FuzzifiED

Toolkit to close the gap for fuzzy sphere numerics

---

1 March, 2025

Abstract : Since its proposal, the fuzzy sphere regularisation has made significant contribution to the study of 3d CFTs. The Julia package FuzzifiED is aimed at simplifying the numerical calculations on the fuzzy sphere. It facilitates the exact diagonalisation (ED) calculations as well as the density matrix renormalisation group (DMRG) with the help of ITensor. It can also be used for generic fermionic and bosonic models. This documentation gives a general introduction to the fuzzy sphere regularisation and a detailed instruction for using the package for numerical calculations.

Documentation : https://docs.fuzzified.world
Source code : https://github.com/FuzzifiED/FuzzifiED.jl

Contents

1 Purpose and outline

Since its proposal, the fuzzy sphere regularisation has made significant contribution to the study of 3d CFTs [1–17]. The Julia package FuzzifiED is aimed at simplifying the numerical calculations on the fuzzy sphere. It facilitates the exact diagonalisation (ED) calculations as well as the density matrix renormalisation group (DMRG) with the help of ITensor [18]. It can also be used for generic fermionic and bosonic models. This package features the following characteristics :

1. Versatality : FuzzifiED can help reproduce almost all the ED and DMRG results in fuzzy sphere works, and it is easy and flexible for the adaption to new models.
2. Usability : Julia interfaces make the code intuitive and short. To help the users get started, we have also provided a collection of examples.
3. Efficiency : FuzzifiED can produce results on reasonable system sizes within minutes.
4. Open source : The code for FuzzifiED is fully open source.

This documentation gives a general introduction to the fuzzy sphere regularisation and a detailed instruction for numerical calculations with FuzzifiED. The rest of this documentation is organised as the follows : The Part I (Sections 2–5) is devoted to an introduction to the fuzzy sphere regularisation and the numerical methods applied to it ; The Part II (Sections 6–9) gives a detailed instruction for numerical calculation with the package FuzzifiED.

- In the rest of Section 2, we make a brief introduction to the conformal field theories in dimensions $d \geq 3$ and the fuzzy sphere regularisation.
- In Section 3, we review the existing works related to the fuzzy sphere.
- In Section 4, we review the setup of fuzzy sphere, the construction of interaction models and the extraction of CFT data.
- In Section 5, we review the numerical methods used in the package FuzzifiED, *viz.* ED and DMRG.
- In Section 6, we give an instruction for installing and getting started with FuzzifiED.
- In Section 7, we give an instruction on performing ED calculation with FuzzifiED.
- In Section 8, we give an instruction on performing DMRG calculation with FuzzifiED.
- In Section 9, we present a collection of practical examples that reproduce many existing

results.

Part I

Introduction to fuzzy sphere

2 Introduction

2.1 Conformal field theory

Conformal field theory (CFT) is one of the central topics of modern physics. It refers to a field theory that is invariant under conformal transformations that preserve the angles between vectors. In spacetime dimension $d > 2$, the global conformal symmetry is generated by translation, SO(d) rotation (In this note we work in Euclidean signature. In Lorentzian signature it is the Lorentz transformation SO$(1, d-1)$), dilatation (scale transformation), and special conformal transformation (SCT) [19, 20]. These transformations altogether generate the conformal group SO$(d+1, 1)$. Each CFT operator must transform under irreducible representations of rotation and dilatation. The representations are labelled by SO(d) spin l and scaling dimension Δ, respectively. A special kind of operators that are invariant under SCT called « primaries » deserve particular attension. By acting the space derivatives on the primaries, their « descendants » are obtained. The conformal symmetry is the maximal spacetime symmetry (except supersymmetry) that a field theory can have. It gives powerful constraint on the property of the field theory. In particular, conformal symmetry uniquely determines the form of two-point and three-point correlation functions. The three-point correlator of three primary operators Φ_i, Φ_j, Φ_k contains a universal coefficient called the OPE coefficient $f_{\Phi_i\Phi_j\Phi_k}$. The collection of scaling dimensions and the OPE coefficients $\{\Delta_{\Phi_i}, f_{\Phi_i\Phi_j\Phi_k}\}$ is called the conformal data. Theoretically, with full knowledge of the CFT data, an arbitrary correlation function of a CFT can be obtained.

CFT has provided important insights into various fields of theoretical physics. In condensed matter physics, it has produced useful prediction about the critical phenomena [21, 22]. Many classical and quantum phase transitions are conjectured to have emergent conformal symmetry in the IR. The universal critical exponents are directly determined by the scaling dimensions of the primary operators. *E.g.*, in Ising transitions that spontaneously break $\mathbb{Z}_2$ symmetry,

most critical exponents are given by the scaling dimensions of the lowest $\mathbb{Z}_2$-odd operator σ and $\mathbb{Z}_2$-even operator ϵ, such as

$$\eta = 2\Delta_\sigma - 1 \qquad \nu = \frac{1}{3 - \Delta_\epsilon}. \tag{2.1}$$

CFT is also closely related to string theory and quantum gravity in high energy physics. In the string theory, CFT describes the 2d worldsheet ; in quantum gravity, there is a conjectured duality between the gravity theory in $(d+1)$-dimensional anti-de Sitter (AdS) space in the bulk and a d-dimensional CFT on the boundary [23]. Moreover, CFT plays an important role in our understanding of quantum field theories. It describes many fixed points in the RG flow and many QFTs can be seen as a CFT with perturbations. It also helps us understand how physics change under a change of scale and reveals some fundamental structure of the RG flow [24].

In 2d CFTs, besides the global conformal symmetry SO(3, 1), there also exists an infinite dimensional local conformal symmetry [25–27]. Altogether, they form the Virasoro algebra. The infinite dimensional conformal algebra has made many theories exactly solvable, especially the rational theories such as the minimal models and more generally the Wess-Zumino-Witten (WZW) theories. On the other hand, going to the higher dimensions, the CFTs are much less well-studied due to a much smaller conformal group. The existing methods include numerical conformal bootstrap and Monte Carlo lattice simulations. Numerical boostrap bounds the conformal data by making use of consistency conditions such as reflection positivity together with some information of the CFT such as the global symmetry and a certain amount of assumptions [28, 29]. It has achieved great success in 3d Ising, O(N) Wilson-Fisher, Gross-Neveu CFTs, *etc*. On the other hand, one can study a CFT by constructing a lattice model that goes through a phase transition in the corresponding universality class, and study the phase transition by Monte Carlo simulation. The extraction of universal data usually involves complicated and expensive finite-size scaling, and only the lowest few CFT operators can be accessed in this way.

Among these higher dimensional CFTs, we especially focus on $d = 3$, as many Lagrangians in $d \geq 4$ flow to free theories.

2.2 Fuzzy sphere

In addition to these existing approaches, the *« fuzzy sphere regularisation »* has recently emerged as a new powerful method to study 3d CFTs. The idea is to put an interacting quantum Hamiltonian on a 2-sphere S^2. This geometry preserves the full rotation symmetry (on the contrary, lattice models often only preserve a discrete subgroup). Moreover, when the system is tuned to a critical point or critical phase, combined with the time evolution direction, the system is described by a quantum field theory living on a generalised cylinder $S^2 \times \mathbb{R}$, a manifold that is conformally equivalent to flat spacetime through the Weyl transformation

$$(\hat{\mathbf{n}}, \tau) \in S^2 \times \mathbb{R} \longmapsto r\hat{\mathbf{n}} \in \mathbb{R}^3, \qquad r = e^{\tau/R} \tag{2.2}$$

where R is the radius of the sphere. This conformal transformation maps each time slice of the cylinder to a cocentric sphere in the the flat spacetime.

Thanks to the conformal flatness that is not owned by other manifolds (*e.g.*, a lattice model with periodic boundary condition lives on the torus T^2 which is not conformally flat), we can make use of some nice properties of conformal field theories, the most important one of which is the state-operator correspondance. Specifically, there is a one-to-one correspondence between the eigenstates of the critical Hamiltonian on the sphere and the CFT operators. One can colloquially understand the state $|\Phi\rangle$ as the insertion of the corresponding operator $\Phi(0)$ at the origin point into the vacuum $|0\rangle : |\Phi\rangle = \Phi(0)|0\rangle$. The state and its corresponding operator has the same SO(3) spin and representation under global symmetry. More importantly, as the Weyl transformation maps the Hamiltonian H corresponding to the time translation on the cylinder to the dilatation D on the flat spacetime, the excitation energy of a state $|\Phi\rangle$ is proportional to the scaling dimension of the corresponding operator Δ_Φ

$$E_\Phi - E_0 = \frac{v}{R}\Delta_\Phi \tag{2.3}$$

where E_0 is the ground state energy, R is the radius of the sphere, and v is the speed of light that is dependent on the microscopic model and is the same for different states. With this property, one can calculate the scaling dimensions simply by obtaining the energy spectrum of the quantum Hamiltonian without doing complicated finite size scalings, and one can obtain the OPE coefficients simply from the inner product of a local operator.

Although the quantum Hamiltonians on a sphere enjoy the full rotation symmetry and the property of state-operator correspondence, it is difficult to put a lattice on the sphere due to the curvature (in particular the non-zero Euler characteristic), especially to recover an SO(3)-symmetric theomodynamic limit. An alternative way we take is to fuzzify the sphere. We consider charged free particles moving on a sphere with a magnetic monopole with a flux $4\pi s$ ($s \in \mathbb{Z}/2$) placed at its centre. The monopole exerts a uniform magnetic field on the sphere, which modifies the single particle Hamiltonian and the single particle eigenstates. Now the single particle eigenstates form highly degenerate spherical Landau levels. The lowest Landau level has a degeneracy $(2s+1)$. By setting the single particle gap to be the leading energy scale, adding interactions, and projecting onto the lowest Landau level, we obtain a finite Hilbert space. For the sake of numerical simulation, the system is analoguous to a length-$(2s+1)$ spin chain with long range interaction, where different Landau level orbitals behave like the lattice sites. The difference is that the $(2s+1)$ orbital forms a spin-s representation of the SO(3) rotation group, and in this way the continuous rotation symmetry is preserved. The word « fuzzy » means the non-commutativity, in our case, due to the presence of magnetic field. The non-commutativity provides a natural length scale which serves as a UV regulator of the quantum field theory. The radius of the sphere scales as $R \sim \sqrt{s}$. The thermodynic limit can be taken as $s \to \infty$, and we then recover a regular sphere without non-commutativity.

The power of this approach has been first demonstrated in the context of the 3D Ising transition, where the presence of emergent conformal symmetry has been convincingly established and a wealth of conformal data has been accurately computed. The study has then been extended to accessing various conformal data such as the OPE coefficients, correlation functions, entropic F-function, conformal generators in the 3d Ising CFT, studying conformal defects and boundaries such as the magnetic line defect, various conformal boundaries in 3D Ising CFT, and realising various 3d CFTs such as Wilson-Fisher CFTs, SO(5) deconfined criticality, Sp(N) symmetric CFTs, *etc.* In the following sections, we shall review the existing works, technical details and the numerical methods.

3 Review of existing work on fuzzy sphere

In this section, we review the existing work related to fuzzy sphere.

The pioneering work [1] This work first proposes the idea of fuzzy sphere and apply it to a pedagogical example of 3d Ising CFT. This work constructs a model with two flavours of fermions that resembles the spin-up and spin-down in the lattice transverse-field Ising model. At half-filling, one can colloquially think that a spin degree of freedom lives on each orbital. The Hamiltonian contains a density-density interaction that resembles the Ising ferromagnetic interaction and a polarising terms that resembles the transverse field. By tuning the ratio of the two terms, a transition between quantum Hall ferromagnet (a two-fold degenerate state where either of the two flavours is completely occupied) and paramagnet (a one-fold degenerate state where the superpositions of the two flavours at each orbital are occupied) occurs. This transition spontaneously breaks a $\mathbb{Z}_2$ symmetry and falls into the Ising criticality. This work then make use of a unique feature of spherical models described by CFT — state-operator correspondence — at the critical point to extract the scaling dimensions of the scaling local operators. This work finds evidence for conformal symmetry, including that (1) there exists a conserved stress tensor with $\Delta = 3$ (which is used as the calibrator), and (2) all the levels can be classified into conformal multiplet where the spacing between operators' scaling dimensions are very close to integer. This is one of the first numerical evidence that 3d Ising transition has emergent conformal symmetry. More remarkably, the scaling dimensions of primaries such as $\sigma, \epsilon, \epsilon'$ are already very close to the most accurate known value by numerical bootstrap with an error within 1.2% at a small system size $N_m = 16$, for which the computational cost is comparable to a 4×4 lattice system. The structure of Ising CFT operator spectrum already starts to show up at an even smaller system size $N_m = 4$. All these clues point towards a curious observation that fuzzy sphere suffers from a remarkably small finite-size effect. The detail for the construction of models is given in Sections 4.3 and 4.4, and the detail for analysis of spectrum is given in Section 4.5.

This seminal work opens a new avenue for studying 3d conformal field theories. After that, most of the researches on fuzzy sphere can roughly be catagorised into three directions (with several exceptions) :

1. Accessing various conformal data,
2. Realising various 3d CFTs, and
3. Studying conformal defects and boundaries.

3.1 Accessing various conformal data

The first direction is to develop methods to calculate various data and quantities of 3d CFTs on fuzzy sphere. Typically, these methods are tested on the simplest example of 3d Ising CFT. For many of those CFT data, fuzzy sphere is the first non-perturbative method to access them ; for the others, the fuzzy sphere has achieved great consistency with previous methods such as quantum Monte Carlo and conformal bootstrap. So far, the accessible CFT data include operator spectrum, OPE coefficients, correlation functions, entropic F-function and conformal generators.

OPE coefficients [2] Apart from the operator spectrum that has been studied in Ref. [], a wealth of CFT data can be obtained from the local operators. This work studies the local observables on the fuzzy sphere, including the density operators and certain four-fermion operators. These observables can be expressed as the linear combination of CFT local scaling operators. After a finite size scaling that takes into account the data from different system sizes, the subleading contribution can be substracted and only the leading contribution are left. In this way, the lowest primaries in Ising CFT in each symmetry sector, *viz.* $\mathbb{Z}_2$-odd σ and $\mathbb{Z}_2$-even ϵ, can be realised. The OPE coefficients are then evaluated by taking the inner product of a fuzzy sphere local observable with two CFT states $\langle\Phi_1|\Phi_2(\hat{\mathbf{n}})|\Phi_3\rangle$. This work computes 17 OPE coefficients of low-lying CFT primary fields with high accuracy, including 4 that has not being reported before. The rest are consistent with numerical bootstrap results. It is also worth noting that this work start to apply DMRG to the fuzzy sphere. The maximal system size is increased from $N_m = 18$ by ED to $N_m = 48$ by DMRG. The detail for calculating the OPE coefficient is given in Section 4.6.

Correlation functions [3] In addition to the OPE coefficients, the local observables can also be used to calculate correlation functions. By taking the inner product of two local observables (density operators) at a time displacement $\langle\Phi_1|\Phi_2(\hat{\mathbf{n}}_0)\Phi_3(\hat{\mathbf{n}},\tau)|\Phi_4\rangle$ with two CFT states, a

general four-point function can be calculated. This piece of CFT data in practice cannot be derived from the rest. This work calculates this four-point function in 3d Ising CFT with DMRG. A non-trivial check of conformality, the crossing symmetry, is verified for the correlator $\langle\sigma\sigma\sigma\sigma\rangle$. The special case — two-point functions by taking $\Phi_1 = \Phi_4 = \mathbb{I}$ — are also studied and compared with the expected results by conformal symmetry. The detail for calculating the correlation functions is given in Section 4.6.

