta_{kl}| \leq 3.5$. The statistics are also aggregated over errors for both F_+ and F_- .

Parameters				Error (%)				
$k_{1,2}^{\max}$	$\ell_{\max}$	$n_{\max}$	n_{deriv}	25%	50%	75%	90%	Mean
10	16	8	21	4.748×10^{-5}	5.000×10^{-4}	4.000×10^{-3}	2.570×10^{-2}	9.730×10^{-2}
10	16	9	28	1.034×10^{-4}	1.200×10^{-3}	9.300×10^{-3}	5.710×10^{-2}	2.142×10^{-1}
10	20	8	21	8.518×10^{-6}	7.437×10^{-5}	5.963×10^{-4}	3.731×10^{-3}	1.989×10^{-2}
10	20	9	28	1.837×10^{-5}	1.839×10^{-4}	1.571×10^{-3}	1.008×10^{-2}	6.874×10^{-2}
16	20	8	21	1.463×10^{-7}	3.310×10^{-6}	5.206×10^{-5}	5.614×10^{-4}	3.754×10^{-3}
16	20	9	28	4.244×10^{-7}	9.964×10^{-6}	1.553×10^{-4}	1.574×10^{-3}	2.365×10^{-2}
20	20	8	21	8.202×10^{-8}	2.657×10^{-6}	4.804×10^{-5}	5.000×10^{-4}	3.800×10^{-3}
20	20	9	28	2.547×10^{-7}	8.003×10^{-6}	1.403×10^{-4}	1.500×10^{-3}	2.370×10^{-2}
25	25	8	21	1.827×10^{-10}	6.949×10^{-9}	1.869×10^{-7}	2.787×10^{-6}	2.128×10^{-5}
25	25	9	28	6.346×10^{-10}	2.628×10^{-8}	7.267×10^{-7}	1.152×10^{-5}	1.913×10^{-4}

improves performance, but accuracy is relatively insensitive to $k_{1,2}^{\max}$.

As a complementary validation, we also compared the output of GOBLOCKS against the closed-form ${}_3F_2$ expression for conformal blocks on the diagonal $z = \bar{z}$, which is valid for $\Delta_{ij} = 0$ and was derived in [20]. This provides an independent cross-check using an exact analytic result.⁷ We evaluated $F_{\pm, \Delta, \ell}$ for a grid of 36 parameter combinations: three values of $\bar{\Delta}_{jk}$ while keeping $\Delta_{ij} = 0$ (including the equal external-dimension case $\Delta_i = \Delta_j = \Delta_k = \Delta_l$), four spins $\ell \in \{0, 1, 2, 4\}$, and three internal operator dimensions $\Delta \in \{1.5, 2.5, 4.5\}$, each at nine diagonal points in the range $z \in [0.3, 0.7]$. All 324 comparisons for F_+ and all 288 nonvanishing comparisons for F_- achieve relative errors below 10^{-5} , with median errors of 5.0×10^{-9} and 2.4×10^{-8} , respectively. The error scales with the distance from the crossing-symmetric point, ranging from $\sim 10^{-10}$ at $z = 1/2$ to $\sim 10^{-7}$ at $|z - 1/2| = 0.2$, and increases modestly with spin, from $\sim 10^{-10}$ at $\ell = 0$ to $\sim 10^{-8}$ at $\ell = 4$.

As a further validation beyond the diagonal, we compared the multipoint output of GOBLOCKS against a high-precision *Mathematica* implementation of the same Zamolodchikov recursion, which uses 200-digit working precision and the order-by-order r -series approach. This comparison covers complex z -points (with $\bar{z} = z^*$) and nonzero $\Delta_{ij} < 3$, $\Delta_{kl} < 3$, regimes inaccessible to the ${}_3F_2$ formula. Across a number of test points spanning the OPE convergence region and a range of Δ_{ij} , Δ_{kl} , all comparisons with $\ell_{\max} = 10$ achieve relative errors below 10^{-3} . Extending to $\ell_{\max} = 20$ brings the most demanding tested cases (of simultaneous $\Delta_{ij} = 3$ and $\Delta_{kl} = 3$ at $z = 0.7$) to $\sim 4 \times 10^{-7}$.

⁷The two methods use different normalization conventions, related by $F_{\pm}^{\text{GOBLOCKS}} = (-1)^{\ell} r_0^{\Delta} F_{\pm}^{F_2}$. All errors were computed after applying this rescaling.

Derivative tests: Table III reports percent errors of the GOBLOCKS derivative approach relative to SCALAR_BLOCKS for different parameters. In this case, the errors are aggregated across $\partial_z \partial_{\bar{z}} F_{+, \Delta, \ell}$ and $\partial_z \partial_{\bar{z}} F_{-, \Delta, \ell}$. At $n_{\max} = 8$, $n_{\text{deriv}} = 21$ derivatives can be supported at high accuracy. The error beyond the 21st derivative increases considerably. At $n_{\max} = 9$, 28 derivatives can be reasonably supported. Although the performance of the derivative approach is very good overall, it is inferior to the multipoint approach. Increasing $n_{\max}$ while keeping the other parameters fixed allows more derivatives to be captured, but generally comes at the price of worse performance overall.

D. Timing

In numerical computations, there is a natural trade-off between speed and accuracy. The higher the accuracy one wishes to achieve, the longer the block-evaluation time. Quantifying this trade-off involves measuring accuracy as a function of runtime, which requires a robust reference standard. In this work, the benchmark is defined by the output of SCALAR_BLOCKS evaluated at an order of 200. This high-precision result is treated as ground truth for systematic comparison, enabling both SCALAR_BLOCKS and the GOBLOCKS derivative approach to be evaluated across a range of parameter settings. All computations were performed on an Intel Core i5-12400 processor running Arch Linux with kernel version 6.17.

Each evaluation involved sampling 100 points in the $(\Delta_{ij}, \Delta_{kl}, \bar{\Delta}_{jk}, \Delta)$ parameter space with six different spins. Parameters shared by both algorithms were set to the same value. For example, `poles` in SCALAR_BLOCKS was matched to `k_1_max` and `k_2_max` in GOBLOCKS, and `max-derivs` was matched to `n_max` and `num-derivs_to_keep`. The main parameters varied were `order` in SCALAR_BLOCKS, and `ell_max` and `tol` in GOBLOCKS.

Fig. 1 presents the runtime of SCALAR_BLOCKS as a function of the order parameter, with the shaded region

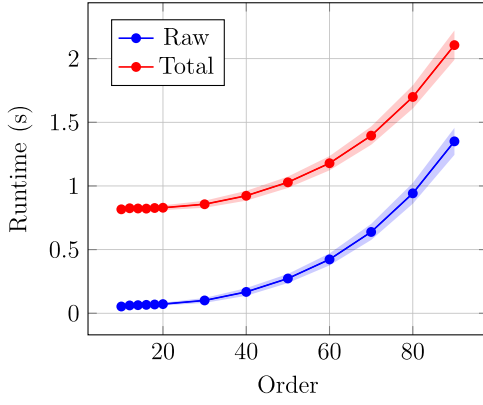


FIG. 1. Runtime of SCALAR_BLOCKS as order varies, with and without postprocessing to construct block derivatives and convert to derivatives of $F_{\pm}$. Without postprocessing, SCALAR_BLOCKS runs in under a second for most order values. With postprocessing, the runtime can increase to two seconds.

indicating the standard deviation across multiple runs. As described previously, SCALAR_BLOCKS outputs expansion coefficients for polynomials centred at the crossing-symmetric point. To obtain the final block derivatives, further processing is required. Derivatives of g are constructed using Eq. (2.1), and then converted to derivatives of $F_{\pm}$ via

Eq. (2.5). This postprocessing was implemented in Python using a multicore JAX backend. The figure displays runtimes both with and without this postprocessing step. Without postprocessing, SCALAR_BLOCKS produces expansion coefficients in under a second for order up to 90. Including postprocessing, runtimes increase substantially, making real-time block evaluation impractical for some applications. In subsequent runtime reports for SCALAR_BLOCKS, only the unprocessed output times are shown, representing the absolute lower bound for evaluation time.

Figure 2 shows the performance of GOBLOCKS derivatives for varying hyperparameters for $n_{\max} = 8$. The parameters $k_{1,2}^{\max}$ and $\ell_{\max}$ have the largest impact on overall accuracy. Figure 2(a) shows the average associated deviation from ground truth for varied $\ell_{\max}$ and fixed $k_{1,2}^{\max} = 20$. The deviation is computed as the L^2 norm of the element-wise difference between the GOBLOCKS results and the benchmark, normalized by the benchmark. Similarly, Fig. 2(b) presents the average runtime of GOBLOCKS for varied $\ell_{\max}$ and fixed $k_{1,2}^{\max} = 20$. We stress that these configurations do not yield the best accuracy-to-runtime proposition, but are instead intended to present an example of how $\ell_{\max}$ affects runtime and block accuracy.

Figure 3 compares the average runtime of SCALAR_BLOCKS and GOBLOCKS at fixed block accuracy. In practice, to achieve

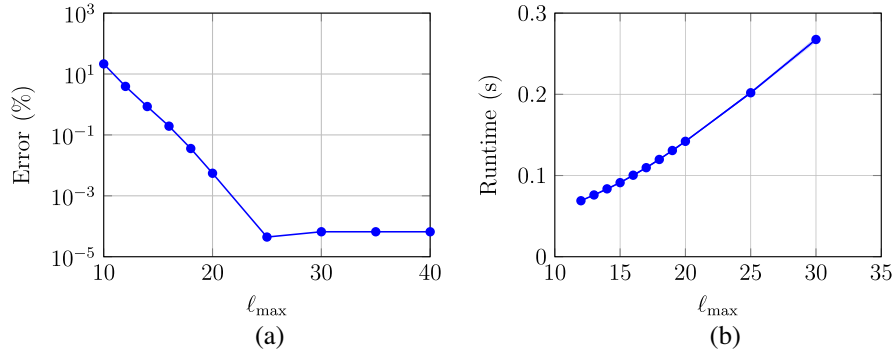


FIG. 2. Accuracy and runtime of GOBLOCKS with increasing $\ell_{\max}$. (a) shows block accuracy varying from $\sim 20\%$ to $\sim 4 \times 10^{-5}\%$ as $\ell_{\max}$ increases from 12 to 30 for fixed $k_{1,2}^{\max} = 20$. (b) shows runtime as a function of $\ell_{\max}$ for fixed $k_{1,2}^{\max} = 20$.