Entropic F-function [10] Beyond the correlators of local operators, a wealth of information can be learnt from the entanglement entropy and entanglement spectrum. A remarkable quantity is called the F-function, which is defined through the scaling behaviour of the entanglement entropy. Specifically, consider a quantum system that lives on $\mathbb{R}^2$. A circle with radius R_d divides the system into inner part A and outer part B. The entanglement entropy is defined and expected to scale with R_d as

$$S_A(R_d) = \mathrm{tr}_A\, \rho \log \rho = \alpha R_d/\delta - F \tag{3.1}$$

where δ is a UV-regulator. The constant part is known as the F-function of a 3d CFT. The F-function is proved to be RG-monotonic, *i.e.*, along a renormalisation group flow from UV to IR, the value of F-function is non-increasing, analogous to the central charge in 2d CFTs. Despite its importance, it has never been calculated before through non-perturbative approaches in interacting 3d CFTs. This work has performed the first non-perturbative computation of F function for paradigmatic 3d Ising CFT on fuzzy sphere. The sphere is cut in the real space into two crowns along a latitude circle θ, and the entanglement entropy $S_A(\theta)$ as a function of θ is calculated. The F-function is extracted from the $S_A(\theta)$ in vicinity of the equator, and the result yields $F_A = 0.0612(5)$ after a finite size scaling.

Conformal generators [14, 15] Within the generators of conformal symmetry, the SO(3) rotation and the dilatation are manifest and act as rotation and time translation on fuzzy sphere. The rest two, *viz.* translation P^μ and special conformal transformation (SCT) K^μ needs to be emergent in the IR at the conformal point but broken along the RG flow. It is worthwhile to construct these IR generators by the UV operators on fuzzy sphere. These works invest in such construction with the help of stress tensor $T^{\mu\nu}$. The time component $T^{\tau\tau}$ of stress

tensor equals the Hamiltonian density $\mathcal{H}$ and it integrates into the generator $\Lambda^\mu = P^\mu + K^\mu = \int \mathrm{d}\hat{\mathbf{n}}\, n^\mu \mathcal{H}$. The action of this generator send a scaling operator to other operators in the same multiplet with the number of partial derivatives increased or decreased by one. These works calculate the matrix elements of the generators Λ^μ and compare it with the theoretical values in the CFT and find good agreement, which is another non-trivial verification of conformal symmetry. Furthermore, the separate generators P^μ and K^μ can be obtained by considering the commutatator $[H, \Lambda^\mu]$, which is useful in determining the primaries. The detail for constructing the conformal generators is given in Section 4.7.

3.2 Realising various 3d CFTs

The second direction is to study various other CFTs beyond 3d Ising. Fuzzy sphere has revealed many new information about these theories ; the previously known results are also consistent with the fuzzy sphere. So far, the accessible CFTs include SO(5) deconfined criticality, O(3) Wilson-Fisher and a series of new theories with Sp(N) symmetry.

The SO(5) **deconfined criticality [4]** The first theory besides Ising CFT to which fuzzy sphere is applied is the SO(5) deconfined quantum critical point (DQCP). Deconfined quantum critical point (DQCP) is one of the pioneering example of phase transitions beyond Landau paradigm. It has led to numerous theoretical surprises including the emergent SO(5) symmetry and the duality between interacting theories. Despite extensive studies over the past two decades, its nature remains controversial. The two competing scenarios are (1) DQCP is truely critical, and (2) DQCP is pseudocritical, *i.e.*, a weakly first-order phase transition that has approximate critical behaviour, and is controlled by a pair of complex fixed points very close to the pseudocritical region.

The DQCP can be conveniently studied on the fuzzy sphere by constructing a non-linear sigma model (NLσM) on target space S^4 with a level-1 topological Wess-Zumino-Witten (WZW) term, which serves as a dual description of the DQCP with an exact SO(5) symmetry. The idea is to construct a four-flavour model with global symmetry $\mathrm{Sp}(2)/\mathbb{Z}_2 = \mathrm{SO}(5)$ ($\mathbb{Z}_2$ means to gauge the pseudoreal representations). At half-filling, it can be described by a NLσM on the Grassmannian $\frac{\mathrm{Sp}(2)}{\mathrm{Sp}(1)\times\mathrm{Sp}(1)} \cong S^4$ and the WZW level can be matched. This work provides ev-

idence that the DQCP exhibits approximate conformal symmetry. This work has identified 19 conformal primaries and their 82 descendants. Furthermore, by examining the renormalisation group flow of the lowest symmetry singlet, this work demonstrates that the DQCP is more likely pseudo-critical, with the approximate conformal symmetry plausibly emerging from nearby complex fixed points.

The O(3) **Wilson-Fisher [8]** The O(N) Wilson-Fisher theories are probably one of the most studied theories for 3d criticalities with wide range of applications. Specifically, this work focus on the O(3) WF CFT. The construction involves two copies of SU(2) ferromagnet with altogether 4 flavours. Briefly speaking, the model contains two competing terms : (1) a SU(2) ferromagnetic interaction which favours a Heisenberg ferromagnetic phase where each of the two copies being half-filled and the symmetry-breaking order parameter lives on a S^2 manifold, (2) a transverse field which favours one of the two copies being completely filled, corresponding to a Heisenberg paramagnet. The transition between these two phases falls into the O(3) Wilson-Fisher universality. Through the energy spectrum at the transition, this work provides evidence that O(3) Wilson-Fisher fixed point exhibits conformal symmetry, as well as revealing a wealth of information about the CFT can be revealed, such as the instability to cubic anisotropy. This work also calculates several OPE coefficients.

A series of new Sp(N)**-symmetric CFTs [16]** The quest to discover new 3d CFTs has been intriguing for physicists. A virgin land on this quest is the parity-breaking CFTs. In 3d, the Chern-Simons-matter theories stand out as the most well known and possibly the only known type of parity-breaking CFTs. Fuzzy sphere is a promising platform to study these theories. This work makes a concrete construction by generalising the DQCP to the WZW-NLσM on the target space of a general symplectic Grassmannian

$$\frac{\mathrm{Sp}(N)}{\mathrm{Sp}(M)\times\mathrm{Sp}(N-M)}. \tag{3.2}$$

Several candidate Chern-Simons-matter theories are known to exist on its phase diagram which have N flavour of gapless bosons or fermions coupled to a non-Abelian (*viz.* Sp(1), Sp(2), etc.) Chern-Simons gauge field. On the fuzzy sphere, this WZW-NLσM can be realised by a $2N$ layer model with Sp(N) global symmetry, and $2M$ out of the $2N$ layers are filled. This

work numerically verifies the emergent conformal symmetry by observing the integer-spaced conformal multiplets and studying the finite-size scaling of the conformality.

3.3 Studying conformal defects and boundaries

Apart from the bulk CFTs, fuzzy sphere can also be used to study their conformal defects and boundaries. Deforming a CFT with interactions living on a sub-dimensional defect may trigger a RG flow towards a non-trivial interacting IR fixed point. A defect IR theory that own a smaller conformal symmetry is called a defect CFT. The dCFTs own rich physical structure such as defect operators and bulk-to-defect correlation functions. Moreover, a bulk CFT can flow to several different dCFTs. So far, the accessible defects/boundaries include the magnetic line defect of 3d Ising CFT, including its defect operator spectrum, correlators, g-function, defect changing operators, its cusp, and the conformal boundaries of 3d Ising CFT.

Conformal magnetic line defect [6] This is the first work that studies conformal defects with fuzzy sphere. The simplest example of magnetic line defect of the 3d Ising CFT, where the defect line is completely polarised and the $\mathbb{Z}_2$ symmetry is explicitly broken. Taking a defect line along z-direction that passes the origin point, after the radial quantisation, this corresponds to the north and south poles of the sphere being polarised. Hence, to realise the magnetic line defect on fuzzy sphere, one only needs to apply a pinning magnetic field to the north and south poles (Since only the $m = +s$ orbital has non-zero amplitude at the north pole and $m = -s$ at the south pole due to the locality, one only need to pin the $m = \pm s$ orbitals).

This work demonstrates that the defect IR fixed point has emergent conformal symmetry $\mathrm{SO}(2,1) \times \mathrm{O}(2)$: in the operator spectrum, there exists a displacement operator as the non-conservation of stress tensor at exactly $\Delta_{\mathrm{D}} = 2$, and the defect primaries and descendants have integer spacing ; the bulk-to-defect one-point (1-pt) and two-point (2-pt) correlation functions follow a power law. This work has identified 6 low-lying defect primary operators and extract their scaling dimensions, as well as computing one-point bulk correlators and two-point bulk-defect correlators.

The g-function and defect changing operators [9] This work studies the g-function of conformal defects and the defect creation and changing operators. Similar to the central charge

and the F-function in bulk CFTs, there exists a RG-monotonic quantity called the g-function for the line defects that is non-increasing along the flow. It is defined as the ratio between the partition functions of the defect CFT and the bulk CFT. On the other hand, consider two semi-infinite magnetic line defects that are pinned towards opposite directions joint at one point, a defect changing operator lives at the joining point. Similarly, a defect creation operator lives at the endpoint of a semi-infinite line defect. The relevance of the defect changing operator is related to the stability of spontaneous symmetry-breaking (SSB) on the line defect.

This work realises the defect creation and changing operators for the Ising mangetic line defect by acting a pinning field at the north pole, and opposite pinning fields at the north and south poles, respectively. The scaling dimensions are calculated through state operator correspondence $\Delta_{\text{creation}} = 0.108(5), \Delta_{\text{changing}} = 0.84(5)$, indicating the instability of SSB on the Ising mangetic line. Moreover, this work shows that the g-function, along with many other CFT data, can be calculated by taking the overlaps between the eigenstates of different defect configurations. Most importantly, this paper has given the first non-perturbative result for the g-function $g = 0.602(2)$.

Cusp [11] A cusp is two semi-infinite defect lines joined at one point at an angle. This can be realised on fuzzy sphere through pinning fields at two points at an angle. This work studies the cusps through various theoretical and numerical approaches. In particular, on fuzzy sphere, this paper calculates the cusp anomalous dimension as a function of the angle for the Ising magnetic line defects, and verifies its relation with the Zamolodchikov norm of the displacement operator.

Conformal boundaries of 3d Ising CFT [12, 13] Apart from line defects, boundaries are also important extended objects in CFT. For the Iisng CFT, there exists several conformal boundaries : normal boundary CFT (bCFT) with explicitly broken $\mathbb{Z}_2$ symmetry, ordinary bCFT that is stable and has preserved $\mathbb{Z}_2$ symmetry, extraordinary bCFT with spontaneously broken $\mathbb{Z}_2$ symmetry, and special bCFT as the transition between ordinary and extraordinary bCFTs. These works focus on the normal and ordinary bCFTs and show that they can be realised by acting a polarising field on a hemisphere. By noting that the LLL orbitals are localised along latitude circles, the bCFTs can equivalently be realised by pinning the orbitals with $m < 0$. By studying the operator spectrum, these works show numerical evidence for conformal symmetry

and estimates the scaling dimensions of the conformal primaries. These works also calculates the bulk-to-boundary 1-pt and 2-pt functions and extract the corresponding OPE coefficients. Interestings, these works notice certain correspondence between the boundary energy spectrum and bulk entanglement spectrum through orbital cut.

3.4 Other works on the fuzzy sphere

Besides the three directions of works, several other works push the boundary of our knowledge of and techniques for the fuzzy sphere.

Conformal perturbation [5] The energy spectrum calculated numerically at finite size does not coincide with that of the CFT. Part of the finite-size correction comes from the higher irrelavant operators that are not exactly tuned to zero (*e.g.*, in the Ising CFT, the irrelavent operators include $\epsilon', C_{\mu\nu\rho\sigma}, T'_{\mu\nu}$, *etc.*, and the lowest singlets ϵ and ϵ' are tuned away through the two parameters). These irrelevant operators exert perturbations on the states and their energy. This paper captures this kind of correction by the conformal perturbation theory. By making use of the fact that the corrections from an irrelevant operator on the energy of the primary and its descendants are not independent, the coefficients of the irrelevant operators can be fitted.

Although this work is does not exactly carry out study on the fuzzy sphere, it opens up a new route of improving the precision of scaling dimensions on fuzzy sphere by making better use of the existing data, and the method to partly remove the finite-size correction through conformal perturbation theory is widely used by following works.

Quantum Monte Carlo on fuzzy sphere [7] Up to the time of this work, the numerical methods that has been applied to fuzzy sphere include exact diagonalisation (ED) and density matrix renormalisation group (DMRG). This work further presents the numerical studies of fuzzy sphere with quantum Monte Carlo (QMC) simulation, which is known for its potential of studying criticalities in $(2 + 1)$ dimensions at larger system size. Specifically, this work makes use of the determinant quantum Monte Carlo (DQMC) method that converts the simulation of fermions into the simulation of bosonic auxiliary fields. To overcome the sign problem, this work considers two copies of the original model and constructs the Ising CFT on a 4-flavour model.

This work determines the lowest energy spectra within each symmetry sector by calculating the time-displaced correlation functions. This work also calculates the equal-time correlation functions and compares them with the two-point functions of CFT.

Ising CFT on top of FQHE state [17] Up to the time of this work, all the constructions of CFTs on fuzzy sphere are based on the quantum Hall ferromagnet. Specifically, before the interaction is added, an integer number of the lowest Landau levels are fully occupied. This state has a finite charge gap that guarentees that the gapless spin degree of freedom do not strongly couple with the charge degree of freedom when one adds the interactions.

This work further explores the possibility to construct CFTs on other states with charge gap — in particular, the Haldane-Laughlin states that captures the fractional quantum Hall effect (FQHE). Specifically, this work studies the fermionic LLL at fillings of $\nu = 1/3$ and $1/5$. The model Hamiltonian contains (1) a dominant projection term that put the ground state on the Haldane-Laughlin state, and (2) an interaction term as a perturbation that drives the Ising-type phase transition. This work shows that the energy spectra at the critical point exhibit conformal symmetry. More noticeably, this work also makes the construction with respect to the bosonic LLL at a filling of $\nu = 1/2$.

4 Model construction on the fuzzy sphere

In this section, we review the process to construct a model on fuzzy sphere and extract conformal data. We are especially devoted to the technical detail that is rarely covered by other literature.

4.1 Projection onto the lowest Landau level

To build the setup of fuzzy sphere, we consider a sphere with radius R and put a $4\pi s$-monopole at its centre. Consider free electrons moving on the sphere. The monopole will modify the single particle Hamiltonian.

$$H_0 = \frac{1}{2MR^2}(\partial^\mu + iA^\mu)^2 \tag{4.1}$$

where $\mu = \theta, \phi$ and the gauge connection is taken as

$$A_\theta = 0, \quad A_\phi = -\frac{s}{R}\,\mathrm{ctg}\,\theta \tag{4.2}$$

The eigenstates of the Hamiltonian are the monopole spherical harmonics

$$Y_{lm}^{(s)}(\hat{\mathbf{n}}), \quad l = s, s+1, \ldots, \quad m = -l, \ldots, l-1, l \tag{4.3}$$

where $\hat{\mathbf{n}}$ is the unit vector of the point on the sphere specified by angular coordinates θ and ϕ, and the energies are

$$E_l = \frac{1}{2MR^2}(l(l+1) - s^2) \tag{4.4}$$

Each level, known as a Landau level, has a degeneracy $(2l+1)$. Specifically, the wavefunctions on the lowest Landau level (LLL) $l = s$ is easy to write out :

$$Y_{sm}^{(s)}(\hat{\mathbf{n}}) = C_m e^{im\phi} \cos^{s+m}\frac{\theta}{2} \sin^{s-m}\frac{\theta}{2}, \quad C_m = \frac{1}{\sqrt{4\pi \mathrm{B}(s+m+1, s-m+1)}} \tag{4.5}$$

where C_m is the normalising factor, and B is the Euler's beta function. The LLL has a degeneracy $N_m = 2s + 1$.

We now consider N_f flavours of fermions moving on the sphere, characterised by the second-quantised fermion operator $\hat{\psi}_f(\hat{\mathbf{n}})$, with a flavour index $f = 1, \ldots, N_f$. We partially fill the lowest Landau level and set the single energy gap to be much larger than the scale of interaction $H_0 \gg H_{\mathrm{int}}$, so that the quantum fluctuation can be constrained on the lowest Landau level. In practice, we often fill integer number of flavours $N_e = kN_m$ ($k \in \mathbb{Z}$) so that a quantum Hall ferromagnet (*i.e.*, the state where integer number of LLLs are filled) is preferred in the absence of interaction, for which the charge degree of freedom is gapped and will not couple strongly to the CFT degree of freedom in the presence of the interaction.

We then project the system onto the LLL. Technically, this can be done by write the fermion operators in terms of the annihilation operators of the LLL orbitals

$$\hat{\psi}_f(\hat{\mathbf{n}}) = \sum_{m=-s}^{s} Y_{sm}^{(s)}(\hat{\mathbf{n}})\hat{c}_{mf} \tag{4.6}$$

where $\hat{c}_{mf}^{(\dagger)}$ annihilates/creates an electron with L^z-quantum number m at the f-th flavour of the lowest Landau level. In the following sections, we will omit the hats on the operators.

After the projection, we obtain a finite Hilbert space on which numerical simulation can be carried out. For the sake of numerical simulation, the system is analoguous to a length-$(2s+1)$ spin chain with long range interaction, where different Landau level orbitals behave like the lattice sites. The difference is that the $(2s+1)$ orbital forms a spin-s representation of the SO(3) rotation group, and in this way the continuous rotation symmetry is preserved. The exact rotation symmetry reduces the UV effect and the finite-size effect, so that the numerical results are considerably accurate even at small system size.

The word « *fuzzy* » means non-commutativity. Here the magnetic field results in a the non-commutativity of the coordinates. More concretely, we project write the coordinate operators as a matrix on the lowest Landau level

$$X^{\mu}_{m_1 m_2} = \int \mathrm{d}^2\hat{\mathbf{n}}\, n^{\mu} \bar{Y}^{(s)}_{sm_1}(\hat{\mathbf{n}}) Y^{(s)}_{sm_2}(\hat{\mathbf{n}}) \tag{4.7}$$

These matrices $\mathbf{X}^{\mu}$ $(\mu = x, y, z)$ satisfy relation

$$\mathbf{X}_{\mu}\mathbf{X}^{\mu} = \frac{s}{s+1}\mathbb{I}, \qquad [\mathbf{X}^{\mu}, \mathbf{X}^{\nu}] = \frac{1}{s+1} i\epsilon^{\mu\nu\rho}\mathbf{X}_{\rho} \tag{4.8}$$

The first equation involves the radius R of the sphere, and the second equation involves the magnetic length l_B that determines the non-commutativity. An arbitrary scale factor can change these lengths but their ratio is fixed and scales as

$$R/l_B \sim \sqrt{s} \sim \sqrt{N_m} \tag{4.9}$$

We can take $l_B = 1$ as the unit length. In this way, the radius scales with the square root of number of orbitals. The thermodynamic limit can be taken by $N_m \to \infty$, where a regular sphere is recovered.

4.2 Density operator

Having constructed the single particle states, we then consider the interacting many-body Hamiltonian. The simplest building block is the density operator, *i.e.*, local fermion bilinear.

$$n_M(\hat{\mathbf{n}}) = \psi^{\dagger}_{f'}(\hat{\mathbf{n}}) M_{f'f} \psi_f(\hat{\mathbf{n}}) \tag{4.10}$$

Here the matrix insertion M put the density operators at a certain representation of the flavour symmetry. For example, for a 2-flavour system, M can be taken as the Pauli matrices $\mathbb{I}, \sigma^x, \sigma^y, \sigma^z$;

for a system with N_f flavours in the fundamental representation of SU(N_f) flavour symmetry, one can put n_M in the singlet or adjoint representation

$$\begin{aligned} n_S(\hat{\mathbf{n}}) &= \psi_c^\dagger(\hat{\mathbf{n}})\psi^c(\hat{\mathbf{n}}) \\ n_a{}^b(\hat{\mathbf{n}}) &= \psi_a^\dagger(\hat{\mathbf{n}})\psi^b(\hat{\mathbf{n}}) - \tfrac{1}{N}\delta_a{}^b\psi_c^\dagger(\hat{\mathbf{n}})\psi^c(\hat{\mathbf{n}} \end{aligned} \tag{4.11}$$

Like the fermion operator, the density operator can also be expressed in the orbital space.

$$n_M(\hat{\mathbf{n}}) = \sum_{lm} Y_{lm}(\hat{\mathbf{n}}) n_{M,lm} \tag{4.12}$$

Conversely,

$$\begin{aligned} n_{M,lm} &= \int \mathrm{d}^2\hat{\mathbf{n}}\, \bar{Y}_{lm} n_M(\hat{\mathbf{n}}) \\ &= \int \mathrm{d}^2\hat{\mathbf{n}}\, \bar{Y}_{lm}\left(\sum_{m_1} \bar{Y}^{(s)}_{sm_1} c^\dagger_{m_1 f_1}\right) M_{f_1 f_2}\left(\sum_{m_2} Y^{(s)}_{sm_2} c_{m_1 f_2}\right) \\ &= \sum_{m_1 m_2} c^\dagger_{m_1 f_1} M_{f_1 f_2} c_{m_1 f_2} \int \mathrm{d}^2\hat{\mathbf{n}}\, \bar{Y}_{lm} \bar{Y}^{(s)}_{sm_1} Y^{(s)}_{sm_2} \\ &= \sum_{m_1} c^\dagger_{m_1 f_1} M_{f_1 f_2} c_{m+m_1, f_2} \times \\ &\qquad (-1)^{s+m+m_1}(2s+1)\sqrt{\frac{2l+1}{4\pi}} \begin{pmatrix} s & l & s \\ m_1 & m & -m_1 - m \end{pmatrix} \begin{pmatrix} s & l & s \\ m_1 & m & -m_1 - m \end{pmatrix} \end{aligned} \tag{4.13}$$

Here we have used the properties of the monopole spherical harmonics[1]

$$\bar{Y}^s_{lm} = (-1)^{s+m} Y^{(-s)}_{l,-m} \tag{4.14a}$$

$$\int \mathrm{d}^2\hat{\mathbf{n}}\, Y^{(s)}_{lm} \bar{Y}^{(s)}_{lm} = \delta_{ll'}\delta_{mm'} \tag{4.14b}$$

$$\int \mathrm{d}^2\hat{\mathbf{n}}\, Y^{(s_1)}_{l_1 m_1} Y^{(s_2)}_{l_2 m_2} Y^{(s_3)}_{l_3 m_3} = \sqrt{\frac{(2l_1+1)(2l_2+1)(2l_3+1)}{4\pi}} \begin{pmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \end{pmatrix} \begin{pmatrix} l_1 & l_2 & l_3 \\ -s_1 & -s_2 & -s_3 \end{pmatrix} \tag{4.14c}$$

[1] For more detail, one can refer to the Wikipedia page https://en.wikipedia.org/wiki/Spin-weighted_spherical_harmonics.

and $\begin{pmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \end{pmatrix}$ is the $3j$-symbol[2]. In this way, we have fully expressed the density operator in terms of the operators in the orbital space $c^{(\dagger)}_{mf}$.

4.3 Density-density interaction

The most straightforward way to construct an interaction term is to add a density-density interaction with a potential function. We note that this is not the simplest construction and we will present the simpler construction in terms of pseudopotentials in the next section.

$$H_{\text{int}} = \int \mathrm{d}^2\hat{\mathbf{n}}_1 \, \mathrm{d}^2\hat{\mathbf{n}}_2 \, U(|\hat{\mathbf{n}}_1 - \hat{\mathbf{n}}_2|) n_M(\hat{\mathbf{n}}_1) n_M(\hat{\mathbf{n}}_2) \tag{4.15}$$

The interacting potentials can be expanded in terms of the Legendre polynomials

$$U(|\mathbf{r}_{12}|) = \sum_l \tilde{U}_l P_l(\cos\theta_{12}) = \sum_{lm} \frac{4\pi}{2l+1} \bar{Y}_{lm}(\hat{\mathbf{n}}_1) Y_{lm}(\hat{\mathbf{n}}_2) \tag{4.16}$$

where $\mathbf{r}_{12} = \hat{\mathbf{n}}_1 - \hat{\mathbf{n}}_2$ and $|\mathbf{r}_{12}| = 2\sin\theta_{12}/2$. Conversely

$$\tilde{U}_l = \int \sin\theta_{12} \mathrm{d}\theta_{12} \, \frac{2l+1}{2} U(|\mathbf{r}_{12}|) P_l(\cos\theta_{12}) \tag{4.17}$$

Specifically, for local and super-local interactions

$$\begin{aligned} U(|\mathbf{r}_{12}|) &= \delta(\mathbf{r}_{12}) & \tilde{U}_l &= 2l+1 \\ U(|\mathbf{r}_{12}|) &= \nabla^2\delta(\mathbf{r}_{12}) & \tilde{U}_l &= -l(l+1)(2l+1) \end{aligned} \tag{4.18}$$

By expanding the density operators into the orbital space and completing the integrals,

$$H_{\text{int}} = \sum_{lm} \frac{4\pi\tilde{U}_l}{2l+1} n^\dagger_{M,lm} n_{M,lm} \tag{4.19}$$

With these ingrediants, we can now consider how to construct models. This comes down to matching the symmetry and phase diagram. *E.g.*, for the Ising model, the $\mathbb{Z}_2$ global symmetry is realised as the exchange of the two flavours $\psi_\uparrow(\mathbf{r}) \leftrightarrow \psi_\downarrow(\mathbf{r})$. We need a phase diagram with a paramagnetic (PM) phase where the $\mathbb{Z}_2$ symmetry is conserved and a ferromagnetic phase

[2] For more detail, one can refer to the Wikipedia page https://en.wikipedia.org/wiki/3-j_symbol.

where the $\mathbb{Z}_2$ symmetry is spontaneously broken. The PM phase is favoured by a polarising term that resembles a transverse field

$$-h \int \mathrm{d}^2\hat{\mathbf{n}}\, n_x(\hat{\mathbf{n}})$$

and the FM phase where either of the two flavours is fully filled is favoured by a repulsion between the two flavours

$$\int \mathrm{d}^2\hat{\mathbf{n}}_1\, \mathrm{d}^2\hat{\mathbf{n}}_2\, U(|\hat{\mathbf{n}}_1 - \hat{\mathbf{n}}_2|) n_\uparrow(\hat{\mathbf{n}}_1) n_\downarrow(\hat{\mathbf{n}}_2)$$

where the density operators are defined as

$$n_x(\hat{\mathbf{n}}) = \psi_\downarrow^\dagger(\hat{\mathbf{n}})\psi_\uparrow(\hat{\mathbf{n}}) + \psi_\uparrow^\dagger(\hat{\mathbf{n}})\psi_\downarrow(\hat{\mathbf{n}}), \quad n_\uparrow = \psi_\uparrow^\dagger(\hat{\mathbf{n}})\psi_\uparrow(\hat{\mathbf{n}}), \quad n_\downarrow = \psi_\downarrow^\dagger(\hat{\mathbf{n}})\psi_\downarrow(\hat{\mathbf{n}})$$

and the potentials can be most conveniently taken as a combination of local and super-local interactions. Altogether the model Hamiltonian reads

$$H_{\text{int}} = \int \mathrm{d}^2\hat{\mathbf{n}}_1\, \mathrm{d}^2\hat{\mathbf{n}}_2\, U(|\hat{\mathbf{n}}_1 - \hat{\mathbf{n}}_2|) n_\uparrow(\hat{\mathbf{n}}_1) n_\downarrow(\hat{\mathbf{n}}_2) - h \int \mathrm{d}^2\hat{\mathbf{n}}\, n_x(\hat{\mathbf{n}}) \tag{4.20}$$

By tuning the ratio between h and $U(\mathbf{r}_{12})$, a phase transition described by the Ising CFT is realised.

4.4 Interaction in terms of pseudopotentials

Another way that is much more convenient to construct the interactions is through Haldane pseudopotential. To explain the idea, we take the 3d Ising model as an example. We first classify all the fermion bilinears $\lambda_{mm'ff'} c_{mf} c_{m'f'}$. To simplify the discussion, we can take a specific isospin index $\lambda_{mm'} c_{m\uparrow} c_{m'\downarrow}$. The fermion bilinears can be classified into irreducible representations of SO(3) rotation symmetry. Since c_{mf} carries the spin-s representation, the spin of its bilinear ranges from 0 to $2s$ and takes integer values. The spin-$(2s - l)$ combination reads

$$\Delta_{lm} = \sum_{m_1} \langle s m_1, s(m - m_1) | (2s - l) m \rangle c_{m_1,\uparrow} c_{m - m_1,\downarrow} \tag{4.21}$$

where $m = -(2s-l), \ldots, (2s-l)$, and the Clebshbar-Gordan coefficients[3] is related to the $3j$-symbol by

$$\langle l_1 m_1, l_2 m_2 | l m \rangle = (-1)^{-l_1+l_2-m}\sqrt{2l+1}\begin{pmatrix} l_1 & l_2 & l \\ m_1 & m_2 & -m \end{pmatrix} \tag{4.22}$$

A four-fermion interaction term is formed by contracting these paring operators with its conjugate.

$$H = \sum_l U_l H_l, \quad H_l = \sum_m \Delta^\dagger_{lm}\Delta_{lm} \tag{4.23}$$

Putting these altogether, the interaction Hamiltonian can be expressed as

$$H = \sum_{l,m_1 m_2 m_3 m_4} U_l C^l_{m_1 m_2 m_3 m_4} c^\dagger_{m_1\uparrow} c^\dagger_{m_2\downarrow} c_{m_3\downarrow} c_{m_4\uparrow} - h\sum_m (c^\dagger_{m\uparrow}c_{m\downarrow} + \text{h.c.}) \tag{4.24}$$

where the matrix elements are

$$C^l_{m_1 m_2 m_3 m_4} = \delta_{m_1+m_2,m_3+m_4}\langle s m_1, s m_2 | (2s-l)(m_1+m_2)\rangle\langle s m_3, s m_4 | (2s-l)(m_3+m_4)\rangle \tag{4.25}$$

The coupling strength U_l of the spin-$(2s-l)$ channel is called the Haldane pseudopotentials.

We need also to consider the constraint that the two fermions must be anti-symmetrised : for even l, the orbital index is symmetrised, so the spin index must be antisymmetrised, so the two fermions form a spin-singlet which is invariant under the SU(2) transformation ; for odd l, the orbital index is anti-symmetrised, so the spin index is symmetrised, breaking the flavour symmetry from SU(2) to $\mathbb{Z}_2$. Hence, an odd-l pseudopotential must be added (This fact escapes the construction by density-density interaction).

The fermion bilinears with other isospin configurations $\lambda_{mm',\pm}(c_{m\uparrow}c_{m'\uparrow} \pm c_{m\downarrow}c_{m'\downarrow})$ can be analysed in a similar way. After that, we have enumerated all possible four-fermion interaction terms.

For systems with more complicated continuous symmetries, classification in terms of representation of flavour symmetry must also be considered, and the indices must be overall anti-symmetrised. We explain that through the example of a $2N$-flavour system with Sp(N) global symmetry []. The maximal flavour symmetry is SU($2N$), so interactions must be added to break

[3]For more detail, one can refer to the Wikipedia page https://en.wikipedia.org/wiki/Clebshbar-Gordan_coefficients.

the symmetry from SU($2N$) to Sp(N). The fermion operators live in the Sp(N) fundamental representation. We shall show that all the allowed terms are

$$H = \sum_{\substack{l\in\mathbb{Z}\\ m_1m_2m_3m_4}} U_l C^l_{m_1m_2m_3m_4} c^\dagger_{m_1a} c^\dagger_{m_2b} c_{m_3b} c_{m_4a} - \frac{1}{2} \sum_{\substack{l\in 2\mathbb{Z}\\ m_1m_2m_3m_4}} V_l C^l_{m_1m_2m_3m_4} \Omega_{aa'}\Omega_{bb'} c^\dagger_{m_1a} c^\dagger_{m_2a'} c_{m_3b'} c_{m_4b} \quad (4.26)$$

where $\Omega = \begin{pmatrix} 0 & \mathbb{I}_N \\ -\mathbb{I}_N & 0 \end{pmatrix}$.