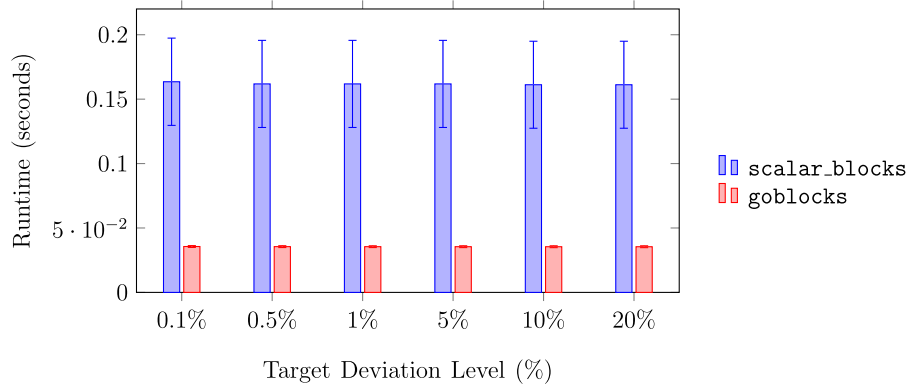


FIG. 3. Comparison of average runtimes between SCALAR_BLOCKS and GOBLOCKS for given accuracies with $k_{1,2}^{\max} = 10$. Overall, GOBLOCKS is approximately 4.6 times faster than SCALAR_BLOCKS between the 0.1% and 10% accuracy levels.

the optimal accuracy-to-speed ratio, the parameters $k_{1,2}^{\max}$ and $\ell_{\max}$ must be tuned. To achieve the results stated in Fig. 3, parameter values $k_{1,2}^{\max} = 10$ and $\ell_{\max} = 16$ were used. Up to the $10^{-4}\%$ accuracy level, we were able to find combinations of the triple $k_{1,2}^{\max}, \ell_{\max}$ which were approximately 4.6 times faster than SCALAR_BLOCKS. Hyperparameter searches for higher accuracy levels were not performed. Appendix E presents additional information on parameter tuning. The reported GOBLOCKS runtime includes the additional step of converting to derivatives of $F_{\pm}$, while the SCALAR_BLOCKS runtime reflects only the generation of raw output. Applications requiring fully constructed derivatives of g or $F_{\pm}$ would incur further significant processing time beyond what is shown for SCALAR_BLOCKS.

III. APPLICATIONS

We will next demonstrate the utility of GOBLOCKS in conformal bootstrap studies with multiple correlators, starting with the 3D Ising model. We will also remark on applying GOBLOCKS to other models in the $O(N)$ vector series.

A. 3D Ising model

The 3D Ising model is one of the most extensively studied CFTs in the conformal bootstrap literature. In [3], the model was analyzed using multiple correlators involving the lowest-lying $\mathbb{Z}_2$ -odd and $\mathbb{Z}_2$ -even scalars, σ and ϵ : $\langle\sigma\sigma\sigma\rangle$, $\langle\sigma\sigma\epsilon\epsilon\rangle$, $\langle\epsilon\epsilon\epsilon\epsilon\rangle$, and $\langle\sigma\epsilon\sigma\epsilon\rangle$. The system of resulting crossing equations is explicitly given by

$$0 = \sum_{\mathcal{O}^+} \lambda_{\sigma\sigma\mathcal{O}}^2 F_{-\Delta,\ell}^{\sigma\sigma,\sigma\sigma}(u, v) \quad (3.1)$$

$$0 = \sum_{\mathcal{O}^+} \lambda_{\epsilon\epsilon\mathcal{O}}^2 F_{-\Delta,\ell}^{\epsilon\epsilon,\epsilon\epsilon}(u, v) \quad (3.2)$$

$$0 = \sum_{\mathcal{O}^-} \lambda_{\sigma\epsilon\mathcal{O}}^2 F_{-\Delta,\ell}^{\sigma\epsilon,\epsilon\sigma}(u, v) \quad (3.3)$$

$$0 = \sum_{\mathcal{O}^+} \lambda_{\sigma\sigma\mathcal{O}} \lambda_{\epsilon\epsilon\mathcal{O}} F_{\mp,\Delta,\ell}^{\sigma\sigma,\epsilon\epsilon}(u, v) \pm \sum_{\mathcal{O}^-} (-1)^{\ell} \lambda_{\sigma\epsilon\mathcal{O}}^2 F_{\mp,\Delta,\ell}^{\epsilon\sigma,\sigma\epsilon}(u, v). \quad (3.4)$$

In the linear-functional approach, all five crossing equations can be combined into a single matrix equation which is analyzed using semidefinite programming

methods. This approach yields highly precise values for the dimensions of the lowest-lying scalars, $\Delta_{\sigma} = 0.51820(14)$ and $\Delta_{\epsilon} = 1.4127(11)$. Achieving high-precision results requires fine control over the accuracy of the conformal blocks, which SCALAR_BLOCKS provides.

In contrast, GOBLOCKS is better suited for situations where such high precision is not essential. This includes the case of truncation methods [9–14]. These methods typically recast the crossing equations as a nonlinear optimization problem, aiming to minimize a nonconvex loss function over a set of variables subject to constraints. In this section we present results obtained by treating the mixed correlator bootstrap of the 3D Ising model in this nonlinear optimization context, using GOBLOCKS for fast online computation of conformal blocks. Our main goal is not to produce new results in the 3D Ising model, but to exhibit the performance of GOBLOCKS in a realistic application.

Truncation methods require the specification of a list of operators that are included in the optimization, called the spin partition. In the reported application, our spin partition consisted of the 18 stable operators with $\Delta \leq 8$ identified in [25]. Table IV lists these operators, grouped by $\mathbb{Z}_2$ parity and spin. In the mixed-correlator bootstrap, each $\mathbb{Z}_2$ -even operator $\mathcal{O}$ appears in the $\sigma \times \sigma$ and $\epsilon \times \epsilon$ OPEs with independent coefficients $\lambda_{\sigma\sigma\mathcal{O}}$ and $\lambda_{\epsilon\epsilon\mathcal{O}}$, contributing three free parameters $(\Delta_{\mathcal{O}}, \lambda_{\sigma\sigma\mathcal{O}}, \lambda_{\epsilon\epsilon\mathcal{O}})$. Each $\mathbb{Z}_2$ -odd operator appears in the $\sigma \times \epsilon$ OPE, contributing two free parameters $(\Delta_{\mathcal{O}}, \lambda_{\sigma\epsilon\mathcal{O}})$ for a total of 47 optimization variables. Note that all scaling dimensions, including those of the external operators σ and ϵ and of the stress tensor $T_{\mu\nu}$, are treated as free parameters in the optimization.

The OPE coefficients for this system of equations can be modeled in several ways. We used the following three combinations: $\lambda_{\sigma\sigma\mathcal{O}}$ and $\lambda_{\epsilon\epsilon\mathcal{O}}$ for the $\mathbb{Z}_2$ -even spectrum, and $\lambda_{\sigma\epsilon\mathcal{O}}$ for the $\mathbb{Z}_2$ -odd spectrum. This approach minimizes the number of optimization variables without introducing extra constraints, while allowing all variables to also take negative values. Other modeling strategies for the OPE coefficients were explored, but performance was similar. The loss function used in the optimization step was defined as the sum of the vector norms of Eqs. (3.1)–(3.4), while the ground-truth values against which we compare the optimization results are taken from [25]. Constraints were also imposed to ensure the scaling dimensions of operators for each spin were ordered. The OPE equality $\lambda_{\sigma\sigma\epsilon} = \lambda_{\sigma\epsilon\sigma}$ was enforced.

TABLE IV. Spin partition used in the search for the truncated 3D Ising spectrum, consisting of the 18 stable operators with $\Delta \leq 8$ from [25]. Each $\mathbb{Z}_2$ -even operator contributes three optimization variables $(\Delta, \lambda_{\sigma\sigma\mathcal{O}}, \lambda_{\epsilon\epsilon\mathcal{O}})$ and each $\mathbb{Z}_2$ -odd operator contributes two, $(\Delta, \lambda_{\sigma\epsilon\mathcal{O}})$, giving 47 optimizable variables in total.

	$\mathbb{Z}_2$ -even Spectrum											$\mathbb{Z}_2$ -odd Spectrum						
	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7
Number																		
Name	ϵ	ϵ'			$T_{\mu\nu}$	$T'_{\mu\nu}$		$C_{\mu\nu\rho\sigma}$				σ	σ'					
Spin	0	0	0	0	2	2	2	4	4	4	6	0	0	2	2	3	4	5
Num Vars	3	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2

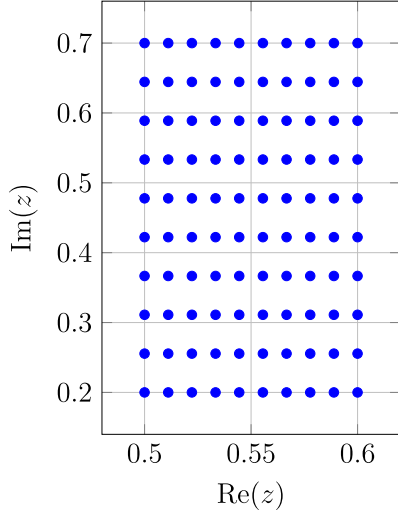


FIG. 4. Sampled points in the complex z plane. This subset of points results in a scalar crossing violation of order 10^{-2} when evaluated on the 18 stable operators of the 3D Ising model with dimensions $\Delta \leq 8$, as reported in [25].

In addition to the GOBLOCKS parameters discussed in Sec. II, other block-related parameters influence the accuracy of the optimization. For the multipoint approach, both the number and placement of sampled points in the complex z plane are important, and naïvely sampling points near the crossing-symmetric point leads to poor performance. Diagnostics using the known spectrum from Table 2 of [25] were used to select an effective set of 100 sampling points, shown in Fig. 4. With this choice, the total crossing violation was of the order of 10^{-2} .

As with the multipoint approach, the normalization of conformal blocks by derivative order (m, n) is crucial for accuracy in the derivative approach. The absolute values of block derivatives can vary widely. Without normalization by a factor appropriate to the derivative order, the total crossing violation becomes large. In the following experiments, derivative terms were normalized by a factor of $4^{m+n} m! n!$, which ensures that each derivative order contributes comparably to the crossing violation. With this normalization, the total crossing violation on the known spectrum was again on the order of 10^{-2} .

Table V presents estimates of the scaling dimensions and OPE coefficients for the lowest-lying scalar operators σ and ϵ using the multipoint approach. The nonconvex optimization was carried out with PyGMO [26] using the interior-point (IPOPT) algorithm, subject to specified bounds for each search variable. Two sets of bounds were used. Narrow bounds restrict the search to within 25% of the scaling dimensions and OPE coefficients of all the operators in the spectrum (12.5% on either side of the true values). Wide bounds allow up to 100% variation from the true values. In a realistic search scenario, such as that of [23], a successful strategy involves starting with very wide

bounds and reducing them sequentially around the mean values of the CFT data with some tolerance.

The table reports the unweighted mean and standard deviation (encoding the stochastic spread) over 100 experiments, as well as the mean and standard deviation weighted by the inverse loss returned by the optimizer. Each experiment took approximately three days and required only modest resources: approximately 8 GB RAM and 6 CPU cores. In practice, many experiments may be parallelized naïvely and run synchronously in an array on an HPC cluster.⁸ To produce OPE coefficients consistent with the conventions in [25], the output of GOBLOCKS had to be multiplied by $4^\Delta (2\nu)_\ell / (\nu)_\ell$, where $\nu := (D - 2)/2$.