To find out all the four-fermion interactions allowed by the rotation symmetry SO(3) and flavour symmetry Sp(N), we classify all the fermion bilinears $c_{m_1a}c_{m_2b}$ into irreducible representations (irrep) of SO(3) $\times$ Sp(N). For each irrep, by contracting the bilinear with its Hermitian conjugate, we obtain an allowed four-fermion interaction term. Each fermion carries SO(3) spin-s and Sp(N) fundamental. For the rotation symmetry SO(3), the bilinear can carry spin-$(2s-l)(l=0,\ldots,2s)$ represetation ; for even l, the orbital indices are symmetrised ; for odd l, the orbital indices are antisymmetrised. For the flavour symmetry Sp(N), the bilinear can carry singlet S, traceless antisymmetric rank-2 tensor A and symmetric rank-2 tensor T representation ; for S and A, the flavour indices are antisymmetrised ; for T, the flavour indices are symmetrised. As the two fermions altogether should be antisymmetrised, the allowed combinations are

1. Sp(N) singlet and SO(3) spin-$(2s-l)$ with even l, the bilinears are

$$\Delta_{lm} = \sum_{m_1m_2} \langle sm_1, sm_2|(2s-l)m\rangle \Omega_{aa'} c_{m_1a} c_{m_2a'} \delta_{m,m_1+m_2} \quad (4.27)$$

The corresponding interaction term $H_{S,l} = \sum_m \Delta^\dagger_{lm}\Delta_{lm}$ is the even-l pseudopotential for the V-term.

2. Sp(N) antisymmetric and SO(3) spin-$(2s-l)$ with even l, the bilinears are

$$\Delta_{lm,[ab]} = \sum_{m_1m_2} \langle sm_1, sm_2|(2s-l)m\rangle \left(c_{m_1a}c_{m_2b} - c_{m_1b}c_{m_2a} - \tfrac{1}{N}\Omega_{ab}\Omega_{cc'}c_{m_1c'}c_{m_2c}\right)\delta_{m,m_1+m_2}. \quad (4.28)$$

The corresponding interaction term $H_{A,l} = \sum_m \Delta^\dagger_{lm,[ab]}\Delta_{lm,[ab]}$ is the even-l pseudopotential for the U-term.

3. Sp(N) symmetric and SO(3) spin-$(2s-l)$ with odd l, the bilinears are

$$\Delta_{lm,(ab)} = \sum_{m_1 m_2} \langle s m_1, s m_2 | (2s-l) m \rangle \left(c_{m_1 a} c_{m_2 b} + c_{m_1 b} c_{m_2 a} \right) \delta_{m, m_1+m_2}. \tag{4.29}$$

The corresponding interaction term $H_{T,l} = \sum_m \Delta^\dagger_{lm,(ab)} \Delta_{lm,(ab)}$ is the odd-l pseudopotential for the U-term.

In summary, all allowed interactions are the U_l terms with both even and odd l, and the V_l terms with only even l.

We also note that each pseudopotential can correspond to a profile of interaction potential functions. The conversion between the pseudopotentials U_l and the Legendre expansion coefficients of the potential function $\tilde{U}_l$

$$U(|\mathbf{r}_{12}|) = \sum_l \tilde{U}_l P_l(\cos\theta_{12}) \tag{4.30}$$

is

$$U_l = \sum_k \tilde{U}_k (-1)^l (2s+1)^2 \begin{Bmatrix} 2s-l & s & s \\ k & s & s \end{Bmatrix} \begin{pmatrix} s & k & s \\ -s & 0 & s \end{pmatrix}^2 \tag{4.31}$$

where $\{...\}$ is the $6j$-symbol. Specifically, a local interaction contains only pseudopotential U_0 ; a superlocal interaction of form $(\nabla^2)^l \delta(\mathbf{r}_{12})$ contains $U_0, U_1, ..., U_l$. Here we give the expressions for the lowest pseudopotentials explicitly.

$$\begin{aligned} &U(|\mathbf{r}_{12}|) = \delta(\mathbf{r}_{12}) && U_0 = \frac{(2s+1)^2}{4s+1} \\ &U(|\mathbf{r}_{12}|) = \nabla^2\delta(\mathbf{r}_{12}) && U_0 = -\frac{s(2s+1)^2}{4s+1} \qquad U_1 = \frac{s(2s+1)^2}{4s-1}. \end{aligned} \tag{4.32}$$

More details are given in Ref. [15].

4.5 Operator spectrum and search for conformal point

Having introduced the construction of an interacting model on fuzzy sphere, we now turn to the verification of the conformal symmetry and the extraction of the CFT data. The most straightforward approach is to extract the scaling dimensions from the energy spectrum through the state-operator correspondence. Specifically, there is a one-to-one correspondence between the eigenstates of the Hamiltonian and the CFT operators. The state and its corresponding

operator has the same SO(3) spin and representation under flavour symmetry, and the excitation energy of a state $|\Phi\rangle$ is proportional to the scaling dimension of the corresponding operator Δ_Φ

$$E_\Phi - E_0 = \frac{v}{R}\Delta_\Phi \tag{4.33}$$

where E_0 is the ground state energy, R is the radius of the sphere (here we take $R = \sqrt{N_m}$), and v is the model-dependent speed of light. The constant v/R can be determined through a calibration process, *i.e.*, comparing the spectrum to some known properties of a CFT spectrum. The criteria to determine the conformal symmetry include

1. The existence of a conserved stress tensor $T^{\mu\nu}$. The stress tensor is the symmetry current of the translation transformation. It is known to be a singlet under the flavour symmetry, have spin-2 under SO(3) rotation and scaling dimension exactly $\Delta_{T^{\mu\nu}} = 3$.

2. The existence of a conserved flavour symmetry current J^μ if there is a continuous flavour symmetry. The symmetry current typically lives in the antisymmetric rank-2 tensor representation of the flavour symmetry. *E.g.*, if the flavour symmetry is U(1), then the symmetry current has charge-0 ; if the flavour symmetry is O(3), then the symmetry current has spin-1 and is odd under the improper $\mathbb{Z}_2$ transformation ; if the flavour symmetry is O(n) ($n \geq 4$) or SU(n) ($n \geq 3$), then the symmetry current lives in the antisymmetric rank-2 tensor representation.

3. The organisation of the operator spectrum into conformal multiplets. All the levels in the spectrum of a CFT can be organised into the conformal primaries and their descendants. The descendants live in the same representation under the flavour symmetry as the primary, and the difference between the scaling dimensions of a primary and its descendant is an integer. Specifically, for a scalar primary Φ, its descendants have the form

$$\Box^n \partial^{\mu_1}\partial^{\mu_2}\ldots\partial^{\mu_l}\Phi - (\text{trace}), \qquad n, l = 0, 1, 2, \ldots$$

with SO(3) spin-l and scaling dimension $\Phi + 2n + l$, where $\Box = \partial_\mu\partial^\nu$. For a spinning primary $\Phi^{\mu_1\cdots\mu_s}$, its descendants has the two forms :

$$\Box^n \partial^{\nu_1}\ldots\partial^{\nu_m}\partial_{\rho_1}\ldots\partial_{\rho_k}\Phi^{\rho_1\cdots\rho_k\mu_1\cdots\mu_{s-k}}, \qquad k = 0, \ldots, s,\ n, m = 0, 1, \ldots$$

with scaling dimension $\Delta_\Phi + k + m + 2n$ and SO(3) spin-$(s - k + m)$, and

$$\Box^n \partial^{\nu_1}\ldots\partial^{\nu_m}\partial_{\rho_1}\ldots\partial_{\rho_k}\epsilon^{\sigma}{}_{\tilde{\mu}\tilde{\nu}}\partial^{\tilde{\nu}}\Phi^{\rho_1\cdots\rho_k\tilde{\mu}\mu_1\cdots\mu_{s-k-1}}, \qquad k = 0, \ldots, s-1,\ n, m = 0, 1, \ldots$$

with scaling dimension $\Delta_\Phi + k + m + 2n + 1$ and SO(3) spin-$(s - k + m)$. For the second form, the fully antisymmetric tensor ϵ alters the parity.

The most convenient way of determining the coefficient v/R is by utilising criteria 1 or 2 :

$$\frac{v}{R} = \frac{E_{T^{\mu\nu}} - E_0}{3} \quad \text{or} \quad \frac{E_{J^\mu} - E_0}{2} \tag{4.34}$$

Alternatively, one can define a cost function that depends on the tuning parameter and the speed of light and compares the scaling dimensions obtained from fuzzy sphere and the prediction by conformal symmetry. *E.g.*, for the Ising CFT, the tuning parameters are the pseudopotentials $\{U_i\}$ and the transverse field h. The criteria for conformal symmetry we use include the stress tensor $T^{\mu\nu}$ and the descendants $\partial^\mu\sigma$, $\partial^\mu\partial^\nu\sigma$, $\Box\sigma$, $\partial^\mu\epsilon$. The cost function is the root-mean-square of the deviations of these criteria from the expectation of the conformal symmetry

$$\begin{aligned} Q(\{U_i\}, h, v; N_m) = \frac{1}{N_s}\Big[&(\Delta^{(\mathrm{FS})}_{T^{\mu\nu}} - 3)^2 + (\Delta^{(\mathrm{FS})}_{\partial^\mu\sigma} - \Delta^{(\mathrm{FS})}_{\sigma} - 1)^2 \\ &+ (\Delta^{(\mathrm{FS})}_{\partial^\mu\partial^\nu\sigma} - \Delta^{(\mathrm{FS})}_{\sigma} - 1)^2 + (\Delta^{(\mathrm{FS})}_{\Box\sigma} - \Delta^{(\mathrm{FS})}_{\sigma} - 1)^2 + (\Delta^{(\mathrm{FS})}_{\partial^\mu\epsilon} - \Delta^{(\mathrm{FS})}_{\epsilon} - 1)^2 \Big] \end{aligned} \tag{4.35}$$

where $N_s = 5$ is the number of criteria, the scaling dimensions of an operator Φ on the fuzzy sphere is determined as

$$\Delta^{(\mathrm{FS})}_{\Phi}(\{U_i\}, h, v; N_m) = \frac{E_\Phi - E_0}{v/R}. \tag{4.36}$$

The optimal conformal point and calibrator are determined by minimising this cost function for each system size N_m. Note that this optimal point depends on the system size. In order to do finite size scaling, if the CFT describes a phase transition, one could fix all but one parameters at the optimal point in the largest accessible system size and tune the last parameter to determine the critical point through a finite size scaling.

4.6 Local observables

We have introduced how to determine the scaling dimensions from the energy spectrum. Beyond that, evaluating other CFT quantities requires realising local CFT operators on the fuzzy sphere. Any gapless local observables $\mathcal{O}(\hat{\mathbf{n}})$ on the fuzzy sphere can be written as the linear combination of CFT operators that lives in the same representation of flavour symmetry and parity.

$$\mathcal{O}(\hat{\mathbf{n}}, \tau) = \sum_\alpha \lambda_\alpha \Phi^{(\mathrm{cyc.})}_\alpha(\hat{\mathbf{n}}, \tau) \tag{4.37}$$

Here special care should be taken for the CFT operator $\Phi_\alpha^{(\mathrm{cyc.})}(\hat{\mathbf{n}},\tau)$ on the cylinder. A conformal transformation produces a scale factor $\Lambda(\mathbf{r})^{\Delta_\Phi}$ to a primary operator Φ. For the Weyl transformation from the flat spacetime to the cylinder, the scale factor is $\Lambda(\mathbf{r}) = r/R$. Hence,

$$\Phi_\alpha^{(\mathrm{cyc.})}(\hat{\mathbf{n}},\tau) = \left(\frac{e^{\tau/R}}{R}\right)^{\Delta_{\Phi_\alpha}} \Phi_\alpha^{(\mathrm{flat})}(\mathbf{r}) \tag{4.38}$$

For descendants, certain other factors may be produced, but the conversion factors still scale with the radius of the sphere as $R^{-\Delta}$ where Δ is the scaling dimension of the descendants. For simplicity, hereafter we focus on the equal-time correlators with $\tau = 0$, for which $\Phi_\alpha^{(\mathrm{cyc.})}(\hat{\mathbf{n}}) = R^{-\Delta_{\Phi_\alpha}}\Phi_\alpha^{(\mathrm{flat})}(\mathbf{r})$. The operator with larger system size decays faster when increasing system size.

The simplest local observable is the density operator defined in Eq. (4.10), and its decomposition into angular modes is given in Eqs. (4.12) and (??). From the CFT perspective, the density operators are the superpositions of scaling operators with corresponding quantum numbers, *i.e.*, with the same representation under flavour symmetry and parity.

Take the Ising model as an example. Consider the density operators n^x and n^z with matrix insertion $M = \sigma^x, \sigma^z$. In the leading order, they can be used as UV realisations of CFT operators σ and ϵ.

$$\begin{aligned} n^x(\hat{\mathbf{n}}) &= \lambda_0 + \lambda_\epsilon \epsilon(\hat{\mathbf{n}}) + \lambda_{\partial^\mu\epsilon}\partial^\mu\epsilon(\hat{\mathbf{n}}) + \lambda_{T^{\mu\nu}} T^{\mu\nu}(\hat{\mathbf{n}}) + \ldots & \epsilon_{\mathrm{FS}} &= \frac{n^x - \lambda_0}{\lambda_\epsilon} + \ldots \\ n^z(\hat{\mathbf{n}}) &= \lambda_\sigma \sigma(\hat{\mathbf{n}}) + \lambda_{\partial^\mu\sigma}\partial^\mu\epsilon(\hat{\mathbf{n}}) + \lambda_{\partial^\mu\partial^\nu\sigma}\partial^\mu\partial^\nu\sigma(\hat{\mathbf{n}}) + \ldots & \sigma_{\mathrm{FS}} &= \frac{n^z}{\lambda_\sigma} + \ldots \end{aligned} \tag{4.39}$$

where the coefficients $\lambda_0, \lambda_\epsilon, \lambda_\sigma, \ldots$ are model-dependent and need to be determined, and all the operators on the right hand side are defined on the cylinder.

We first consider the insertion of a single operator $\langle\Phi_1|\Phi_2(\hat{\mathbf{n}})|\Phi_3\rangle$. It helps us produce the OPE coefficients. For the simplest example of three scalars,

$$f_{\Phi_1\Phi_2\Phi_3} = \lim_{r_\infty\to\infty} r_\infty^{-2\Delta_{\Phi_1}} \langle\Phi_1(x_\infty)\Phi_2(x)\Phi_3(0)\rangle_{\mathrm{flat}} = \langle\Phi_1|\Phi_2^{(\mathrm{flat})}(x)|\Phi_3\rangle \tag{4.40}$$

where x_∞ is a point on the sphere with radius r_∞, x is a point on the unit sphere, the states are obtained from acting the operator at the origin point on the vacuum state

$$|\Phi_3\rangle = \Phi_3(0)|0\rangle \tag{4.41}$$

and its Hermitian conjugate is defined as

$$\Phi_1^\dagger(\infty) = (\Phi_1(0))^\dagger = \lim_{r_\infty \to \infty} r_\infty^{2\Delta_{\Phi_1}} \Phi_1(x_\infty), \qquad \langle \Phi_1 | = \langle 0 | \Phi_1^\dagger(\infty) \tag{4.42}$$

After the Weyl transformation from the flat spacetime to the cylinder, we obtain the expression on fuzzy sphere

$$f_{\Phi_1\Phi_2\Phi_3} = R^{\Delta_{\Phi_2}} \langle \Phi_1 | \Phi_2^{(\text{cyl.})}(\hat{\mathbf{n}}) | \Phi_3 \rangle. \tag{4.43}$$

The UV realisation of Φ_2 contains many other operators with different spins. By integrating the correlation function against different spherical harmonics, *i.e.*, take the angular modes of the operator inserted

$$\int \mathrm{d}\hat{\mathbf{n}}\, \bar{Y}_{lm}(\hat{\mathbf{n}}) \langle \Phi_1 | \Phi_2(\hat{\mathbf{n}}) | \Phi_3 \rangle = \langle \Phi_1 | \Phi_{2,lm} | \Phi_3 \rangle, \tag{4.44}$$

we can filter out the subleading contributions with different spin. For the spinning operators, this will also tell us about different OPE structures. By taking $\Phi_3 = \mathbb{I}$, we can recover the two point functions

$$\begin{aligned} &\langle \Phi_2 | \Phi_{2,00} | 0 \rangle = R^{-\Phi_2} \\ &\Phi_2(\hat{\mathbf{n}}) | 0 \rangle = R^{-\Phi_2} \Big[|\Phi_2\rangle + \lambda'_\mu(\hat{\mathbf{n}}) |\partial^\mu \Phi_2\rangle + \lambda''(\hat{\mathbf{n}}) |\Box \Phi_2\rangle + \lambda''_{\mu\nu}(\hat{\mathbf{n}}) |\partial^\mu \partial^\nu \Phi_2\rangle \Big] \end{aligned} \tag{4.45}$$

It is worthnoting acting a primary $\Phi_2(\hat{\mathbf{n}})$ on the vacuum will also produce various descendants in the multiplet.

In the example of Ising CFT, we first use the insertion of a single operator to determine the coefficients $\lambda_0, \lambda_\epsilon, \lambda_\sigma$.

$$\lambda_0 = \frac{1}{\sqrt{4\pi}} \langle 0 | n_{00}^x | 0 \rangle, \quad \lambda_\epsilon = \frac{R^{\Delta_\epsilon}}{\sqrt{4\pi}} \langle \epsilon | n_{00}^x | 0 \rangle, \quad \lambda_\sigma = \frac{R^{\Delta_\sigma}}{\sqrt{4\pi}} \langle \sigma | n_{00}^z | 0 \rangle \tag{4.46}$$

Take the OPE coefficient $f_{\sigma\sigma\epsilon}$ as an example. It can be expressed either as a one point function of σ or ϵ

$$\begin{aligned} f_{\sigma\sigma\epsilon} &= R^{\Delta_\sigma} \langle \epsilon | \sigma(\hat{\mathbf{n}}) | \sigma \rangle = \frac{\langle \epsilon | n_{00}^z | \sigma \rangle}{\langle 0 | n_{00}^z | \sigma \rangle} + \mathcal{O}(R^{-2}) \\ &= R^{\Delta_\epsilon} \langle \sigma | \epsilon(\hat{\mathbf{n}}) | \sigma \rangle = \frac{\langle \sigma | n_{00}^x | \sigma \rangle - \langle 0 | n_{00}^x | 0 \rangle}{\langle \epsilon | n_{00}^x | 0 \rangle} + \mathcal{O}(R^{-(3-\Delta_\epsilon)}) \end{aligned} \tag{4.47}$$

For the first line, the subleading contribution comes from the contribution of the descendant $\Box\sigma$ to n^z_{00}. As $\sigma(\hat{\mathbf{n}})$ scales as $R^{-\Delta_\sigma}$ and $\Box\sigma(\hat{\mathbf{n}})$ as $R^{-\Delta_\sigma-2}$,

$$\begin{aligned}\langle\epsilon|n^z_{00}|\sigma\rangle &= f_{\sigma\sigma\epsilon}\lambda_\sigma R^{-\Delta_\sigma}(1+c_1R^{-2}+\dots)\\ \langle\epsilon|n^z_{00}|\sigma\rangle &= \lambda_\sigma R^{-\Delta_\sigma}(1+c_1'R^{-2}+\dots)\\ \frac{\langle\epsilon|n^z_{00}|\sigma\rangle}{\langle 0|n^z_{00}|\sigma\rangle} &= f_{\sigma\sigma\epsilon}+\mathcal{O}(R^{-2})\end{aligned} \tag{4.48}$$

Here c_1 and c_1' are constant factors that represents the contribution of $\Box\sigma$ and does not scale with system size. Hence, the subleading contribution scales as R^{-2}. For the second line, the subleading contribution comes from the stress tensor $T^{\mu\nu}$. Similarly, the power of the scaling is the difference of the scaling dimension $R^{-(\Delta_{T^{\mu\nu}}-\Delta_\epsilon)}=R^{-(3-\Delta_\epsilon)}$.

We then proceed to the insertion of two operators. This can help us determine up to a four-point function. Through conformal transformation, any four point function can be expressed in the form of

$$\langle\Phi_1|\Phi_2^{(\text{cyl.})}(\hat{\mathbf{n}},\tau)\Phi_3^{(\text{cyl.})}(\hat{\mathbf{z}})|\Phi_4\rangle = \frac{e^{\Delta_{\Phi_2}\tau/R}}{R^{\Delta_{\Phi_2}+\Delta_{\Phi_3}}}\langle\Phi_1^\dagger(\infty)\Phi_2(\mathbf{r})\Phi_3(\hat{\mathbf{z}})\Phi_4(0)\rangle \tag{4.49}$$

where the time-displaced operator can be defined as

$$\Phi_2(\hat{\mathbf{n}},\tau)=e^{-H\tau}\Phi_2(\hat{\mathbf{n}})e^{H\tau} \tag{4.50}$$

As a sanity check, By taking $\Phi_1=\Phi_4=\mathbb{I}$, $\Phi_2=\Phi_3$ and $\tau=0$, the two-point function on the unit sphere is recovered

$$\begin{aligned}\langle 0|\Phi_2^{(\text{cyl.})}(\hat{\mathbf{n}})\Phi_2^{(\text{cyl.})}(\hat{\mathbf{z}})|0\rangle &= R^{-2\Delta_{\Phi_2}}\langle\Phi_2(\hat{\mathbf{n}})\Phi_2(\hat{\mathbf{z}})\rangle\\ &= \frac{1}{R^{2\Delta_{\Phi_2}}|\hat{\mathbf{n}}-\hat{\mathbf{z}}|^{2\Delta_{\Phi_2}}}=\frac{1}{R^{2\Delta_{\Phi_2}}(1-\cos\theta)^{\Delta_{\Phi_2}}}.\end{aligned} \tag{4.51}$$

4.7 Conformal generators

So far, in the conformal group, we know that the rotation and the dilatation is manifest on the fuzzy sphere. The rest, *viz.* translation and SCT, are emergent. In this section, we consider how to express the generators of these emergent symmetries in terms of the microscopic operators.

A general Noether current and corresponding generator of the infinitesimal spacetime transformation $x^\mu \mapsto x^\mu + \epsilon^\mu(x)$ can be expressed in terms of the stress tensor

$$j^\mu_\epsilon(x) = \epsilon^\nu(x) T^\mu{}_\nu(x), \quad Q_\epsilon = \int_\Sigma \mathrm{d}^{d-1}x \sqrt{g} j^0_\epsilon(x) \tag{4.52}$$

where for the second equation, the integral is evaluated on a closed surface Σ. Specifically, for the generators P^μ, K^μ of translation and SCT in the embedded sphere

$$P^\mu = \int \mathrm{d}^2\hat{\mathbf{n}}\, (n^\mu T^0{}_0 + i T^{0\mu}), \quad K^\mu = \int \mathrm{d}^2\hat{\mathbf{n}}\, (n^\mu T^0{}_0 - i T^{0\mu}) \tag{4.53}$$

Hence, the conformal generator $\Lambda^\mu = P^\mu + K^\mu$ is the $l = 1$ component of the Hamiltonian density $\mathcal{H} = T^0{}_0$[4]

$$\Lambda_m = P_m + K_m = \sqrt{\frac{4\pi}{3}} \int \mathrm{d}^2\hat{\mathbf{n}}\, \bar{Y}_{1m}(\hat{\mathbf{n}}) \mathcal{H}(\hat{\mathbf{n}}). \tag{4.54}$$

By acting it on the the states, the number of derivatives is increased or decreased by 1, *e.g.*, for a primary Φ

$$\begin{aligned} \Lambda^\mu |\Phi\rangle &= \text{const.} \times |\partial^\mu \Phi\rangle \\ \Lambda^\mu |\partial_\mu \Phi\rangle &= \text{const.} \times |\Phi\rangle + \text{const.} \times |\partial^\mu \partial^\nu \Phi\rangle + \text{const.} \times |\Box \Phi\rangle \end{aligned} \tag{4.55}$$

The derivation of the expression and the constant factors are calculated and given in Ref. [].

We then need to find the expression for the Hamiltonian density. For example, for Ising model, it is the local density operator and density-density interactions with some full derivatives

$$\mathcal{H}(\hat{\mathbf{n}}) = n_z \left(g_0 + g_1 \nabla^2\right) n_z - h n_x + g_{D,1} \nabla^2 n_x + g_{D,2} \nabla^2 n_z^2 + \ldots \tag{4.56}$$

where $g_{D,i}$ are undetermined constants that does not affect the Hamiltonian $H = \int \mathrm{d}^2\hat{\mathbf{n}}\, \mathcal{H}$. We have only listed a few examples of the allowed full derivatives.

To determine those constants, we consider another strategy by consider all the possible two-fermion and four-fermion operators that are singlet under flavour symmetry and spin-1 under SO(3). We consider the example of Ising CFT. The two-fermion terms include the density operactors

$$n^x_{1m} \quad \text{and} \quad n^0_{1m}.$$

[4]Here the indices μ and m are two equivalent way to express the components

Similar to what we have done for Hamiltonian, the four-fermion operators can be obtained by combining the fermion bilinears Δ_{lm}

$$\sum_{l_1 l_2 m_1 m_2} \tilde{U}_{l_1 l_2} \Delta^\dagger_{l_1 m_1} \Delta_{l_2 m_2} \langle (2s - l_1)(-m_1), (2s - l_2) m_2 | 1 m \rangle \tag{4.57}$$

For $l_1 \in 2\mathbb{Z}$, the spin index in the pairing operator is anti-symmetrised ; For $l_1 \in 2\mathbb{Z} + 1$, the spin index in the pairing operator is symmetrised. Therefore, $l_1 - l_2 \in 2\mathbb{Z}$ for non-zero results. And since $|l_1 - l_2| \leq 1$, we conclude $l_1 = l_2$. so

$$\Lambda_m = \sum_{l m_1 m_2} \tilde{U}_l \Delta^\dagger_{l m_1} \Delta_{l m_2} \begin{pmatrix} 2s - l & 2s - l & 1 \\ -m_1 & m_2 & m \end{pmatrix} + \tilde{h} n^x_{1m} + \tilde{\mu} n^0_{1m} \tag{4.58}$$

Here, $\tilde{U}_l, \tilde{h}, \tilde{\mu}$ are tuning parameters.

After obtaining $\Lambda^\mu = P^\mu + K^\mu$, the separate P^μ and K^μ can be obtained by considering the commutator with the dilatation generator D, which is proportional to the Hamiltonian. As $[D, P^\mu] = P^\mu$ and $[D, K^\mu] = -K^\mu$.

$$\begin{aligned} P^\mu &= \tfrac{1}{2} \Lambda^\mu + \tfrac{1}{2} [D, \Lambda^\mu] \\ K^\mu &= \tfrac{1}{2} \Lambda^\mu - \tfrac{1}{2} [D, \Lambda^\mu]. \end{aligned} \tag{4.59}$$

5 Numerical methods on the fuzzy sphere

In this section, we briefly review the numerical methods supported in FuzzifiED. The numerical methods that have been applied to fuzzy sphere include exact diagonalisation (ED), density matrix renormalisation group (DMRG) and determinant quantum Monte Carlo (DQMC). Among these ED and DMRG have been implemented in FuzzifiED.

5.1 Exact diagonalisation (ED)

Exact diagonalisation (ED) might be the most straightforward method to solve a quantum many-body Hamiltonian. In ED, one construct a many-body basis and write down all the elements of the Hamiltonian matrix on these basis. The eigenstates of the Hamiltonian with the lowest energy can be solved without finding the full eigensystem by Arnoldi or Lanczos algorithm.

Briefly speaking, the Arnoldi algorithm is an iterative method. In each iteration, it constructs an orthonormal basis of the Krylov subspace from an initial vector and finds an approximation to the eigenvector in that basis. This approximate eigenvector is then used as the initial vector for the next iteration. An example of Krylov subspace is spanned by acting the matrix H repeatedly on the initial vector $|i\rangle$

$$\mathcal{K}_r(H, |i\rangle) = \text{span}\left\{|i\rangle, H|i\rangle, H^2|i\rangle, \dots, H^{n-1}|i\rangle\right\} \tag{5.1}$$

The ED calculation can be optimised in several ways. The storage of the Hamiltonian matrix may be compressed by data structure taylored for sparse matrix such as compressed sârse column (CSC). The Hamiltonian matrix is usually block diagonal due to symmetry of the Hamiltonian. The Hilbert space is divided into several sectors that carry different representation under the symmetry, and acting the Hamiltonian on a state in a sector results in a state in the same sector. *E.g.*, in the ED calculation for the Ising model on the fuzzy sphere, the symmetries we can use include two U(1) symmetries, *viz.* the conservation of particle number and the angular momentum in the z-direction, and three $\mathbb{Z}_2$ symmetries, *viz.* the Ising $\mathbb{Z}_2$ flavour symmetry, the particle-hole symmetry and the π-rotation along the y-axis[5].

The ED method enjoy several advantages, including (1) the full knowledge of the eigenstate wavefunction and (2) the ability to access relatively high excited states. However, despite these optimisations, the dimension of the Hilbert space scales exponentially with the number of orbitals. This results in exponentially growing space and time complexity. *E.g.*, for the Ising model on the fuzzy sphere, for $N_m = 12$, the dimension of Hilbert space $\dim \mathcal{H} = 1.6 \times 10^4$ and the number of elements in the Hamiltonian is $N_{\text{el}} = 6.5 \times 10^5$; for $N_m = 14$, the number have already grown to $\dim \mathcal{H} = 1.8 \times 10^5$ and $N_{\text{el}} = 1.1 \times 10^7$, which translates to a memory demand of 0.2 gigabytes.

In FuzzifiED, we use the celebrated Fortran library Arpack to perform the Arnoldi algorithm.

[5] So far, FuzzifiED only supports U(1) and $\mathbb{Z}_p$ symmetries. We are still trying to implement non-abelian symmetries.

5.2 Density matrix renormalisation group (DMRG)

To overcome the size limit of ED, density matrix renormalisation group (DMRG) is a powerful method calculating the ground state of a quasi-one-dimensional system. It has been first first invented by S. R. White as an improvement to the numerical renormalisation group (NRG) used in the Kondo problem. Since its proposal, it has been proven powerful in various problems in condensed matter physics, such as the static and dynamic properties of one-dimensional models such as the Heisenberg, t–-J and Hubbard models. Later, Schollwöck has discovered a new point of view that implements the DMRG in the language of matrix product states (MPS).

Briefly speaking, in this language, DMRG is a variational method that optimises the fidelity between the exact ground state and the variational MPS. To find the lowest excited state, one need to add projection $|0\rangle\langle 0|$ of the ground state $|0\rangle$ to the Hamiltonian by hand. Due to the complexity of this process, the higher excited state are difficult to access.

Although the fuzzy sphere deals with $(2+1)$-dimensional quantum systems, the basis of lowest Landau level provides a natural way to express it as a quasi-1d problem. Therefore, DMRG has been a powerful numerical method for fuzzy sphere. However, like other $(2+1)$d models, the DMRG on fuzzy sphere also suffers from the divergence of the required maximal bond dimension with system size. One should thus be careful with checking the convergence of the results when doing DMRG.

In FuzzifiED, we use the ITensor Package in Julia to perform the DMRG calculations.

Part II

Numerical calculation with FuzzifiED

6 Getting started with FuzzifiED : installation and usage

The package FuzzifiED is implemented in the Julia language[6]. Some useful links are given in Table 1. To install the package, run the following command in the Julia REPL (read-eval-print loop).

```
using Pkg ; Pkg.add("FuzzifiED")
```

To use the package, include at the start of the Julia script

```
using FuzzifiED
```

To obtain the documentation for an interface, type '`?`' followed by the keyword in the Julia REPL, *e.g.*, '`?Confs`'.

Installation for Julia	https://julialang.org/downloads
Homepage	https://www.fuzzified.world
Documentation	https://docs.fuzzified.world
Julia source code	https://github.com/FuzzifiED/FuzzifiED.jl
JLL wrapper	https://github.com/FuzzifiED/FuzzifiED_jll.jl
Fortran source code	https://github.com/FuzzifiED/FuzzifiED_Fortran
Registry of the package	https://juliahub.com/ui/Packages/General/FuzzifiED

Table 1. Some useful links.

7 Exact diagonalisation (ED) with FuzzifiED

In this section, we briefly describe the procedure for exact diagonalisation (ED) calculation and give an instruction for using FuzzifiED for ED.

[6] The most intensive core functionalities are written in Fortran and wrapped by Julia interfaces, but the users are not required to have any interaction or knowledge with the Fortran part of the source code.

Practically, the ED calculation can be divided into 4 steps, which will be described in detail in the following sections.

1. Construct a many-body basis that respect a given set of quantum numbers (Sections 7.2 and 7.3). Specifically, in FuzzifiED we support quantum numbers of commuting U(1) or discrete $\mathbb{Z}_p$ symmetries.