With narrow search bounds, the estimates for Δ_σ , $\lambda_{\sigma\sigma\epsilon}$, and $\lambda_{\sigma\epsilon\epsilon}$ are accurate to three decimal places compared to those of [25], while the remaining parameters in Table V are accurate to within one or two decimal places. The standard deviations are small, and the true values fall within these intervals. Weighting the results by the inverse loss further improves the estimates, indicating that the loss function effectively models the problem and is sensitive to the z -point sampling. When the bounds are widened, the estimates become less accurate, but Δ_σ and Δ_ϵ remain within approximately 2% and 5% of their true values, demonstrating that the algorithm can reliably explore a large search space.

Table VI presents estimates of the conformal data using the derivative approach in GOBLOCKS. The results are comparable to—though slightly less accurate than—those from the multipoint approach. With narrow bounds, the estimates for Δ_σ , $\lambda_{\sigma\sigma\epsilon}$, and $\lambda_{\sigma\epsilon\epsilon}$ are accurate to within two decimal places. This experiment employed $n_{\max} = 8$ and $\ell_{\max} = 6$ with a population size of 1,000. When the search bounds were widened, the scaling dimensions remained accurate to within 5% of their true values, using $n_{\max} = 9$ and $\ell_{\max} = 10$. Higher values of these parameters are required for broader search regions. Alternative normalizations of the derivative orders can be used to mitigate bias, but some choices significantly degrade performance. For example, another normalization resulted in $\Delta_\sigma = 0.5147$ and $\Delta_\epsilon = 1.3690$. Notably, the unweighted results outperform the weighted ones, suggesting the loss function, which depends on the chosen normalization, is not yet optimally tuned for this problem.

Similar results have been obtained for the 3D Ising model in the past (see, e.g., [10]) using truncation methods in the derivative approach. Note, however, that the details of the truncation scheme and the input assumptions in [10] were different.

B. Scaling up in the $O(N)$ model series

The multicorrelator numerical bootstrap in the linear-functional approach has been extended beyond the Ising

⁸All the computations reported here were carried out on the Apocrita HPC cluster at Queen Mary University of London [27].

TABLE V. Estimates of the scaling dimensions and OPE coefficients of σ and ϵ in the 3D Ising model using the multipoint approach. Narrow bounds correspond to searching within 25% of the true values for all data, while wide bounds correspond to 100%. Both experiments consisted of 100 runs with a population size of 10,000.

Variable	Truth	Narrow bounds		Wide bounds	
		Weighted	Unweighted	Weighted	Unweighted
Δ_σ	0.5181	0.5180 ± 0.0015	0.5186 ± 0.0035	0.5275 ± 0.0221	0.5332 ± 0.0245
Δ_ϵ	1.4126	1.4105 ± 0.0198	1.4178 ± 0.0455	1.4836 ± 0.3073	1.5572 ± 0.3286
$\lambda_{\sigma\sigma\epsilon}$	1.0519	1.0520 ± 0.0141	1.0483 ± 0.0367	1.0382 ± 0.2015	1.0084 ± 0.2148
$\lambda_{\epsilon\epsilon\epsilon}$	1.5324	1.5244 ± 0.0263	1.5217 ± 0.0617	1.3994 ± 0.2703	1.3728 ± 0.3166
$\lambda_{\sigma\epsilon\sigma}$	1.0519	1.0516 ± 0.0133	1.0451 ± 0.0333	0.9967 ± 0.2300	0.9443 ± 0.2442

TABLE VI. Estimates of the scaling dimensions and OPE coefficients of σ and ϵ in the 3D Ising model using the derivative approach. Both experiments consisted of 100 runs with a population size of 1,000. The narrow bounds used the parameters $n_{\max} = 8$ and $\ell_{\max} = 6$, while wide bounds used $n_{\max} = 9$ and $\ell_{\max} = 10$.

Variable	Truth	Narrow bounds		Wide bounds	
		Weighted	Unweighted	Weighted	Unweighted
Δ_σ	0.5181	0.5173 ± 0.0020	0.5188 ± 0.0041	0.5168 ± 0.0220	0.5287 ± 0.0340
Δ_ϵ	1.4126	1.4006 ± 0.0215	1.4167 ± 0.0437	1.3419 ± 0.2293	1.4556 ± 0.3120
$\lambda_{\sigma\sigma\epsilon}$	1.0519	1.0569 ± 0.0163	1.0522 ± 0.0349	1.1212 ± 0.1302	1.0865 ± 0.1783
$\lambda_{\epsilon\epsilon\epsilon}$	1.5324	1.5131 ± 0.0379	1.5143 ± 0.0605	1.4334 ± 0.2495	1.3789 ± 0.2663
$\lambda_{\sigma\epsilon\sigma}$	1.0519	1.0559 ± 0.0155	1.0452 ± 0.0319	1.0938 ± 0.1653	1.0116 ± 0.2265

model in the $O(N)$ vector model series (see, e.g., [28,29]). Higher values of N introduce a significant increase in compute requirements. For example, the three main searches in the $O(3)$ bootstrap of [29] together required approximately 3 million CPU-hours. It is interesting to ask how things scale up in a truncation scheme. In this section we discuss the computational complexity involved when extending the use of GOBLOCKS to the $O(2)$ and $O(3)$ vector models.

Recall that in the derivative approach to GOBLOCKS, there are three major computational steps: recursion, evaluation, and conversion (see Fig. 6). The recursion step is fast (~ 50 ms for $n_{\max} = 8$) and its runtime is independent of Δ_{ij} and Δ_{kl} . The conversion step dominates for large $n_{\max}$ on the first call, but its cost is amortized through caching. Alternatively, GOBLOCKS supports independent cache building, preventing potentially memory-intensive processes being called at runtime, and thus allowing cluster resources to be tuned without exceeding allocation limits on the first call. The evaluation and conversion steps scale linearly with the number of exchanged operators, but are much less computationally expensive. Conversion to derivatives of F_+ must be computed separately from derivatives of F_- , as they involve different derivative orders (m, n). The total computation time for a single GOBLOCKS call is the sum of these steps. Further optimization is possible across multiple calls, such as during an optimization over different operators and representations. From Eq. (2.1), g depends only on the differences Δ_{ij} and Δ_{kl} , not on the dimensions of the individual external operators. This allows us to reuse the recursion results when these differences

coincide, reducing the total number of recurse steps required. The number of evaluate and convert steps remains unchanged, as they depend on the individual operator dimensions and spins.

The structure of the crossing equations directly determines the number of computational steps per optimizer iteration. For each distinct pair of external operator differences Δ_{ij}, Δ_{kl} where blocks are evaluated, add one recurse and one evaluate step. For every unique combination of Δ_{ij}, Δ_{kl} , and $F_\pm$, add a convert step.

In the Ising model, Eq. (3.1) to Eq. (3.4) yield five unique Δ_{ij}, Δ_{kl} permutations: three from the first three equations and two from the last. Only three recurse steps are needed, since $\langle \sigma\sigma\sigma\sigma \rangle$, $\langle \epsilon\epsilon\epsilon\epsilon \rangle$, and $\langle \sigma\sigma\epsilon\epsilon \rangle$ correspond to $\Delta_{ij}=0=\Delta_{kl}$ and hence the same conformal block g . There are seven unique Δ_{ij}, Δ_{kl} and $F_\pm$ combinations: three from the first three equations and four from the last. The total runtime per optimizer step is $3t_{\text{recurse}} + 5t_{\text{evaluate}} + 7t_{\text{convert}}$. If the recurse step dominates and its runtime is given by Fig. 3, each step takes $3 \times 0.035 = 0.105$ s.

The same estimates can be applied to the $O(2)$ and $O(3)$ models. In the $O(2)$ case, operators fall into irreducible representations (irreps) of $O(2) \cong U(1) \times \mathbb{Z}_2$. Beyond the trivial $\mathbf{0}^+$ and sign $\mathbf{0}^-$ representations, there is an infinite family of two-dimensional irreps labeled by q , spanned by states with $U(1)$ charge $\pm q$ exchanged by $\mathbb{Z}_2$. In general, irreps are labeled $(\ell^\pm, \mathbf{q}^\pm)$, where $\ell^\pm$ denotes spin and spatial parity, and $\mathbf{q}^\pm$ denotes the q irrep and its $\mathbb{Z}_2$ parity. There are 22 crossing equations, as given in Eq. (17) of [28]. The number of computational steps can be directly

TABLE VII. The number of recurse, evaluate, and convert steps needed for the $O(2)$ model by representation. Assuming the total runtime is dominated by the recurse step, the best case total time per optimizer step is ~ 0.5 seconds. Recurse depends only on the differences of the external dimensions Δ_{ij} and Δ_{kl} , meaning some computations can be reused; naïve and best refer to the number of steps both without and with reuse, respectively.

Steps	Representations							Naïve	Best
	$(\ell^+, 0^+)$	$(\ell^-, 0^-)$	$(\ell^\pm, 1)$	$(\ell^+, 2)$	$(\ell^-, 2)$	$(\ell^\pm, 3)$	$(\ell^+, 4)$		
Recurse	6	3	6	4	2	2	1	24	10
Evaluate	6	3	6	4	2	2	1	24	24
Convert	11	6	10	7	3	4	2	43	43

read off from the $\vec{V}$ vectors for each irrep in that equation. Table VII summarizes the steps per irrep in the various crossing channels. In total, there are 24 recurse and evaluate steps, and 43 convert steps. The 24 recurse steps reduce to 10 due to redundancies. Assuming the recurse step dominates, the total runtime is $10 \times 0.035 = 0.35$ s.

For the $O(3)$, or Heisenberg, model, irreducible representations are also labeled by $\mathbf{q}^\pm$. There are 28 crossing equations given in Eq. (14) of [29], with $\vec{V}$ vectors provided in a separate notebook. Table VIII lists the number of processing steps per representation. The minimal number of recurse steps required is 11, only one more than in the $O(2)$ model, yielding a best-case runtime of 0.55 s. The computational complexity of the $O(3)$ model is comparable to that of $O(2)$, particularly when redundancies in g are taken into account.

The following points warrant special mention. First, these estimates do not account for scaling with the number of operators in the truncation. Both the evaluate and convert steps scale with this number and should be included in realistic runtime estimates. In the Ising model, most computation time is spent in the recurse step. Second, this analysis applies only to the derivative approach in GOBLOCKS. In the points approach, there is no convert step, but recurse and evaluate must be performed twice, once for $g(u, v)$ and once for $g(v, u)$, for each z point. Despite this, the points approach is significantly faster than the derivatives approach, so actual runtimes can be much lower, even when accounting for the additional recursions.

IV. OUTLOOK

In this work we introduced GOBLOCKS, a fast, parallel, and flexible conformal block generator supporting both

multipoint and derivative-based bootstrap strategies. The GOBLOCKS package is designed for scenarios where computational speed and moderate accuracy are critical, but ultrahigh precision is not essential. In such contexts, the widely used SCALAR_BLOCKS package can become a computational bottleneck.