2. Construct the sparse matrix corresponding to the Hamiltonian in the basis above (Sections 7.4 and 7.5).

3. Find the lowest eigenstates and their corresponding eigenenergies of the sparse matrix (Section 7.6).

4. Making measurements on the eigenstates (Sections 7.7, 7.8 and 7.9). This including the total angular momentum, density operators, entanglement, *etc.*

To demonstrate the usage of FuzzifiED interfaces, in this section, we use an example that calculates the eigenstates for the Ising model on fuzzy sphere. Specifically, it

1. calculates the lowest eigenstates in the symmetry sector $L^z = 0$ and $(\mathcal{P}, \mathcal{Z}, \mathcal{R}) = (+,+,+)$,

2. measures their total angular momenta, and

3. calcultes the OPE coefficient $f_{\sigma\sigma\epsilon} = \langle\sigma|n^z_{00}|\epsilon\rangle/\langle\sigma|n^z_{00}|0\rangle$.

The full code is collected in Appendices B.1 and B.2.

7.1 Setup

Before starting the calculation, we need to input the setup for the system, including the number of flavours N_f, orbitals N_m and sites N_o

- A 'flavour' is labelled by f. The number of flavours is N_f.
- An 'orbital' is specified by the magnetic quantum number labelled by m. The number of orbitals is $N_m = 2s + 1$.
- A 'site' is specific by both the flavour and the orbital index $o = (f, m)$. The number of sites is $N_o = N_m N_f$. In practice, we label the sites with an integer from 1 to N_o. We store the sites in an ascending order of first m and then f : $o = (m + s)N_f + f$.

In the example of Ising model with $s = 5.5$,

```
nm = 12
```

```
nf = 2
no = nm * nf
```

FuzzifiED also provides three environment parameters that defines how FuzzifiED works.

- `FuzzifiED.ElementType` — set the default type of the operator elements, either `ComplexF64` or `Float64`.
- `FuzzifiED.NumThreads` — an integer to define how many threads OpenMP uses.
- `FuzzifiED.SilentStd` — a flag to determine whether logs of the FuzzifiED functions should be turned off.

7.2 Constructing the configurations

The first step for the ED calculation is to construct the basis tha respects the symmetries of the Hamiltonian. This is divided into two steps : (1) generate the 'configurations' that carry the diagonal quantum numbers, and (2) generate the 'basis' that also carry the off-diagonal quantum numbers (under discrete transformations). The '*configurations*' are the collection of states that can be written as direct product of occupied $|1\rangle$ or empty $|0\rangle$ on each site and carries certain diagonal quantum numbers (QNDiag).

The QNDiags supported by FuzzifiED are the charges of U(1) or $\mathbb{Z}_p$ symmetry in the form of

$$\begin{aligned} Q &= \sum_o q_o n_o && \text{U(1) symmetry} \\ Q &= \sum_o q_o n_o \mod p && \mathbb{Z}_p \text{ symmetry} \end{aligned} \tag{7.1}$$

where $n_o = c_o^\dagger c_o$ is the particle number on each site, and q_o is the charge that each orbital carries. FuzzifiED restricts q_o to be integer-valued. In FuzzifiED, the QNDiags are recorded in the mutable type `QNDiag`.

`QNDiag` — Type

The type contains the fields

- `name :: String` — the name of the diagonal quantum number[7].
- `charge :: Vector{Int64}` — the symmetry charge q_o of each site.

[7] The name is only needed for conversion into quantum numbers in ITensor.

- `modul :: Vector{Int64}` — the modulus p, set to 1 for U(1) QNDiags.

and can be initialised by the method

`QNDiag([name :: String, ]charge :: Vector{Int64}[, modul :: Int64])`

where the arguments in the brackets are facultative.

Several useful QNDiags are built-in[8]

- `GetNeQNDiag(no :: Int64)` — the numberof electrons ;
- `GetLz2QNDiag(nm :: Int64, nf :: Int64)` — twice the angular momentum $2L_z$;
- `GetFlavQNDiag(nm :: Int64, nf :: Int64, qf :: Union{Dict{Int64, Int64}, Vector{Int64}}[, id :: Int64 = 1, modul :: Int64 = 1])`[9] — a linear combination of number of electrons in each flavour $Q = \sum_f q_f n_f$, where $\{q_f\}$ is stored in `qf` in the format of either an array or a dictionary.
- `GetZnfChargeQNDiag(nm :: Int64, nf :: Int64)` — a $\mathbb{Z}_{N_f}$-charge $Q = \sum_{f=0}^{N_f-1} f n_f$ mod N_f.
- `GetPinOrbQNDiag(no :: Int64, pin_o :: Vector{Int64}[, id :: Int64 = 1])` — the number of electrons in the subset of sites `pin_o`. This QNDiag is useful for defects and boundaries.

The collection of configurations is generated from the QNDiags. It is recorded in the mutable type `Confs`.

`Confs` — Type

The mutable type contains the fields

- `no :: Int64` — the number of sites.
- `ncf :: Int64` — the number of configurations.
- `conf :: Vector{Int64}` is an array of length `ncf` containing all the configurations.
- `nor :: Int64`, `lid :: Vector{Int64}` and `rid :: Vector{Int64}` contain the information of Lin table that is used to inversely look up the index from the configuration.

and can be constructed by the method

[8] For a more detailed description of the interfaces, refer to the documentation at https://docs.fuzzified.world.

[9] In Julia, `Union{A,B}` means that both the types A and B are acceptable. *E.g.*, for $Q = n_{f=1} - n_{f=3}$ in a 4-flavour system, both `qf = [1, 0, -1, 0]` or `qf = Dict(1 => 1, 3 => -3)` are acceptable.

```
Confs(no :: Int64, secd :: Vector{Int64}, qnd :: Vector{QNDiag})
```

where `qnd` is the array of QNDiags, and `secd` is the array of charges Q of each QNDiag[10].

Here each configuration is stored as a binary number with N_o bits. If the o-th site in the configuration is occupied, the $(o-1)$-th bit of the configuration is 1; if the site is empty, then the bit is 0. Besides the storation of the configuration, we also need a reverse look-up process that returns the index from the binary string. This is realised by a Lin table stored in `lid` and `rid`. The details are given in Appendix A.1.

In the example of Ising model Eq. (4.20), there are two QNDiags, *viz.* the particle number and the angular momentum.

$$\begin{aligned} Q_1 &= N_e, & q_{1,mf} &= 1 \\ Q_2 &= 2L_z, & q_{2,mf} &= 2m \end{aligned} \tag{7.2}$$

The full code to generating the configurations in the $L_z = 0$ sector is

```
qnd = [
    QNDiag(fill(1, no)),
    QNDiag([ 2 * m - nm - 1 for m = 1 : nm for f = 1 : nf ])
]
cfs = Confs(no, [nm, 0], qnd)
```

Alternatively, using the built-in models,

```
qnd = [
    GetNeQNDiag(no),
    GetLz2QNDiag(nm, nf)
]
cfs = Confs(no, [nm, 0], qnd)
```

[10] In general, many methods in FuzzifiED admits keyword arguments `num_th :: Int64` that specifies the number of threads and `disp_std :: Bool` that specifies whether or not the log shall be displayed. Hereafter, we will omit these two arguments.

7.3 Constructing the basis

Having constructed the configurations, we now construct the basis of the Hilbert space. The '*basis*' is the collection of states that are linear combinations of the configuration carrying certain diagonal and $\mathbb{Z}_p$ off-diagonal quantum numbers (QNOffd).

The QNOffds supported by FuzzifiED are the $\mathbb{Z}_p$ symmetry that are in the form of

$$\mathcal{Z}: \ c_o \to \alpha_o^* c_{\pi_o}^{(p_o)}, \quad c_o^\dagger \to \alpha_o c_{\pi_o}^{(1-p_o)} \tag{7.3}$$

where we use a notation $c^{(1)} = c^\dagger$ and $c^{(0)} = c$ for convenience, π_o is a permutation of the sites $1, ... N_o$, α_o is a coefficient, and p_o specified whether or not particle-hole transformation is performed for the site. Note that one must guarentee that all these transformations commute with each other and also commute with the diagonal quantum numbers. In FuzzifiED, the QNOffds are recorded in the mutable type `QNOffd`.

`QNOffd` — Type

The mutable type contains the fields

- `perm :: Vector{Int64}` — a length-N_o array that records the permutation π_o.
- `ph :: Vector{Int64}` — a length-N_o array that records p_o to determine whether or not to perform a particle-hole transformation.
- `fac :: Vector{ComplexF64}` — a length-N_o array that records the factor α_o in the transformation.
- `cyc :: Int64` — the cycle p.

It can be initialised by the methods

```
QNOffd(perm :: Vector{Int64}[, ph :: Vector{Int64}][, fac :: Vector{
    ComplexF64}][, cyc :: Int64])
QNOffd(perm :: Vector{Int64}, ph_q :: Bool[, fac :: Vector{ComplexF64}])
```

By default `ph` is set all 0, `fac` is set to all 1 and `cyc` is set to 2. If `ph_q` is set to be true, `ph` is set to all 1.

Several useful QNOffds are built-in

- `GetParityQNOffd(nm :: Int64, nf :: Int64[, permf, fac])` — the particle-hole transformation $\mathcal{P}: c^\dagger_{mf} \mapsto \alpha_f c_{m\pi_f}$, with the permutation of flavours π_f and the factors α_f stored in `permf` and `fac` as either an array or a dictionary.

- `GetFlavPermQNOffd(nm :: Int64, nf :: Int64, permf, fac][, cyc :: Int64])` — the flavour permutation transformation $\mathcal{Z}: c^\dagger_{mf} \mapsto \alpha_f c^\dagger_{m\pi_f}$, with the permutation of flavours π_f and the factors α_f stored in `permf` and `fac` as either an array or a dictionary, and the cycle stored in `cyc`.

- `GetRotyQNOffd(nm :: Int64, nf :: Int64)` — the π-rotation with respect to the y-axis $\mathcal{R}: c^\dagger_{mf} \mapsto (-1)^{m+s} c^\dagger_{(-m)f}$

After implementing the QNOffds, a state in the new basis should look like

$$|I\rangle = \lambda_{i_{I1}}|i_{I1}\rangle + \lambda_{i_{I2}}|i_{I2}\rangle + \cdots + \lambda_{i_{Im_I}}|i_{Im_I}\rangle \tag{7.4}$$

where the $|i\rangle$'s are configurations, and $|I\rangle$ is a linear combination of them. This process can be regarded as organising the configurations into groups of size m_I. In FuzzifiED, the basis $\{|I\rangle\}$ is recorded in the mutable type `Basis`.

`Basis` — Type

The mutable type contains the fields

- `cfs :: Confs` — the configurations $\{|i\rangle\}$ that respect the QNDiags.

- `dim :: Int64` — the dimension of the basis.

- `szz :: Int64` — the maximum size $\max m_I$ of groups.

- `cfgr :: Vector{Int64}` — an array of length `cfs.ncf` and records which group $|I\rangle$ each configuration $|i\rangle$ belong to.

- `cffac :: Vector{ComplexF64}` — an array of length `cfs.ncf` and records the coefficients λ_i of each configuration.

- `grel :: Matrix{Int64}` — a `szz`$\times$`dim` matrix that records the configurations in each group $|i_{Ik}\rangle$ ($k = 1, \ldots, m_I$).

- `grsz :: Vector{Int64}` — an array of length `dim` that records the size m_I of each group.

It can be constructed by the methods

```
Basis(cfs :: Confs, secf :: Vector{ComplexF64}, qnf :: Vector{QNOffd})
Basis(cfs :: Confs)
```

where `secf` records the eigenvalue of each transformation, typically in the form $e^{i2\pi q/p}$ where p is the cycle and q is the $\mathbb{Z}_p$ charge.

In the example of Ising model Eq. (4.20), There are three $\mathbb{Z}_2$ symmetries, *viz.* the particle-hole transformation $\mathcal{P}$, the π-rotation along the y-axis $\mathcal{R}_y$, and the flavour (Ising) symmetry $\mathcal{Z}$

$$\begin{aligned} \mathcal{P}: \ & c^\dagger_{\sigma m} \mapsto \sigma c_{-\sigma,m} \\ \mathcal{Z}: \ & c^\dagger_{\sigma m} \mapsto c^\dagger_{-\sigma,m} \\ \mathcal{R}_y: \ & c^\dagger_{\sigma m} \mapsto c^\dagger_{\sigma,-m} \end{aligned} \tag{7.5}$$

The code to generate the basis in the all-positive sector is

```
qnf = [
    QNOffd([ isodd(o) ? o + 1 : o - 1 for o = 1 : no], true, ComplexF64[
        isodd(o) ? -1 : 1 for o = 1 : no]),
    QNOffd([ isodd(o) ? o + 1 : o - 1 for o = 1 : no]),
    QNOffd([ isodd(o) ? no - o : no + 2 - o for o = 1 : no], ComplexF64(-1)
        .^ (collect(0 : nm * nf - 1) .÷ nf))
]
bs = Basis(cfs, [1, 1, 1], qnf)
```

Alternatively, using the built-in functions

```
qnf = [
    GetParityQNOffd(nm, 2, [2, 1], [-1, 1]),
    GetFlavPermQNOffd(nm, 2, [2, 1]),
    GetRotyQNOffd(nm, 2)
]
bs = Basis(cfs, [1, 1, 1], qnf)
```

7.4 Recording the many-body operator terms

Having constructed the basis, we now construct the many-body operators. A general many-body operator can be written as

$$\mathcal{O} = \sum_{t=1}^{N_t} U_t c_{o_{t1}}^{(p_{t1})} c_{o_{t2}}^{(p_{t2})} ... c_{o_{tl_t}}^{(p_{tl_t})} \tag{7.6}$$

where $c^{(0)} = c$ and $c^{(1)} = c^\dagger$. In FuzzifiED, this is recorded as an array of `Term`, and each `Term` records the building block $U c_{o_1}^{(p_1)} c_{o_2}^{(p_2)} ... c_{o_l}^{(p_l)}$.

`Term` — Type

The mutable type contains the fields

- `coeff :: ComplexF64` — the coefficient U.
- `cstr :: Vector{Int64}` — a length-$2l$ array $\{p_1, o_1, p_2, o_2, ...p_l, o_l\}$ recording the operator string.

It can be initialised by the method

```
Term(coeff :: ComplexF64, cstr :: Vector{Int64})
```

The addition and multiplication of terms are supported, and the terms can be simplified by the method

```
SimplifyTerms(tms :: Vector{Term})
```

After the simplification, the resulting terms satisfy

1. Each term is normal ordered — the creation operator is in front of the annihilation operator ; the site index of the creation operators are in ascending order and the annihilation operators in descending order.
2. Like terms are combined, and terms with zero coefficients are removed.

In FuzzifiED, several useful operator terms are built-in :

- `GetDenIntTerms(nm :: Int64, nf :: Int64[, ps_pot :: Vector{<:Number}][, mat_a :: Matrix{<:Number}[, mat_b :: Matrix{<:Number}]]):: Terms` — the normal-ordered density-density interaction term in the Hamiltonian

$$\sum_{l\{m_i f_i\}} U_l C^l_{\{m_i\}} M^A_{f_1 f_4} M^B_{f_2 f_3} c^\dagger_{m_1 f_1} c^\dagger_{m_2 f_2} c_{m_3 f_3} c_{m_4 f_4} \tag{7.7}$$

where $C^l_{\{m_i\}}$ is given in Eq. (4.25).

- `GetPairIntTerms(nm :: Int64, nf :: Int64, ps_pot :: Vector{<:Number}, mat_a :: Matrix{<:Number}[, mat_b :: Matrix{<:Number}])` — the normal-ordered pair-pair interaction term in the Hamiltonian

$$\sum_{l\{m_i f_i\}} U_l C^l_{\{m_i\}} M^A_{f_1 f_2} M^B_{f_3 f_4} c^\dagger_{m_1 f_1} c^\dagger_{m_2 f_2} c_{m_3 f_3} c_{m_4 f_4}. \tag{7.8}$$

- `GetPolTerms(nm :: Int64, nf :: Int64[, mat :: Matrix{<:Number}])` — the polarisation term in the Hamiltonian

$$\sum_{m f_1 f_2} c^\dagger_{m f_1} M_{f_1 f_2} c_{m f_2}. \tag{7.9}$$

- `GetL2Terms(nm :: Int64, nf :: Int64)` — the total angular momentum.
- `GetC2Terms(nm :: Int64, nf :: Int64, mat_gen :: Vector{Matrix{<:Number}})` — the quadratic casimir

$$C_2 = \sum_{imm'\{f_i\}} \frac{(c^\dagger_{m f_1} G^i_{f_1 f_2} c_{m f_2})(c^\dagger_{m' f_3} (G^i_{f_3 f_4})^\dagger c_{m' f_4})}{\mathrm{tr}\, G_i^\dagger G_i} \tag{7.10}$$

where G^i are the generator matrices.