The first part of this study benchmarked GOBLOCKS against existing methods in terms of both speed and accuracy. The multipoint approach in GOBLOCKS achieves agreement with SCALAR_BLOCKS at the crossing-symmetric point to within an average percent error as low as 2.5×10^{-8} and agreement with exact analytic expressions along the $\bar{z} = z$ diagonal at median 5×10^{-9} . Overall, for block accuracies below $10^{-4}\%$, and likely even higher, GOBLOCKS is able to attain up to a five-fold speed increase over the SCALAR_BLOCKS benchmark.

In the second part, GOBLOCKS was applied in directly solving for the CFT data of the 3D Ising model with hard truncation, formulated as a nonconvex optimization problem. With narrow parameter bounds around the state-of-the-art numerical solution, the optimizer recovered most low-lying CFT data to within three decimal places. For wider bounds, the optimizer achieved results within 2%–5% of the true values, demonstrating the robustness of the approach. The computational scaling was further analyzed by quantifying the number of steps required per optimizer iteration in the $O(2)$ and $O(3)$ models. The algorithm exhibits favorable scaling, with both models requiring only about 0.5 seconds per step using the derivative approach. The multipoint approach is substantially faster, offering even lower runtimes with similar accuracy.

The multipoint and derivative approaches to conformal-block evaluation in GOBLOCKS offer complementary

TABLE VIII. The number of recurse, evaluate, and convert steps needed for the $O(3)$ model broken out by representation. Assuming the total runtime is dominated by the recurse step, the total time per optimizer step is of the order of 0.55 seconds.

	Representations										
Steps	$(\ell^+, 0^+)$	$(\ell^-, 1^+)$	$(\ell^\pm, 1^-)$	$(\ell^+, 2^+)$	$(\ell^-, 2^+)$	$(\ell^\pm, 2^-)$	$(\ell^-, 3^+)$	$(\ell^\pm, 3^-)$	$(\ell^+, 4^+)$	Naïve	Best
Recurse	6	3	6	7	2	2	1	2	1	30	11
Evaluate	6	3	6	7	2	2	1	2	1	30	30
Convert	11	6	10	12	3	4	2	4	2	54	54

trade-offs. The multipoint approach is five to ten times faster, owing to the absence of a convert step, and in the 3D Ising bootstrap it recovered Δ_σ , Δ_ϵ and the leading OPE coefficients to three decimal places with narrow search bounds, compared to two decimal places for the derivative approach under comparable conditions. However, each method introduces its own source of tuning: the multipoint approach requires a choice of sampling points in the z -plane to ensure rapid OPE convergence [21], whereas the derivative approach is sensitive to the normalization of derivative orders, with suboptimal choices significantly degrading performance.

These tuning choices have a counterpart in interpolation-based methods, where accuracy hinges on how the blocks are sampled. In [30], the polynomial P of Eq. (2.1) is approximated using optimal density nodes for the weight $\mu(\Delta) := r_0^\Delta / Q_{\kappa, \ell}(\Delta)$, which improves accuracy against the true blocks at low to intermediate scaling dimensions. The technique requires only the ability to evaluate P at a chosen set of scaling dimensions. While GOBLOCKS does not expose this quantity directly, it could be modified to return it, enabling an interpolated variant to be benchmarked against SCALAR_BLOCKS. The same work also treats the optimal estimation of functionals acting on the blocks, which arise in the dual formulation of the bootstrap. There, interpolating the analytically known optimal functional of the spin-two gap problem, or dispersive functional, again yields improved estimation. This is most relevant to the dual bootstrap, whereas our applications target the primal bootstrap, where the closest analogs are the choice of sampling points and the normalization of derivative orders. A systematic study of how these parameters affect the recovery of CFT data across models would be a natural next step.

The difficulties discussed above are well-known inherent issues of the hard truncation approach, which we expect to be mitigated in the future by more systematic, flexible and reliable high-dimensional search strategies in the conformal bootstrap, aiming not only to constrain low-lying CFT data, but also to reconstruct full correlation functions. Conformal-block evaluation tools, like SCALAR_BLOCKS and GOBLOCKS, will be useful in such developments. In this context, it would be interesting to explore novel methodologies that combine the speed of GOBLOCKS with the high-precision of SCALAR_BLOCKS (for instance, in potential variants of search algorithms like the Navigator function [31]). It would also be interesting to explore variants of GOBLOCKS itself beyond the four-point scalar bootstrap, e.g., for higher-spin [19] or higher-point [32] conformal blocks.

ACKNOWLEDGMENTS

The authors would like to thank P. Richmond for collaboration at early stages of this work and S. Rychkov for correspondence on the implementation of the recursive algorithm. C. P. was partially supported by the Science and Technology Facilities Council (STFC) Consolidated Grant No. ST/X00063X/1 “Amplitudes,

Strings and Duality.” A. G. S. acknowledges support from Pierre Andurand. This research utilized the Apocrita HPC facility, supported by QMUL Research-IT [27].

DATA AVAILABILITY

Along with this paper, we have made two packages publicly available: (1) GOBLOCKS, available at [24]. (2) BOOTSTOP-MULTI-CORRELATOR: The multiobjective implementation of the bootstrap stochastic optimizer, BOOTSTOP, available at [33].

APPENDIX A: INTERPOLATION SCHEMES FOR CONFORMAL BLOCKS

A fast way of generating block derivatives at the crossing-symmetric point involves pre-computing block values with SCALAR_BLOCKS on a fixed grid of scaling dimensions, followed by multilinear interpolation to estimate values at arbitrary points in the space of scaling dimensions. This method offers high speed and straightforward implementation, though accuracy may degrade in regions where the blocks exhibit significant variation or lack smoothness. One practical realization employs a $k - d$ tree to identify the k nearest neighbors to a target point. The corresponding F -block values, as defined in Eq. (2.5), are then combined using weights inversely proportional to their distances from the query location.

To assess the accuracy of multilinear interpolation, a reference set of F -block values was generated on a new grid of scaling dimensions with 100,000 points. For simplicity, this grid was restricted to spin $\ell = 0$. The interpolation error was computed by comparing the interpolated F values to these reference values. Table IX reports percentiles of the average component error for various choices of the number of neighbors k . The average component error is defined as the mean relative error across all components of the F -block derivatives. At the 25th percentile, errors remain moderate, typically in the range of 14%–30%. At the 90th percentile, errors increase substantially, reaching 200%–300%. The lowest overall errors are observed for $k = 2$ neighbors. Higher errors are expected for larger spin values.

The results show that, although multilinear interpolation is computationally efficient, it does not provide sufficient accuracy for most applications. The precomputed block grid was sampled with a spacing of 0.1 in the scaling dimensions, so the final optimization results are limited to an accuracy of $\delta\Delta = 0.1$. Increasing the grid resolution can improve accuracy, but this requires significantly more memory to store and access the blocks, which constrains the practical resolution achievable with this approach.

Polynomial interpolation is an alternative to multilinear interpolation. In this approach, the k nearest neighbors to a query point x are identified and used to construct a design matrix X_{poly} , where each column represents a polynomial

TABLE IX. Percent errors of block derivatives for multilinear interpolation as a function of the number of neighbors k .

Parameters	Error (%)			
	25%	50%	75%	90%
k				
1	19.32	42.28	91.97	195.26
2	14.34	32.67	71.61	167.86
4	14.22	32.12	75.61	188.50
8	20.28	44.84	105.06	257.85
16	30.39	66.65	154.56	374.81

feature of degree up to n and each row corresponds to a neighbor. The associated block values are collected in the vector y . The interpolation coefficients β are obtained by solving the ridge regression problem

$$(X_{\text{poly}}^T X_{\text{poly}} + \lambda I) \beta = X_{\text{poly}}^T y \quad (\text{A1})$$

for a given regularization parameter λ . The interpolated block value at the query point is then given by $\vec{x}_{\text{poly}} \cdot \beta$, where $\vec{x}_{\text{poly}}$ is the vector of polynomial features evaluated at x .

Table X presents error statistics for the five best-performing parameter choices, ranked by the 90th percentile error. Polynomial interpolation yields lower errors than multilinear interpolation, though at the cost of increased runtime. Despite this improvement, the error levels remain too high for use in precision bootstrap analyses. These limitations motivate the development of block representations that can be evaluated directly at arbitrary query points, enabling both speed and accuracy without relying on interpolation.

A more principled approach to interpolation is developed in [30], where the nodes are placed not on a uniform grid but at the optimal density for the weight $\mu(\Delta) := r_0^\Delta / Q_{\kappa, \ell}(\Delta)$, improving accuracy against the true blocks at low to intermediate scaling dimensions. This targets directly the failure mode seen above, where a fixed grid spacing both caps the achievable resolution and forces errors to grow in regions of rapid block variation. Adapting these node-placement strategies to the block representations used here would be a natural way to recover the speed of interpolation without the accuracy penalties reported above.

TABLE X. Percent errors of block derivatives for polynomial interpolation as a function of the number of neighbors k , degree of the polynomial n , and regularization λ .

Parameters			Error (%)			
k	n	λ	25%	50%	75%	90%
16	2	0.001	9.55	21.38	51.71	136.40
8	2	0.001	11.09	25.12	59.17	144.31
8	3	0.100	10.04	24.00	59.63	150.38
16	3	0.100	9.83	22.12	56.04	151.84
2	2	0.001	15.59	32.89	68.88	155.14

APPENDIX B: ALGORITHMIC FLOW OF GOBLOCKS

The package GOBLOCKS supports both multipoint and derivative approaches. Figure 5 shows the algorithmic flow of the multipoint approach, while Fig. 6 depicts the flow of the derivative approach. In the multipoint approach, the recursion step computes r -series coefficients sequentially (each order depends on the previous ones), but the

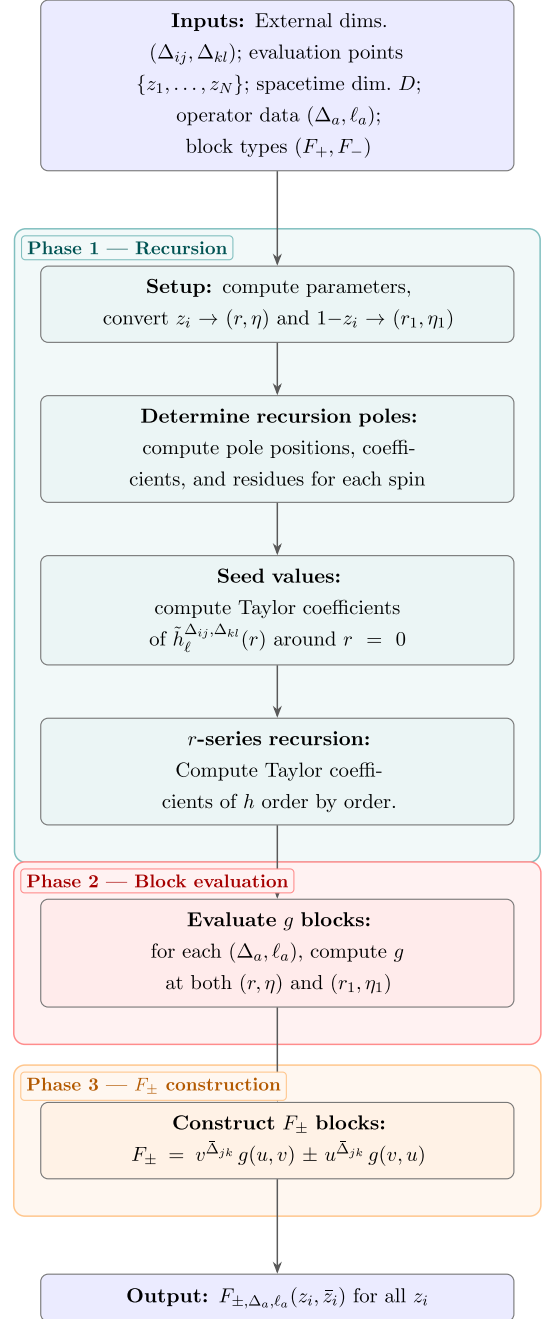


FIG. 5. Schematic overview of multipoint block evaluation steps.