In the example of Ising model, the full code that records the Hamiltonian Eq. (4.20) is

```
using WignerSymbols
ps_pot = [ 4.75, 1.0 ] * 2.0
fld_h = 3.16
tms_hmt = Term[]
for m1 = 0 : nm - 1
    f1 = 0
    o1 = m1 * nf + f1 + 1
    m1r = m1 - s
    for m2 = 0 : nm - 1
        f2 = 1
        o2 = m2 * nf + f2 + 1
        m2r = m2 - s
        for m3 = 0 : nm - 1
```

```
            f3 = 1
            o3 = m3 * nf + f3 + 1
            m3r = m3 - s
            m4 = m1 + m2 - m3
            (m4 < 0 || m4 >= nm) && continue
            f4 = 0
            o4 = m4 * nf + f4 + 1
            m4r = m4 - s
            val = 0
            for l in eachindex(ps_pot)
                (abs(m1r + m2r) > nm - 1 || abs(m3r + m4r) > nm - 1) && break
                val += ps_pot[l] * clebschgordan(s, m1r, s, m2r, nm - 1) *
                    clebschgordan(s, m4r, s, m3r, nm - 1)
            end
            tms_hmt += Terms(val, [1, o1, 1, o2, 0, o3, 0, o4])
        end
    end
    o1x = o1 + 1
    tms_hmt += Terms(-fld_h, [1, o1, 0, o1x])
    tms_hmt += Terms(-fld_h, [1, o1x, 0, o1])
end
```

Alternatively, using the built-in functions

```
sg1 = [ 1 0 ; 0 0 ]
sg2 = [ 0 0 ; 0 1 ]
sgx = [ 0 1 ; 1 0 ]
sgz = [ 1 0 ; 0 -1]
ps_pot = [ 4.75, 1.0 ] * 2.0
fld_h = 3.16
tms_hmt = SimplifyTerms(
    GetDenIntTerms(nm, 2, ps_pot, sg1, sg2)
    - fld_h * GetPolTerms(nm, 2, sgx)
)
```

We also need to construct the total angular momentum. It is defined as

$$L^2 = L^+L^- + (L^z)^2 - L^z, \tag{7.11}$$

as c_m carries the SO(3) spin-s representation,

$$L^z = \sum_{mf} m c^\dagger_m c_m, \quad L^\pm = \sum_{mf} \sqrt{(s \mp m)(s \pm m + 1)} c^\dagger_{m\pm1} c_m \tag{7.12}$$

we can first construt its building blocks and use the addition and multiplication of the terms

```
tms_lz =
    [ begin m = div(o - 1, nf)
        Term(m - s, [1, o, 0, o])
    end for o = 1 : no ]
tms_lp =
    [ begin m = div(o - 1, nf)
        Term(sqrt(m * (nm - m)), [1, o, 0, o - nf])
    end for o = nf + 1 : no ]
tms_lm = tms_lp'
tms_l2 = SimplifyTerms(tms_lz * tms_lz - tms_lz + tms_lp * tms_lm)
```

Alternatively, using the built-in functions,

```
tms_l2 = GetL2Terms(nm, nf)
```

7.5 Generating sparse matrix

Having gotten the terms in the many-body operator, we now need to generate the matrix elements given the initial and final basis and find its eigenstates.

In FuzzifiED, the mutable type `Operator` records the terms together with information about its symmetry and the basis of the state it acts on and the basis of the resulting state.

`Operator` — Type

The type can be initialised with the method

```
Operator(bsd :: Basis[, bsf :: Basis], terms :: Vector{Term} ; red_q :: Int64
    , sym_q :: Int64)
```

where the arguments

- `bsd :: Basis` — the basis of the initial state.
- `bsf :: Basis` — the basis of the final state. Facultative, the same as `bsd` by default.
- `terms :: Vector{Term}` — the terms.

- `red_q :: Int64` — a flag that records whether or not the conversion to a sparse matrix can be simplified : if `bsd` and `bsf` have exactly the same set of quantum numbers, and the operator fully respects the symmetries, then `red_q = 1` ; otherwise `red_q = 0` ; Facultative, if `bsf` is not given, 1 by default, otherwise 0 by default.
- `sym_q :: Int64` — the symmetry of the operator : if its corresponding matrix is Hermitian, then `sym_q = 1` ; if it is symmetric, then `sym_q = 2` ; otherwise `sym_q = 0`. Facultative, if `bsf` is not given, 1 by default, otherwise 0 by default.

The sparse matrix is recorded in the format of compressed sparse column (CSC), which is described in detail in Appendix A.2. In FuzzifiED, the sparse matrix is stored in the mutable type `OpMat{T}`

`OpMat{T}` — Type

---

where `T` is the type of the elements, it can either be `ComplexF64` or `Float64`. It contines the fields

- `dimd :: Int64`, `dimf :: Int64`, `nel :: Int64`, `symq :: Int64` — Parameters of the sparse matrix, the number of columns, rows, elements and the symmetry of matrix, respectively.
- `colptr :: Vector{Int64}` with length `dimd :: Int64 + 1`.
- `rowid :: Vector{Int64}` with length `nel`.
- `elval :: Vector{T}` with length `nel`.

It can be generated from the method

```
OpMat[{T}](op :: Operator)
```

7.6 Finding eigenstates

After generating the sparse matrix, the method `GetEigensystem` uses the Fortran Arpack package to calculate its lowest eigenstates.

`GetEigensystem` — Method

```
GetEigensystem(mat :: OpMat{T}, nst :: Int64 ; tol :: Float64, ncv :: Int64,
    initvec :: Vector{T}) where T <: Union{ComplexF64, Float64} :: Tuple{
    Vector{T}, Matrix{T}}
```

The arguments are

- `mat :: OpMat{T}` — the matrix.
- `nst :: Int64` — the number of eigenstates to be calculated.
- `tol :: Float64` — the tolerence for the Arpack process. The default value is 10^{-8}.
- `ncv :: Int64` — an auxiliary parameter needed in the Arpack process. The default value is `max(2 * nst, nst + 10)`.
- `initvec :: Vector{T}` — the initial vector. If empty, a random initialisation shall be used. Facultative, empty by default.

The output include two items

- A length-`nst` array recording the eigenvalues, and
- A `dimd`$\times$`nst` matrix where every column records an eigenstate.

In the example of Ising model, the full code to calculate the lowest $N_{\text{st}} = 10$ eigenstates from the basis and the terms is

```
nst = 10
hmt = Operator(bs, tms_hmt)
hmt_mat = OpMat(hmt)
enrg, st = GetEigensystem(hmt_mat, nst)
```

7.7 Inner product of states, operators and transformations

Having obtained the eigenstates, we need to make measurements on it. The simplest kind of measurements is the inner product of a many body operator with two states $\langle j|\mathcal{O}|i\rangle$. FuzzifiED supports the inner product and vector product of `Operator` and `OpMat{T}` with vectors that represent the state

```
(op :: Operator) * (st_d :: Vector{T}) :: Vector{T}
(mat :: OpMat{T}) * (st_d :: Vector{T}) :: Vector{T}
(st_f :: Vector{T}) * (op :: Operator) * (st_d :: Vector{T}) :: T
(st_f :: Vector{T}) * (mat :: OpMat{T}) * (st_d :: Vector{T}) :: T
```

For example, the code to measure the angular momenta of each state is

```
tms_l2 = GetL2Terms(nm, 2)
l2 = Operator(bs, tms_l2)
```

```
l2_mat = OpMat(l2)
l2_val = [ st[:, i]' * l2_mat * st[:, i] for i in eachindex(enrg)]
```

One might also need to act transformations on the state $\mathcal{Z}|i\rangle$. In FuzzifiED, the mutable type `Transf` records the transformation together with the basis of the initial and final states

---

`Transf` — Type

The type can be initialised from a `QNOffd` by

```
Transf(bsd :: Basis[, bsf :: Basis], qnf :: QNOffd)
```

where the arguments

- `bsd :: Basis` — the basis of the initial state.
- `bsf :: Basis` — the basis of the final state. Facultative, the same as `bsd` by default.
- `qnf :: QNOffd` — records the transformation $c_o \to \alpha_o^* c_{\pi_o}^{(p_o)}$.

It can act on a state by

```
(trs :: Transf) * (st_d :: Vector{T}) :: Vector{T}
```

7.8 Measuring local observables

Local observables are a kind of particularly useful operators on fuzzy sphere. Their value at a point on the sphere can be decomposed into spherical components, and the multiplication of the components follows the triple integral formula of monopole spherical harmonics

$$\begin{aligned}\mathcal{O}(\hat{\mathbf{n}}) &= \sum_{lm} Y_{lm}^{(s)}(\hat{\mathbf{n}})\mathcal{O}_{lm}\\ (\mathcal{O}_1\mathcal{O}_2)_{lm} &= \sum_{l_1 l_2 m_1 m_2} (\mathcal{O}_1)_{l_1 m_1}(\mathcal{O}_2)_{l_2 m_2}\\ &\qquad \times (-1)^{s+m}\sqrt{\frac{(2l_1+1)(2l_2+1)(2l_3+1)}{4\pi}}\begin{pmatrix} l_1 & l_2 & l \\ m_1 & m_2 & -m\end{pmatrix}\begin{pmatrix} l_1 & l_2 & l \\ -s_1 & -s_2 & s\end{pmatrix}\end{aligned} \tag{7.13}$$

In FuzzifiED, they are stored in the type `SphereObs`.

`SphereObs` — Type

The type contains the fields

- `s2 :: Int64` and `l2m :: Int64` — is twice the spin $2s$ and twice the maximal angular momentum $2l_{\max}$ of the observable.
- `get_comp :: Function` — a function that sends the component specified by a tuple of integers $(2l, 2m)$ to a list of terms that specifies the expression of the component.
- `stored_q :: Bool` — a boolean that specifies whether or not the components of the observable is stored.
- `comps :: Dict{Tuple{Int64, Int64}, Terms}` — each component of the observable stores in the format of a dictionary whose keys are the pairs of integers $(2l, 2m)$ and values are the lists of terms that specifies the expression of the component.

and can be initialised by the methods

```
SphereObs(s2 :: Int64, l2m :: Int64, get_comp :: Function)
SphereObs(s2 :: Int64, l2m :: Int64, comps :: Dict)
```

Their adjoint, addition and multiplication are supported. The related functions are

- `StoreComps(obs :: SphereObs)` calculates and stores each component of the observable.
- `Laplacian(obs :: SphereObs)` takes the Laplacian of an observable.
- `GetComponent(obs :: SphereObs, l :: Number, m :: Number)` returns a spherical component of the observable $\mathcal{O}_{lm}$.
- `GetPointValue(obs :: SphereObs, theta :: Float64, phi :: Float64)` returns an observable at one point $\mathcal{O}(\hat{\mathbf{n}})$.

Several important types of spherical observables are built-in in FuzzifiED

- `GetElectronObs` returns an electron annihilation operator $\psi_f(\hat{\mathbf{n}})$.
- `GetDensityObs` returns the density operator $n_M(\hat{\mathbf{n}}) = \sum_{ff'} \psi_f^\dagger(\hat{\mathbf{n}}) M_{ff'} \psi_{f'}(\hat{\mathbf{n}})$.
- `GetPairingObs` returns the pair operator $\Delta_M(\hat{\mathbf{n}}) = \sum_{ff'} \psi_f(\hat{\mathbf{n}}) M_{ff'} \psi_{f'}$.

In the example of Ising model, to calculate the OPE coefficient $f_{\sigma\sigma\epsilon} = \langle\sigma|n_{00}^z|\epsilon\rangle / \langle\sigma|n_{00}^z|0\rangle$, one need to first calculate the eigenstates in the $\mathbb{Z}_2$-odd sector

```
bs1 = Basis(cfs, [1, -1, 1], qnf)
hmt1 = Operator(bs1, bs1, tms_hmt ; red_q = 1, sym_q = 1)
hmt_mat1 = OpMat(hmt1)
enrg1, st1 = GetEigensystem(hmt_mat1, 10)
```

```
stI = st[:, 1]
ste = st[:, 2]
sts = st1[:, 1]
```

and then construct the density operator

```
obs_nz = GetDensityObs(nm, 2, sgz)
tms_nz00 = SimplifyTerms(GetComponent(obs_nz, 0.0, 0.0))
nz00 = Operator(bs, bs1, tms_nz00 ; red_q = 1)
f_sse = abs((sts' * nz00 * ste) / (sts' * nz00 * stI))
```

Besides the spherical observable, we also provide a type `AngModes` that superposes under the rule of angular momentum superposition instead of spherical harmonics triple integral

$$(\mathcal{A}_1\mathcal{A}_2)_{lm} = \sum_{l_1m_1l_2m_2} (\mathcal{A}_1)_{l_1m_1}(\mathcal{A}_2)_{l_2m_2}\langle l_1m_1l_2m_2|lm\rangle. \tag{7.14}$$

The interfaces are similar.

7.9 Measuring the entanglement

A non-local quantity that bears particular significance is the entanglement. To calculate the entanglement, we divide the sphere into two parts A and B. The reduced density matrix of part A is obtained by tracing the density matrix over the part B

$$\rho_A(\Psi) = \mathrm{tr}_B\,|\Psi\rangle\langle\Psi| \tag{7.15}$$

The entanglement entropy is $S = -\mathrm{tr}\,\rho_A \log \rho_A$ and the entanglement spectrum is the collection of eigenvalues of ρ_A taken negative logarithm.

The detail of the calculation is given in Ref. [30]. Here we only sketch the process. The creation operator in each orbital is divided into the creation on A part and the creation on B part.

$$c_o^\dagger = \alpha_o c_{o,A}^\dagger + \beta_m c_{o,B}^\dagger \tag{7.16}$$

where $|\alpha_o|^2 + |\beta_o|^2 = 1$. For the cut in orbital space m_c,

$$\alpha_{mf} = \Theta(m_c - m)$$

where Θ is the Heaviside function ; for the cut in real space along latitude circle θ_c,

$$\alpha_{mf} = \mathrm{B}_{\cos^2\theta_c/2}(s-m+1, s+m+1)^{1/2}$$

where B is the incomplete beta function.

To calculate the reduced density matrix, we decompose the state into the direct-product basis of two subsystems

$$|\Psi\rangle = \sum_{K_0} v_{K_0}|K_0\rangle = \sum_{I_A J_B} M_{I_A J_B}|I_A\rangle|J_B\rangle \tag{7.17}$$

where the indices $K_0 \in \mathcal{H}, I_A \in \mathcal{H}_{\mathcal{A}}, J_B \in \mathcal{H}_{\mathcal{B}}$ are in the overall Hilbert space and the Hilbert space of subsystem A and B. The density matrix is then

$$\rho_A = \mathbf{M}\mathbf{M}^\dagger \tag{7.18}$$

and the entanglement spectrum can be obtained from the SVD decomposition of the $\mathbf{M}$ matrix. Like the Hamiltonian, the $\mathbf{M}$ matrix is block diagonal, and each block carries different quantum numbers of the Hilbert spaces of A and B subsystem [11].

In FuzzifiED, the decomposition of states into matrix $M_{I_A J_B}$ is done by the funciton `StateDecompMat` , and the calculation of entanglement spectrum is done by the funciton `GetEntSpec`

`GetEntSpec` — Function

```
GetEntSpec(st :: Vector{<:Number}, bs0 :: Basis, secd_lst :: Vector{Vector{
    Vector{Int64}}}, secf_lst :: Vector{Vector{Vector{<:Number}}} ; qnd_a ::
    Vector{QNDiag}[, qnd_b :: Vector{QNDiag}], qnf_a :: Vector{QNOffd}[, qnf_b
     :: Vector{QNOffd}], amp_oa :: Vector{<:Number}[, amp_ob :: Vector{<:
    Number}]) :: Dict{@NamedTuple{secd_a, secf_a, secd_b, secf_b}, Vector{
    Float64}}
```

The arguments are

- `st :: Vector{<:Number}` — the state to be decomposed into direct-product basis of two subsystems.