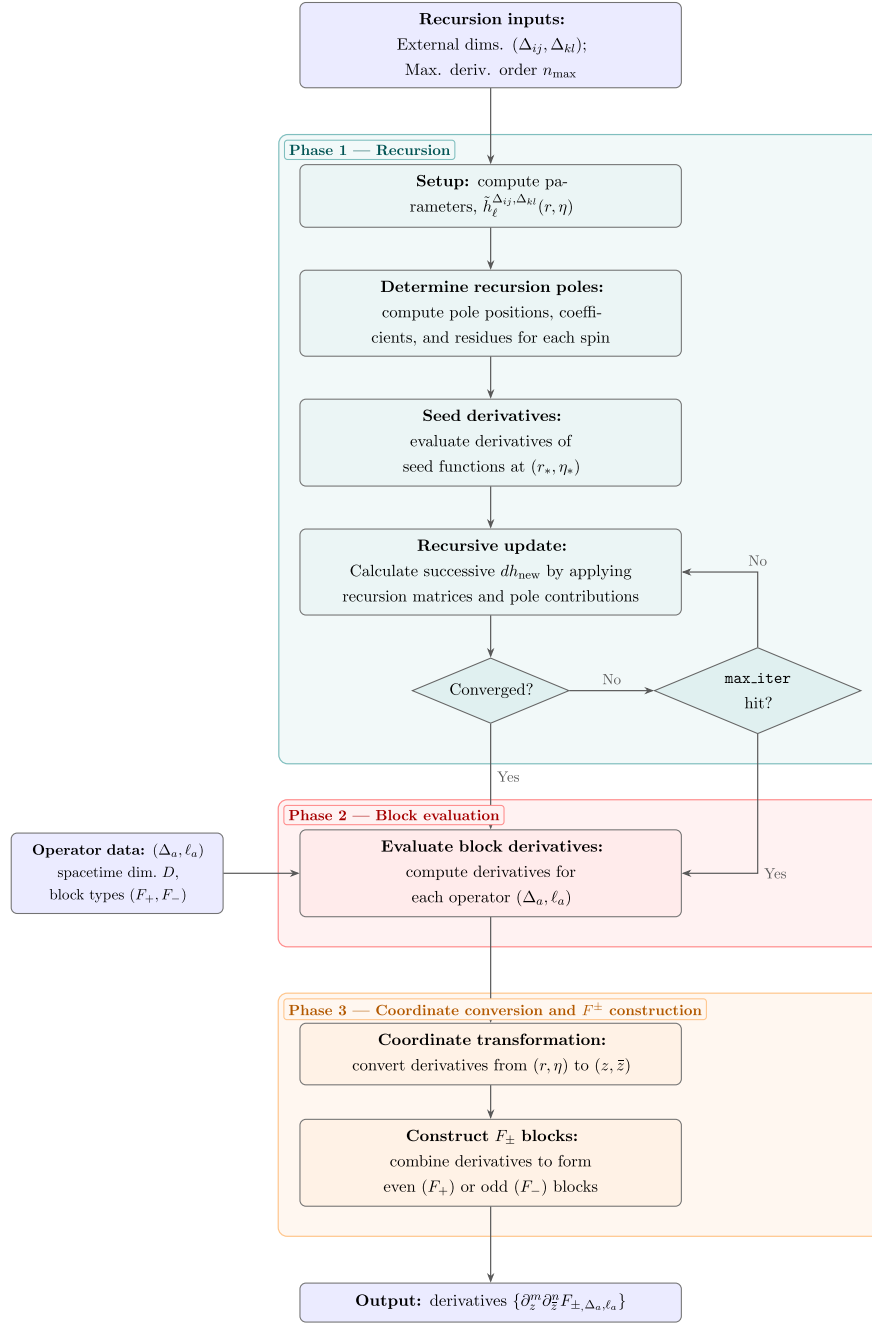


FIG. 6. Schematic overview of block derivative evaluation steps.

evaluation and $F_{\pm}$ construction steps are parallelized across z -points. In the derivative approach, each successive derivative order depends on all lower orders, further limiting parallelization.

APPENDIX C: DERIVATION OF A RECURSIVE RELATION FOR BLOCK DERIVATIVES

We wish to find a recursive relation for derivatives of the conformal blocks $\partial_r^m \partial_\eta^n h_\ell^{\Delta_{ij}, \Delta_{kl}}(r, \eta)$ given the equation

$$h_{\Delta, \ell}^{\Delta_{ij}, \Delta_{kl}}(r, \eta) = \tilde{h}_\ell^{\Delta_{ij}, \Delta_{kl}}(r, \eta) + \sum_i \frac{c_i^{\Delta_{ij}, \Delta_{kl}}}{\Delta - \Delta_i} r^{n_i} h_{\Delta_i + n_i, \ell_i}^{\Delta_{ij}, \Delta_{kl}}(r, \eta), \quad (\text{C1})$$

where as in [3], we define

$$\begin{aligned} \tilde{h}_\ell^{\Delta_{ij}, \Delta_{kl}}(r, \eta) &= \frac{\ell!}{(2\nu)_\ell} \frac{(-1)^\ell C_\ell^\nu(\eta)(1+r^2+2r\eta)^\alpha(1+r^2-2r\eta)^\beta}{(1-r^2)^\nu}, \end{aligned} \quad (\text{C2})$$

with $C_\nu^\nu(\eta)$ denoting the ordinary Gegenbauer polynomial, with $\nu := (D - 2)/2$, and

$$\alpha := -\frac{1}{2}(1 + \Delta_{ij} - \Delta_{kl}), \quad \beta := -\frac{1}{2}(1 - \Delta_{ij} + \Delta_{kl}). \quad (\text{C3})$$

The pole coefficients c_i are defined in a table in [3], so we refer the reader there. The main tool in this calculation will be the Leibniz theorem, which states

$$\partial_r^m \partial_\eta^n f(r, \eta) g(r, \eta) = \sum_{i=0}^m \sum_{j=0}^n \binom{m}{i} \binom{n}{j} \partial_r^i \partial_\eta^j f(r, \eta) \partial_r^{m-i} \partial_\eta^{n-j} g(r, \eta). \quad (\text{C4})$$

Applying Eq. (C4) to the second term in Eq. (C1), we obtain

$$\partial_r^m \partial_\eta^n h_{\Delta, \ell}^{\Delta_{ij}, \Delta_{kl}}(r, \eta) = \partial_r^m \partial_\eta^n \tilde{h}_{\ell}^{\Delta_{ij}, \Delta_{kl}}(r, \eta) + \sum_i \frac{c_i^{\Delta_{ij}, \Delta_{kl}}}{\Delta - \Delta_i} \sum_{s=0}^m \sum_{t=0}^n \binom{m}{s} \binom{n}{t} [\partial_r^{m-s} \partial_\eta^{n-t} (r^{n_i}) \cdot \partial_r^s \partial_\eta^t (h_{\Delta_i + n_i, \ell_i}^{\Delta_{ij}, \Delta_{kl}}(r, \eta))]. \quad (\text{C5})$$

The sum over η collapses since $\partial_r^{m-s} \partial_\eta^{n-t} r^{n_i} = (n_i)_{(m-s)} r^{n_i - m + s} \delta_{n,t}$ where $(a)_{(b)}$ denotes the falling factorial. Thus,

$$\partial_r^m \partial_\eta^n h_{\Delta, \ell}^{\Delta_{ij}, \Delta_{kl}}(r, \eta) = \partial_r^m \partial_\eta^n \tilde{h}_{\ell}^{\Delta_{ij}, \Delta_{kl}}(r, \eta) + \sum_i \frac{c_i^{\Delta_{ij}, \Delta_{kl}}}{\Delta - \Delta_i} \sum_{s=0}^m \binom{m}{s} [(n_i)_{(m-s)} r^{n_i - m + s} \cdot \partial_r^s \partial_\eta^n (h_{\Delta_i + n_i, \ell_i}^{\Delta_{ij}, \Delta_{kl}}(r, \eta))]. \quad (\text{C6})$$

In order to find the full derivative blocks, we first tackle $\partial_r^m \partial_\eta^n \tilde{h}_{\ell}^{\Delta_{ij}, \Delta_{kl}}(r, \eta)$. In order to use Eq. (C4), we let $f(r, \eta) := f_1(r, \eta) f_2^\alpha(r, \eta) f_3^\beta(r, \eta)$, and $g(\eta) := C_\nu^\nu(\eta)$, where

$$f_1 := (1 - r^2)^{-\nu}, \quad f_2 := 1 + r^2 + 2r\eta, \quad f_3 := 1 + r^2 - 2r\eta. \quad (\text{C7})$$

In order to calculate the $\partial_r^m \partial_\eta^n f(r, \eta)$ part, we need to once again apply the Leibniz rule

$$\partial_r^i \partial_\eta^j f(r, \eta) = \sum_{\substack{i_1 + i_2 + i_3 = i \\ j_1 + j_2 + j_3 = j}} \frac{i!}{i_1! i_2! i_3!} \frac{j!}{j_1! j_2! j_3!} (\partial_r^{i_1} \partial_\eta^{j_1} f_1(r)) (\partial_r^{i_2} \partial_\eta^{j_2} f_2^\alpha(r, \eta)) (\partial_r^{i_3} \partial_\eta^{j_3} f_3^\beta(r, \eta)). \quad (\text{C8})$$

Notice

$$\partial_r^i \partial_\eta^j f_1(r) =: \phi_1^{(i,j)}(r) = \begin{cases} \partial_r^i f_1(r) & \text{if } j = 0, \\ 0 & \text{otherwise.} \end{cases} \quad (\text{C9})$$

This equation is independent of the external scaling dimensions and is solely dependent on the spacetime dimension D . Since ν need not be an integer, one must employ the Faà di Bruno formula. Let

$$h(u) = u^{-\nu}, \quad u(r) = 1 - r^2, \quad (\text{C10})$$

such that $\phi_1^{i,0}(r)$ may be written as

$$\frac{\partial^i}{\partial r^i} (1 - r^2)^{-\nu} = \frac{\partial^i}{\partial r^i} h(u(r)). \quad (\text{C11})$$

Using Faà di Bruno's formula,

$$\frac{\partial^i}{\partial r^i} h(u(r)) = \sum_{k=0}^{\lfloor i/2 \rfloor} h^{(i-k)}(u) \frac{i!}{k!(i-2k)!} \left(\frac{u''}{2}\right)^k (u')^{i-2k}. \quad (\text{C12})$$

The derivatives of h are

$$h^{(m)}(u) = (-1)^m (\nu)_m u^{-\nu-m}, \quad (\text{C13})$$

leading to

$$\frac{\partial^i}{\partial r^i} (1 - r^2)^{-\nu} = \sum_{k=0}^{\lfloor i/2 \rfloor} (-1)^{i-k} (\nu)_{i-k} (1 - r^2)^{-\nu-i+k} \frac{i!}{k!(i-2k)!} (-1)^k (-2r)^{i-2k}.$$

The signs simplify to one, yielding

$$\frac{\partial^i}{\partial r^i} (1-r^2)^{-\nu} = i! \sum_{k=0}^{\lfloor i/2 \rfloor} \frac{(\nu)_{i-k}}{k!(i-2k)!} (2r)^{i-2k} (1-r^2)^{-\nu-i+k}. \quad (\text{C14})$$

It should be noted that $\phi_1^{(i,j)}$ rapidly grows very large [$n_{20} \sim \mathcal{O}(10^{17})$].

Turning to the $f_2^\alpha(r, \eta)$ and $f_3^\beta(r, \eta)$ components, we first compute the η derivatives

$$\partial_\eta^j f_2^\alpha(r, \eta) = (\alpha)_{(j)} (2r)^j (f_2)^{\alpha-j} \quad (\text{C15})$$

$$\partial_\eta^j f_3^\beta(r, \eta) = (\beta)_{(j)} (-2r)^j (f_3)^{\beta-j}. \quad (\text{C16})$$

Suppressing the explicit (r, η) dependence on f_2 for notational brevity, consider

$$\begin{aligned} \partial_r^i \partial_\eta^j f_2^\alpha(r, \eta) &= \phi_2^{(i,j)}(r, \eta) = (\alpha)_{(j)} \partial_r^i [(2r)^j f_2^{\alpha-j}] \\ &= (\alpha)_{(j)} 2^j \sum_{k=0}^i \binom{i}{k} [\partial_r^k r^j \cdot \partial_r^{i-k} f_2^{\alpha-j}] \\ &= (\alpha)_{(j)} 2^j \sum_{k=0}^i \binom{i}{k} [(j)_{(k)} r^{j-k} \cdot \partial_r^{i-k} f_2^{\alpha-j}] \end{aligned} \quad (\text{C17})$$

where to obtain the second line from the first we use a special case of Eq. (C4), namely

$$\partial_r^n (f(r)g(r)) = \sum_{k=0}^n \binom{n}{k} \partial_r^k f(r) \partial_r^{n-k} g(r). \quad (\text{C18})$$

The last nontrivial term in Eq. (C17) is $\partial_r^{i-k} f_2^{\alpha-j}$. Since the exponent is in general not an integer, we cannot use the Leibniz rule, and thus must employ the Faà di Bruno formula, which states

$$\partial_r^a g_1(g_2(r)) = \sum_{s=1}^a g_1^{(s)}(g_2(r)) B_{a,s}(g_2^{(1)}(r), \dots, g_2^{(a-s+1)}(r)), \quad (\text{C19})$$

where $B_{a,s}$ is an ordinary Bell polynomial. Letting $g_1(u) := u^{\alpha-j}$ and $g_2 := f_2(r, \eta)$, we may write

$$\begin{aligned} \partial_r^{i-k} f_2^{\alpha-j}(r, \eta) &= \sum_{s=1}^{i-k} (\alpha-j)_{(s)} f_2^{\alpha-j-s}(r, \eta) \\ &\quad \times B_{i-k,s}(f_2^{(1)}(r, \eta), \dots, f_2^{(i-k-s+1)}(r, \eta)). \end{aligned} \quad (\text{C20})$$

This simplifies further since there are only two nonzero r -derivatives of $f_2(r, \eta)$, namely

$$\partial_r^a f_2(r, \eta) = \begin{cases} 2r + 2\eta & \text{if } a = 1, \\ 2 & \text{if } a = 2, \\ 0 & \text{if } a \geq 3. \end{cases} \quad (\text{C21})$$

Using the definition of the Bell polynomial, $\partial_r f_2^{\alpha-j}$ may be written as

$$\partial_r^{i-k} f_2^{\alpha-j} = \sum_{s=1}^{i-k} (\alpha-j)_{(s)} f_2^{\alpha-j-s}(r, \eta) \sum_{\substack{j_1+j_2=s \\ j_1+2j_2=i-k}} \frac{(i-k)!}{j_1! j_2!} (f_2'(r, \eta))^{j_1} \left(\frac{f_2''(r, \eta)}{2!} \right)^{j_2} \quad (\text{C22})$$

$$= \sum_{s=\lceil \frac{i-k}{2} \rceil}^{i-k} (\alpha-j)_{(s)} f_2^{\alpha-j-s}(r, \eta) \frac{(i-k)!}{(2s-i+k)!(i-k-s)!} (2r+2\eta)^{2s-i+k}, \quad (\text{C23})$$

where the second equality is realized by solving the simultaneous equations in the sum index. Hence, one arrives at an expression for $\phi_2^{(i,j)}(r, \eta)$, namely

$$\phi_2^{(i,j)}(r, \eta) = (\alpha)_{(j)} 2^j \sum_{k=0}^i \left[\binom{i}{k} (j)_{(k)} r^{j-k} \sum_{s=\lceil \frac{i-k}{2} \rceil}^{i-k} (\alpha-j)_{(s)} f_2^{\alpha-j-s}(r, \eta) \frac{(i-k)!}{(2s-i+k)!(i-k-s)!} (2r+2\eta)^{2s-i+k} \right]. \quad (\text{C24})$$

Similar reasoning can be applied to $f_3(r, \eta)$, resulting in analogs of Eqs. (C17) and (C23),

$$\partial_r^i \partial_\eta^j f_3^\beta(r, \eta) = \phi_3^{(i,j)}(r, \eta) = (\beta)_{(j)} \partial_r^i [(-2r)^j f_3^{\beta-j}] = (\beta)_{(j)} (-2)^j \sum_{k=0}^i \binom{i}{k} [(j)_{(k)} r^{j-k} \cdot \partial_r^{i-k} f_3^{\beta-j}], \quad (\text{C25})$$

where

$$\partial_r^{i-k} f_3^{\beta-j} = \sum_{s=\lceil \frac{i-k}{2} \rceil}^{i-k} (\beta-j)_{(s)} f_3^{\beta-j-s}(r, \eta) \frac{(i-k)!}{(2s-i+k)!(i-k-s)!} (2r-2\eta)^{2s-i+k}. \quad (C26)$$

Combining the pieces,

$$\phi_3^{(i,j)}(r, \eta) = (-2)^j (\beta)_{(j)} \sum_{k=0}^i \left[\binom{i}{k} (j)_{(k)} r^{j-k} \sum_{s=\lceil \frac{i-k}{2} \rceil}^{i-k} (\beta-j)_{(s)} f_3^{\beta-j-s}(r, \eta) \frac{(i-k)!}{(2s-i+k)!(i-k-s)!} (2r-2\eta)^{2s-i+k} \right]. \quad (C27)$$

We now turn to the Gegenbauer polynomial $g(r)$. Notice that the Gegenbauer polynomial is independent of r , meaning we only need to consider the $m-i=0$ contribution. Using the identity

$$\partial_\eta C_n^\lambda(\eta) = 2\lambda C_{n-1}^{\lambda+1}(\eta), \quad (C28)$$

we find by inspection

$$\begin{aligned} \partial_\eta^{n-j} g(r, \eta) &= \partial_\eta^{n-j} C_\ell^\nu(\eta) \\ &= 2^{n-j} (\nu)_{n-j} C_{\ell-n+j}^{\nu+n-j}, \end{aligned} \quad (C29)$$

where to denote the rising factorial we have used $(\nu)_j := \nu(\nu+1)\cdots(\nu+j-1)$ as in Sec. II B.

Putting everything together,

$$\begin{aligned} \partial_r^m \partial_\eta^n \tilde{h}_\ell^{\Delta_{ij}, \Delta_{kl}} &= \frac{(-1)^\ell \ell!}{(2\nu)_\ell} \partial_r^m \partial_\eta^n (f(r, \eta) g(r, \eta)) \\ &= \frac{(-1)^\ell \ell!}{(2\nu)_\ell} \sum_{j=0}^n \binom{n}{j} \left[\sum_{\substack{i_1+i_2+i_3=m \\ j_1+j_2+j_3=j}} \frac{m!}{i_1!i_2!i_3!} \frac{j!}{j_1!j_2!j_3!} \phi_1^{(i_1, j_1)} \phi_2^{(i_2, j_2)} \phi_3^{(i_3, j_3)} \right] [2^{n-j} (\nu)_{n-j} C_{\ell-n+j}^{\nu+n-j}]. \end{aligned} \quad (C30)$$

In GOBLOCKS, Eq. (C6) is implemented with $\partial_r^m \partial_\eta^n \tilde{h}_\ell^{\Delta_{ij}, \Delta_{kl}}$ defined in Eq. (C30).

APPENDIX D: DERIVATIVES OF r, η WITH RESPECT TO $z, \bar{z}$

GOBLOCKS computes derivatives with respect to the radial coordinates r, η . In order to use the results in the nonconvex optimization problem arising from the primal formulation (or compare with SCALAR_BLOCKS), one must transform derivatives to $z, \bar{z}$ coordinates. In this Appendix we derive the necessary transformation formulas.

Define

$$\phi(z) := \frac{z}{(1 + \sqrt{1-z})^2} \quad (D1)$$

and denote the radial coordinates as

$$\eta(z, \bar{z}) := \frac{\phi(z) + \phi(\bar{z})}{2r(z, \bar{z})}, \quad r(z, \bar{z}) := \sqrt{\phi(z)\phi(\bar{z})}. \quad (D2)$$

In order to simplify calculations, define

$$\phi_1(z, \bar{z}) := \frac{\sqrt{z}(1 + \sqrt{1-\bar{z}})}{\sqrt{\bar{z}}(1 + \sqrt{1-z})}, \quad (D3)$$

$$\phi_2(z, \bar{z}) := \frac{\sqrt{z\bar{z}}}{(1 + \sqrt{1-z})(1 + \sqrt{1-\bar{z}})}, \quad (D4)$$

$$\phi_3(z, \bar{z}) := \sqrt{\bar{z}}(1 + \sqrt{1-\bar{z}}), \quad (D5)$$

allowing one to write

$$\eta(z, \bar{z}) = \frac{1}{2}(\phi_1(z, \bar{z}) + \phi_1(\bar{z}, z)), \quad r(z, \bar{z}) = \phi_2(z, \bar{z}), \quad (D6)$$

and

$$\phi_1(z, \bar{z}) = \frac{\phi_3(z, \bar{z})}{\phi_3(\bar{z}, z)}. \quad (D7)$$

Our aim is to calculate $\partial_m \bar{\partial}_n r(z, \bar{z})$, $\partial_m \bar{\partial}_n \eta(z, \bar{z})$, where $\partial_m := \frac{\partial}{\partial z^m}$, and $\bar{\partial}_n := \frac{\partial}{\partial \bar{z}^n}$. Focus first on the constituents of the η derivatives

$$\bar{\partial}_n \phi_3 = \sqrt{\bar{z}} [\delta_{n,0} + \bar{\partial}_n \sqrt{1-\bar{z}}]. \quad (D8)$$

Noticing that

$$\bar{\partial}_n (\sqrt{1-\bar{z}}) = (-1)^n (1/2)_{(n)} (1-\bar{z})^{\frac{1}{2}-n}, \quad (D9)$$

the derivative becomes

$$\bar{\partial}_n \phi_3(z, \bar{z}) = \sqrt{z}(\delta_{n,0} + (-1)^n (1/2)_{(n)} (1 - \bar{z})^{\frac{1}{2}-n}). \quad (\text{D10})$$

Taking the derivative with respect to z , this becomes

$$\partial_m \bar{\partial}_n \phi_3(z, \bar{z}) = (1/2)_{(m)} z^{\frac{1}{2}-m} (\delta_{n,0} + (-1)^n (1/2)_{(n)} (1 - \bar{z})^{\frac{1}{2}-n}). \quad (\text{D11})$$

In order to obtain the full $\eta(z, \bar{z})$ derivative, one must consider the derivative $\partial_m \bar{\partial}_n (\frac{\phi_3(z, \bar{z})}{\phi_3(\bar{z}, z)})$, where it is convenient to utilize the Leibniz rule, namely

$$\begin{aligned} \partial_m \bar{\partial}_n [f(z, \bar{z})g(z, \bar{z})] \\ = \sum_{i=0}^m \sum_{j=0}^n \binom{m}{i} \binom{n}{j} \partial_i \bar{\partial}_j f(z, \bar{z}) \partial_{m-i} \bar{\partial}_{n-j} g(z, \bar{z}). \end{aligned} \quad (\text{D12})$$

In this case, one can identify $f(z, \bar{z}) := \phi_3(z, \bar{z})$ and $g(z, \bar{z}) := \phi_3^{-1}(\bar{z}, z)$. We turn our attention now to $\partial_{m-i} \bar{\partial}_{n-j} g(z, \bar{z})$. In order to calculate

$$\partial_a \bar{\partial}_b \phi_3^{-1}(\bar{z}, z), \quad (\text{D13})$$

consider simply the $\bar{z}$ derivatives first

$$\tilde{\psi}_0(z) := (1 + \sqrt{1-z})^{-1},$$

$$\tilde{\psi}_{a \geq 1}(z) := \sum_{s=1}^a (-1)^s s! (1 + g_2(z))^{-(1+s)} B_{a,s}(g_2^{(1)}(z), \dots, g_2^{(a-s+1)}(z)), \quad (\text{D18})$$

and $B_{a,s}(x)$ are generalized exponential Bell polynomials.

Combining results,

$$\partial_m \bar{\partial}_n \phi_1(z, \bar{z}) = \sum_{i=0}^m \sum_{j=0}^n \binom{m}{i} \binom{n}{j} (1/2)_{(i)} z^{\frac{1}{2}-i} (\delta_{j,0} + (-1)^j (1/2)_{(j)} (1 - \bar{z})^{\frac{1}{2}-j}) \psi_{m-i, n-j}(\bar{z}, z) \quad (\text{D19})$$

and one can realise a formula for the tower of η derivatives by noting that

$$\partial_m \bar{\partial}_n \eta(z, \bar{z}) = \frac{1}{2} (\partial_m \bar{\partial}_n \phi_1(z, \bar{z}) + \partial_m \bar{\partial}_n \phi_1(\bar{z}, z)). \quad (\text{D20})$$

We can now turn to the r derivatives. One begins by writing

$$r(z, \bar{z}) = f_1(z, \bar{z}) f_2(z) f_3(\bar{z}), \quad (\text{D21})$$

$$\bar{\partial}_b \phi_3^{-1}(\bar{z}, z) = (1 + \sqrt{1-z})^{-1} \bar{\partial}_b \bar{z}^{-\frac{1}{2}}$$

$$= (-1/2)_{(b)} (1 + \sqrt{1-z})^{-1} \bar{z}^{-(\frac{1}{2}+b)}, \quad (\text{D14})$$

before taking z derivatives,

$$(-1/2)_{(b)} \partial_a [(1 + \sqrt{1-z})^{-1}] \bar{z}^{-(\frac{1}{2}+b)}. \quad (\text{D15})$$

The last derivative $\partial_a [(1 + \sqrt{1-z})^{-1}]$ can be simplified by employing the Faà di Bruno formula Eq. (C19). Let $g_1(u) := (1+u)^{-1}$ and $g_2(z) := \sqrt{1-z}$. The derivatives of $g_1(u)$ and $g_2(z)$ are

$$\begin{aligned} g_1^{(s)}(u) &= (-1)^s s! (1+u)^{-(1+s)}, \\ g_2^{(s)}(z) &= (-1)^s (1/2)_{(s)} (1-z)^{\frac{1}{2}-s}. \end{aligned} \quad (\text{D16})$$

The final formula is thus

$$\psi_{a,b}(\bar{z}, z) := \partial_a \bar{\partial}_b \phi_3^{-1}(\bar{z}, z) = \tilde{\psi}_a(z) [\bar{z}^{-(\frac{1}{2}+b)} (-1/2)_{(b)}], \quad (\text{D17})$$

where $\tilde{\psi}_a(z)$ is defined such that

where

$$f_1(z, \bar{z}) := (z\bar{z})^{\frac{1}{2}} \quad (\text{D22})$$

$$f_2(z) := (1 + \sqrt{1-z})^{-1} \quad (\text{D23})$$

$$f_3(\bar{z}) := (1 + \sqrt{1-\bar{z}})^{-1}. \quad (\text{D24})$$

Using the Leibniz formula once more,

$$\partial_m \bar{\partial}_n r(z, \bar{z}) = \sum_{\substack{m_1+m_2+m_3=m \\ n_1+n_2+n_3=n}} \frac{m!}{m_1!m_2!m_3!} \frac{n!}{n_1!n_2!n_3!} (\partial_{m_1} \bar{\partial}_{n_1} f_1(z, \bar{z})) (\partial_{m_2} \bar{\partial}_{n_2} f_2(z)) (\partial_{m_3} \bar{\partial}_{n_3} f_3(\bar{z})). \quad (\text{D25})$$

Consider first $\partial_{m_1} \bar{\partial}_{n_1} f_1(z, \bar{z})$

$$\begin{aligned} \partial_{m_1} \bar{\partial}_{n_1} f_1(z, \bar{z}) &= \partial_{m_1} \bar{\partial}_{n_1} (z\bar{z})^{\frac{1}{2}} \\ &= \sum_{m=0}^{m_1} \sum_{n=0}^{n_1} \binom{m_1}{m} \binom{n_1}{n} \partial_m \bar{\partial}_n (z^{\frac{1}{2}}) \partial_{m_1-m} \bar{\partial}_{n_1-n} (\bar{z}^{\frac{1}{2}}). \end{aligned} \quad (\text{D26})$$

The innermost double derivatives of the $z, \bar{z}$ terms may be realized as

$$\partial_m \bar{\partial}_n (z^{\frac{1}{2}}) = \delta_{n,0} (1/2)_{(m)} z^{\frac{1}{2}-m} \quad (\text{D27})$$

$$\partial_{m_1-m} \bar{\partial}_{n_1-n} (\bar{z}^{\frac{1}{2}}) = \delta_{m_1-m,0} (1/2)_{(n_1-n)} \bar{z}^{\frac{1}{2}-n_1+n}. \quad (\text{D28})$$

Combining with Eq. (D26),

$$\partial_{m_1} \bar{\partial}_{n_1} (z\bar{z})^{\frac{1}{2}} = (1/2)_{(m_1)} (1/2)_{(n_1)} z^{\frac{1}{2}-m_1} \bar{z}^{\frac{1}{2}-n_1}. \quad (\text{D29})$$

Then, focusing our attention on the $f_2(z)$ and $f_3(\bar{z})$ derivatives, we notice that

$$\partial_{m_2} \bar{\partial}_{n_2} f_2(z) = \delta_{n_2,0} \partial_{m_2} [(1 + \sqrt{1-z})^{-1}] = \delta_{n_2,0} \tilde{\psi}_{m_2}(z) \quad (\text{D30})$$

and

$$\partial_{m_3} \bar{\partial}_{n_3} f_3(\bar{z}) = \delta_{m_3,0} \partial_{n_3} [(1 + \sqrt{1-\bar{z}})^{-1}] = \delta_{m_3,0} \tilde{\psi}_{n_3}(\bar{z}). \quad (\text{D31})$$

The final formula for the r derivatives is then

$$\partial_m \bar{\partial}_n r(z, \bar{z}) = \sum_{\substack{m_1+m_2=m \\ n_1+n_2=n}} \frac{m!n!}{m_1!m_2!n_1!n_2!} (1/2)_{(m_1)} (1/2)_{(n_1)} z^{\frac{1}{2}-m_1} \bar{z}^{\frac{1}{2}-n_1} \tilde{\psi}_{m_2}(z) \tilde{\psi}_{n_2}(\bar{z}). \quad (\text{D32})$$

APPENDIX E: TUNING GOBLOCKS FOR OPTIMAL PERFORMANCE

As discussed in Sec. II D, the performance of GOBLOCKS is predominantly controlled by $k_{1,2}^{\max}$ and $\ell_{\max}$. In the absence of a canonical strategy for configuring these parameters, it is useful to tune them in order to minimize the runtime for a desired level of block accuracy. This is particularly relevant when calling GOBLOCKS many times, as, e.g., in the process of solving the crossing equations via stochastic optimization. To some tolerance, there exists an equivalence class of combinations of $k_{1,2}^{\max}, \ell_{\max}$ resulting in a given level of performance. A well-motivated technique to

find the optimal parameter combination is to perform a Bayesian sweep in the space of configurations. Alternatively, one may adopt a naïve approach and set each of the aforementioned parameter to a common value. Figure 7 presents block accuracies for various parameters compared to those evaluated with SCALAR_BLOCKS at varying orders, for common values of $k_{1,2}^{\max}$ and $\ell_{\max}$. We stress that while the hyperparameter combinations depicted in Fig. 7 are likely suboptimal, for a band of accuracies, there exist parameter combinations where GOBLOCKS is more computationally efficient than SCALAR_BLOCKS. Despite this, for large $n_{\max}$ and high block accuracies, SCALAR_BLOCKS offers better performance.

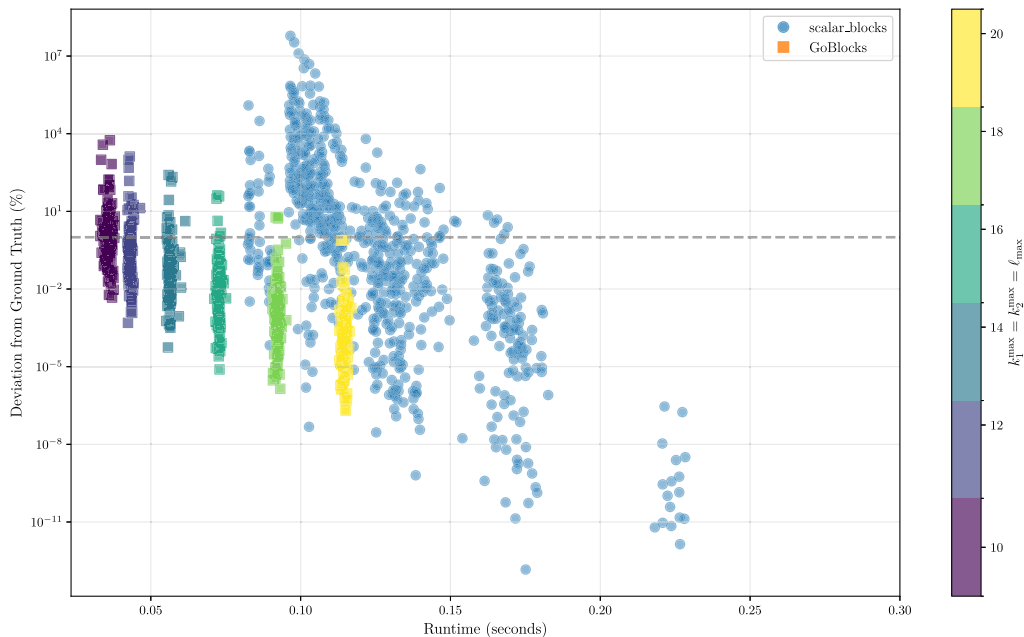


FIG. 7. Runtime versus accuracy for SCALAR_BLOCKS and GOBLOCKS with a selection of hyperparameters, with the 1% accuracy level depicted by a dashed horizontal line.

-
- [1] R. Rattazzi, V. S. Rychkov, E. Tonni, and A. Vichi, Bounding scalar operator dimensions in 4D CFT, *J. High Energy Phys.* **12** (2008) 031; D. Poland, S. Rychkov, and A. Vichi, The conformal bootstrap: Theory, numerical techniques, and applications, *Rev. Mod. Phys.* **91**, 015002 (2019); S. Rychkov and N. Su, New developments in the numerical conformal bootstrap, *Rev. Mod. Phys.* **96**, 045004 (2024).
 - [2] 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**, 025022 (2012).
 - [3] F. Kos, D. Poland, and D. Simmons-Duffin, Bootstrapping mixed correlators in the 3D Ising model, *J. High Energy Phys.* **11** (2014) 109.
 - [4] F. Kos, D. Poland, D. Simmons-Duffin, and A. Vichi, Precision islands in the Ising and $O(N)$ models, *J. High Energy Phys.* **08** (2016) 036.
 - [5] M. Billò, V. Gonçalves, E. Lauria, and M. Meineri, Defects in conformal field theory, *J. High Energy Phys.* **04** (2016) 091; E. Lauria, M. Meineri, and E. Trevisani, Radial coordinates for defect CFTs, *J. High Energy Phys.* **11** (2018) 148.
 - [6] D. M. McAvity and H. Osborn, Conformal field theories near a boundary in general dimensions, *Nucl. Phys.* **B455**, 522 (1995); P. Liendo, L. Rastelli, and B. C. van Rees, The bootstrap program for boundary CFT_d, *J. High Energy Phys.* **07** (2013) 113.
 - [7] L. Iliesiu, M. Koloğlu, R. Mahajan, E. Perlmutter, and D. Simmons-Duffin, The conformal bootstrap at finite temperature, *J. High Energy Phys.* **10** (2018) 070.
 - [8] D. Poland, V. Prilepina, and P. Tadić, The five-point bootstrap, *J. High Energy Phys.* **10** (2023) 153; D. Poland, V. Prilepina, and P. Tadić, Improving the five-point bootstrap, *J. High Energy Phys.* **05** (2024) 299; D. Poland, V. Prilepina, and P. Tadić, Mixed five-point correlators in the 3d Ising model, *J. High Energy Phys.* **10** (2025) 237.
 - [9] F. Gliozzi, Constraints on conformal field theories in diverse dimensions from the bootstrap mechanism, *Phys. Rev. Lett.* **111**, 161602 (2013).
 - [10] F. Gliozzi and A. Rago, Critical exponents of the 3d Ising and related models from Conformal Bootstrap, *J. High Energy Phys.* **10** (2014) 042.
 - [11] F. Gliozzi, P. Liendo, M. Meineri, and A. Rago, Boundary and interface CFTs from the conformal bootstrap, *J. High Energy Phys.* **05** (2015) 036; F. Gliozzi, Truncatable bootstrap equations in algebraic form and critical surface exponents, *J. High Energy Phys.* **10** (2016) 037; W. Li, New method for the conformal bootstrap with OPE truncations, *arXiv:1711.09075*.
 - [12] G. Kántor, V. Niarchos, and C. Papageorgakis, Solving conformal field theories with artificial intelligence, *Phys. Rev. Lett.* **128**, 041601 (2022).
 - [13] G. Kántor, V. Niarchos, and C. Papageorgakis, Conformal bootstrap with reinforcement learning, *Phys. Rev. D* **105**, 025018 (2022); G. Kántor, V. Niarchos, C. Papageorgakis,

- and P. Richmond, 6D (2,0) bootstrap with the soft-actor-critic algorithm, *Phys. Rev. D* **107**, 025005 (2023).
- [14] W. Li, Easy bootstrap for the 3D Ising model: A hybrid approach of the lightcone bootstrap and error minimization methods, *J. High Energy Phys.* **07** (2024) 047.
- [15] R. Hu and W. Li, Accurate boundary bootstrap for the three-dimensional $O(N)$ normal universality class, *Phys. Rev. Lett.* **136**, 221601 (2026); Y.-t. Huang, S.-C. Lee, H. Liao, and J. Rumbutis, Bootstrapping non-unitary CFTs, [arXiv:2512.07706](https://arxiv.org/abs/2512.07706).
- [16] V. Niarchos, C. Papageorgakis, A. Stratoudakis, and M. Woolley, Deep finite temperature bootstrap, *Phys. Rev. D* **112**, 126012 (2025).
- [17] D. Simmons-Duffin, A semidefinite program solver for the conformal bootstrap, *J. High Energy Phys.* **06** (2015) 174.
- [18] The Bootstrap Collaboration, SCALAR_BLOCKS: Conformal Blocks for Scalar Correlators, 2023, https://gitlab.com/boostrapcollaboration/scalar_blocks.
- [19] R. S. Erramilli, L. V. Iliesiu, P. Kravchuk, W. Landry, D. Poland, and D. Simmons-Duffin, BLOCKS_3D: Software for general 3D conformal blocks, *J. High Energy Phys.* **11** (2021) 006.
- [20] M. Hogervorst and S. Rychkov, Radial coordinates for conformal blocks, *Phys. Rev. D* **87**, 106004 (2013).
- [21] A. Castedo Echeverri, B. von Harling, and M. Serone, The effective bootstrap, *J. High Energy Phys.* **09** (2016) 097.
- [22] A. Cavaglià, N. Gromov, J. Julius, and M. Preti, Integrability and conformal bootstrap: One dimensional defect conformal field theory, *Phys. Rev. D* **105**, L021902 (2022); A. Cavaglià, N. Gromov, J. Julius, and M. Preti, Bootstrability in defect CFT: Integrated correlators and sharper bounds, *J. High Energy Phys.* **05** (2022) 164.
- [23] V. Niarchos, C. Papageorgakis, P. Richmond, A. G. Stapleton, and M. Woolley, Bootstrability in line-defect CFTs with improved truncation methods, *Phys. Rev. D* **108**, 105027 (2023).
- [24] J. Chryssanthacopoulos, V. Niarchos, C. Papageorgakis, and A. G. Stapleton, GOBLOCKS, 2026, github.com/xand-stapleton/goblocks.
- [25] D. Simmons-Duffin, The lightcone bootstrap and the spectrum of the 3D Ising CFT, *J. High Energy Phys.* **03** (2017) 086.
- [26] F. Biscani and D. Izzo, A parallel global multiobjective framework for optimization: Pagmo, *J. Open Source Software* **5**, 2338 (2020).
- [27] T. King, S. Butcher, and L. Zalewski, Apocrita—High Performance Computing Cluster for Queen Mary University of London, 2017, [10.5281/zenodo.438045](https://zenodo.org/record/438045).
- [28] S. M. Chester, W. Landry, J. Liu, D. Poland, D. Simmons-Duffin, N. Su, and A. Vichi, Carving out OPE space and precise $O(2)$ model critical exponents, *J. High Energy Phys.* **06** (2020) 142.
- [29] S. M. Chester, W. Landry, J. Liu, D. Poland, D. Simmons-Duffin, N. Su, and A. Vichi, Bootstrapping Heisenberg magnets and their cubic instability, *Phys. Rev. D* **104**, 105013 (2021).
- [30] C.-H. Chang, V. Dommes, P. Kravchuk, D. Poland, and D. Simmons-Duffin, Accurate bootstrap bounds from optimal interpolation, *J. High Energy Phys.* **05** (2026) 275.
- [31] M. Reehorst, S. Rychkov, D. Simmons-Duffin, B. Sirois, N. Su, and B. van Rees, Navigator function for the conformal bootstrap, *SciPost Phys.* **11**, 072 (2021).
- [32] V. Gonçalves, R. Pereira, and X. Zhou, 20' five-point function from $AdS_5 \times S^5$ supergravity, *J. High Energy Phys.* **10** (2019) 247; D. Poland and V. Prilepina, Recursion relations for 5-point conformal blocks, *J. High Energy Phys.* **10** (2021) 160.
- [33] J. Chryssanthacopoulos, V. Niarchos, C. Papageorgakis, and A. G. Stapleton, BOOTSTOP-MULTI-CORRELATOR, 2026, github.com/jchryssanthacopoulos/BootSTOP-multi-correlator.