[11] The M_{IJ} and α_o in our convention is equivalent to $\mathcal{F}_{m,A}$ and $R^A_{\mu\nu}$ in the convension of Ref. [30], the conversions are $\alpha_{mf} = \sqrt{\mathcal{F}_{m,A}}$ and $M_{IJ} = R^A_{\mu\nu}/\sqrt{p}$.

- `bs0 :: Basis` — the basis of the original state.
- `secd_lst :: Vector{Vector{Vector{Int64}}}` — the list of QNDiag sectors of subsystems to be calculated. Each of its elements is a two element vector ; the first specifies the sector for subsystem A, and the second specifies the sector for subsystem B.
- `secf_lst :: Vector{Vector{Vector{ComplexF64}}}` — the list of QNOffd sectors of subsystems to be calculated. Each of its elements is a two element vector ; the first specifies the sector for subsystem A, and the second specifies the sector for subsystem B.
- `qnd_a :: Vector{QNDiag}, qnd_b :: Vector{QNDiag}, qnf_a :: Vector{QNOffd}, qnf_b :: Vector{QNOffd}` — the diagonal and off-diagonal quantum numbers of the subsystems A and B. By default `qnd_b = qnd_a` and `qnf_b = qnf_a`.
- `amp_oa :: Vector{ComplexF64}` ` and ` `amp_ob :: Vector{ComplexF64}` — arrays that specify the amplitudes α_o and β_o. By default $\beta_o = \sqrt{1 - \alpha_o^2}$.

and the output is a dictionary whose keys are named tuples that specify the sector containing entries `secd_a`, `secf_a`, `secd_b`, `secf_b` and values are lists of eigenvalues of the density matrix in those sectors.

In the example of Ising model, to calculate the entanglement entropy cut from the equator, we first need to specify the quantum numbers of the subsystems : the conservation of N_e, L_z and the $\mathbb{Z}_2$ symmetry.

```
qnd_a = [ GetNeQNDiag(no), GetLz2QNDiag(nm, nf) ]
qnf_a = [ GetFlavPermQNOffd(nm, nf, [2, 1]) ]
```

we then specify the sectors to calculate : The number of electrons in subsystem A run from 0 to N_m ; the angular momenta in subsystem A can take all permitted values ; for subsystem B, $N_{e,B} = N_m - N_{e,A}$, $L_{z,B} = -L_{z,A}$; the $\mathbb{Z}_2$ sectors of the two subsystems are the same.

```
secd_lst = Vector{Vector{Int64}}[]
for nea = 0 : nm
    neb = nm - nea
    for lza = -min(nea, neb) * (nm - 1) : 2 : min(nea, neb) * (nm - 1)
        lzb = -lza
        push!(secd_lst, [[nea, lza], [neb, lzb]])
    end
```

```
end
secf_lst = [ [[1], [1]], [[-1], [-1]] ]
```

Finally, we specify the list of amplitute α_m.

```
amp_oa = [ sqrt(beta_inc(m, nm - m + 1, 0.5)) for f = 1 : 2 for m = 1 : nm]
```

To calculate the entanglement spectrum,

```
ent_spec = GetEntSpec(st_g, bs, secd_lst, secf_lst ; qnd_a, qnf_a, amp_oa)
```

The entanglement entropy can be calculated by collecting all the eigenvalues of the density matrix.

```
eig_rho = vcat(values(ent_spec)...)
ent_entropy = -sum(eig_rho .* log.(eig_rho))
```

7.10 Fuzzifino — module for boson-fermion mixture

Fuzzifino is a module for ED calculation on the fuzzy sphere for systems with both bosons and fermions. The procedure is similar to FuzzifiED. We define several new types `SQNDiag`, `SQNOffd`, `SConf`, `SBasis`, `STerm` and `SOperator` that is parallel to the original versions `QNDiag`, `QNOffd`, `Conf`, `Basis`, `Term` and `Operator`. Several points should be noted.

- For each configuration, a boson part and a fermion part are stored. The boson configurations are indexed in the ascending order from the last site to the first site. A maximal total occupation should be given. *E.g.*, for 3 sites with maximal occupation 2, the configurations are 000, 001, 002, 010, 011, 020, 100, 101, 110, 200 numbered from 1 to 10. The reserse look-up is realised by a similar Lin-table.
- The operator string in a term is still stored in the form $\{p_1, o_1, p_2, o_2, \ldots, p_l, o_l\}$, but now positive o represents fermions, and negative o represents boson with site number $|o|$. The bosonic creation and annihilation operator acts with an additional factor, *e.g.*, $b^\dagger|n\rangle = \sqrt{n+1}|n+1\rangle, b|n\rangle = \sqrt{n}|n-1\rangle$ for a single site.

8 Density matrix renormalisation group (DMRG) with FuzzifiED

Having introduced ED, we now turn to density matrix renormalisation group (DMRG) that deals with larger systems. We briefly describe its procedure and give an instruction for using

FuzzifiED for DMRG.

8.1 DMRG with ITensor

Practically, the `dmrg` function in ITensor package automatically uses DMRG to optimise a matrix product state (MPS) to be the lowest eigenstate of a Hermitian Hamiltonian represented as a matrix product operator (MPO). To generate the input of the function, one needs to

1. construct a set of sites that carries a certain set of QNDiags,
2. construct a MPO representing the Hamiltonian on the sites from a set of terms (or OpSum in ITensor), and
3. construct an initial MPS on the sites in the desired symmetry sector.

In FuzzifiED, a new SiteType `"FuzzyFermion"` is defined that behaves similar to the built-in `"Fermion"` type and a single site can be generated from the QNDiags

```
siteind("FuzzyFermion" ; o :: Int64, qnd :: Vector{QNDiag})
```

The set of sites can be generated by

```
GetSites(qnd :: Vector{QNDiag})
```

In the example of Ising model, for convenience we exchange the Pauli matrices σ^x and σ^z so that the two flavours carry $\mathbb{Z}_2$-charge 0 and 1. The sites can be constructed by

```
nm = 12
nf = 2
no = nm * nf
sites = GetSites([
    GetNeQNDiag(nm * nf),
    GetLz2QNDiag(nm, nf),
    GetZnfChargeQNDiag(nm, nf)
])
```

In ITensor, the MPO is generated from an OpSum and the sites. The OpSum can be directly converted from the array of terms. In the example of Ising model,

```
sgx = [  0  1 ;  1  0 ]
sgz = [  1  0 ;  0 -1 ]
ps_pot = [4.75, 1.] ./ 2
tms_hmt = SimplifyTerms(
```

```
    GetDenIntTerms(nm, 2, ps_pot) -
    GetDenIntTerms(nm, 2, ps_pot, sgx) -
    3.16 * GetPolTerms(nm, nf, sgz)
)
os_hmt = OpSum(tms_hmt)
hmt = MPO(os_hmt, sites)
```

To calculate the $\mathbb{Z}_2$-even $L^z = 0$ sector, the initial state can be taken as the all the $\mathbb{Z}_2$-even sites being filled and all the $\mathbb{Z}_2$-odd sites being empty[12]

```
cf0 = [ isodd(o) ? 1 : 0 for o = 1 : no ]
st0 = MPS(sites, string.(cf0))
```

Having these ingrediants ready, we can call the `dmrg` function. To ensure performance, the maximal bond dimension should be increased gradually and the noise decreased gradually to 0. An example that deals with maximal bond dimension 500 is

```
EI, stI = dmrg(hmt, st0 ; nsweeps = 10, maxdim = [10,20,50,100,200,500],
    noise = [1E-4,3E-5,1E-5,3E-6,1E-6,3E-7], cutoff = [1E-8])
```

To generate a $\mathbb{Z}_2$-odd initial state, we can simply flip the spin on the first orbital

```
cf1 = cf0
cf1[1] = 0
cf1[2] = 1
st1 = MPS(sites, string.(cf1))
Es, sts = dmrg(hmt, st1 ; nsweeps = 10, maxdim = [10,20,50,100,200,500],
    noise = [1E-4,3E-5,1E-5,3E-6,1E-6,3E-7], cutoff = [1E-8])
```

The first excited $\mathbb{Z}_2$-even state can be generated by adding a projector $w|0\rangle\langle 0|$ to the MPO

```
Ee, ste = dmrg(hmt, [stI], st0 ; nsweeps = 10, maxdim =
    [10,20,50,100,200,500], noise = [1E-4,3E-5,1E-5,3E-6,1E-6,3E-7], cutoff =
    [1E-8], weight = 100)
```

The inner product can be measured by the ITensor function `inner`. For example, to measure the angular momentum L^2 of the ground state,

```
tms_l2 = GetL2Terms(nm, 2)
```

[12] Note that ITensor takes the string `"1"` instead of the number `1` as occupied and `"0"` instead of `0` as filled

```
l2 = MPO(OpSum(tms_l2))
val_l2I = inner(stI', l2, stI)
```

To measure the OPE coefficient $f_{\sigma\sigma\epsilon} = \langle\sigma|n^x_{00}|\epsilon\rangle/\langle\sigma|n^x_{00}|0\rangle$[13]

```
obs_nx = GetDensityObs(nm, 2, sgx)
tms_nx00 = SimplifyTerms(GetComponent(obs_nx, 0.0, 0.0))
nx00 = MPO(OpSum(tms_nx00))
f_sse = abs(inner(sts', nx00, ste) / inner(sts', nx00, stI))
```

8.2 The EasySweep extension

The extension EasySweep facilitates the management of DMRG process. It automatically records the intermediate results and recover these results if a job is stopped and run again on HPC. It also manages the gradual increase of maximal bond dimensions and the determination of convergence by the criteria of energy. This extension contains the following functions :

- `GetMPOSites` returns the MPO and sites for given operator terms and a Hilbert space with given quantum numbers. The function first checks if the MPO and sites are already stored in a specified file. If they are already stored, they are read and returned. Otherwise the sites are generated from with the quantum numbers and the MPO is generated from the terms. The MPO and sites are then written into the file and returned.

- `GetMPO` returns the MPO for given operator terms and a given set of sites. The function first checks if the MPO is already stored in a specified file. If it is already stored, it is read and returned. Otherwise MPO is generated from the terms. The MPO is then written into the file and returned.

- `SweepOne` performs one round of DMRG sweeps for a given maximal bond dimensions D_m and returns the energy and the MPS. The function first checks if the calculation has already been done and the results are already stored in a specified file. If they are already stored, they are read and returned. Otherwise the DMRG process is performed with a specified maximal bond dimension and number of sweeps. The sweeps are ended if the energy difference is less than a tolerence or some alternative criteria. The resulting energy and MPS are written into file and returned.

[13] Note that the indices x and z have already been exchanged here.

- `EasySweep` automatically performs several rounds of DMRG sweeps with increasing bond dimensions and returns energy and MPS. The function first checks the calculation has been partly done and the intermediate results have been stored in a specified file. The calculation is then picked up from the round that is previously stopped. The entire process is stopped if the energy difference between two rounds is less than a certain tolerence or the bond dimension of the result is less than 0.9 times the maximal bond dimension. The resulting energy and MPS are written into file and returned.

To use the this extension, one need to use the packages `ITensors`, `ItensorMPS` and `HDF5`. A path need to be created a priori to store the result HDF5 files. We recommend using the package `ITensorMPOConstruction` to generate the MPO.

```
using FuzzifiED
using ITensors, ITensorMPS, HDF5
using ITensorMPOConstruction
const sgx = [  0  1 ;  1  0 ]
const sgz = [  1  0 ;  0 -1 ]

function MyMPO(os, sites)
    operatorNames = [ "I", "C", "Cdag", "N" ]
    opCacheVec = [ [OpInfo(ITensors.Op(name, n), sites[n]) for name in
        operatorNames] for n in eachindex(sites)  ]
    return MPO_new(os, sites ; basis_op_cache_vec = opCacheVec)
end

nm = 12
nf = 2
no = nm * nf

path = "nm_$(nm)_tmp/"
mkpath(path)
```

Like the previous section, we first put in the terms for Hamiltonian and the QNDiags

```
ps_pot = [4.75, 1.] ./ 2
tms_hmt = SimplifyTerms(
    GetDenIntTerms(nm, 2, ps_pot) -
```

```
    GetDenIntTerms(nm, 2, ps_pot, sgx) -
    3.16 * GetPolTerms(nm, 2, sgz)
)
qnd = [
    GetNeQNDiag(no),
    GetLz2QNDiag(nm, nf),
    GetZnfChargeQNDiag(nm, nf)
]
```

The Sites and Hamiltonian MPO can be generated with the function `GetMPOSites`.

```
hmt, sites = GetMPOSites("hmt", tms_hmt, qnd ; path, mpo_method = MyMPO)
```

To generate the initial MPS that respects the $\mathbb{Z}_2$ symmetry, we can use a direct product state.

```
cf0 = [ isodd(o) ? 1 : 0 for o = 1 : no ]
st0 = MPS(sites, string.(cf0))
cf1 = cf0
cf1[1] = 0
cf1[2] = 1
st1 = MPS(sites, string.(cf1))
```

The lowest eigenenergies and the eigenstate MPSs $|0\rangle, |\sigma\rangle, |\epsilon\rangle$ can be easily generated by the function `EasySweep`.

```
EI, stI = EasySweep("g", hmt, st0 ; path)
Ee, ste = EasySweep("e", hmt, st0 ; path, proj = ["g"])
Es, sts = EasySweep("s", hmt, st1 ; path)
```

To measure the angular momentum L^2 of the ground state, we generate the MPO for L^2.

```
tms_l2 = GetL2Terms(nm, 2)
l2 = GetMPO("l2", tms_l2, sites ; path, mpo_method = MyMPO)
val_l2I = inner(stI', l2, stI)
```

Similarly, to measure the OPE coefficient $f_{\sigma\sigma\epsilon} = \langle\sigma|n^x_{00}|\epsilon\rangle / \langle\sigma|n^x_{00}|0\rangle$

```
obs_nx = GetDensityObs(nm, 2, sgx)
tms_nx00 = SimplifyTerms(GetComponent(obs_nx, 0.0, 0.0))
nx00 = GetMPO("nx00", tms_nx00, sites ; path, mpo_method = MyMPO)
f_sse = abs(inner(sts', nx00, ste) / inner(sts', nx00, stI))
```

9 Practical examples with FuzzifiED

FuzzifiED can help reproduce almost all the ED and DMRG results in fuzzy sphere works. We offer a series of examples that reproduces these results to help the users get started with the package. The code can be found in the `examples` directory of the source code repository.

- `ising_spectrum.jl` calculates the spectrum of 3d Ising model on fuzzy sphere at $N_m = 12$. For each (P, Z, R) sector, 20 states are calculated. This example reproduces Table I and Figure 4 in Ref. [1].
- `ising_phase_diagram.jl` calculates the phase diagram of fuzzy sphere Ising modelby calculating the order parameter $\langle M^2 \rangle$. This example reproduces Figure 3 in Ref. [1].
- `ising_ope.jl` calculates various OPE coefficients at $N_m = 12$ by taking overlaps between CFT states and density operators and composite. This example reproduces Figure 2 and Table I in Ref. [2].
- `ising_correlator.jl` calculates the $\sigma\sigma$ two-point function on sphere and the $\sigma\sigma\sigma\sigma$ four-point function on sphere, 0 and ∞. This example reproduces Figures 1c and 2a in Ref. [3].
- `ising_optimisation.jl` defines a cost function as the square sum of the deviations of descendants and stress tensor to evaluate the conformal symmetry for Ising model and minimises this cost function to find the best parameter.
- `ising_full_spectrum.jl` calculates the full spectrum of 3d Ising model on fuzzy sphere at $N_m = 10$ for sector $(P, Z, R) = (1, 1, 1)$.
- `ising_space_entangle.jl` calculates the entanglement entropy of the Ising ground state along the real space cut of $\theta = 0.500\pi$ and 0.499π respectively, and use these two data to extract finite size F-function without sustracting the IQHE contribution. This example reproduces [10].
- `ising_orbital_entangle.jl` calculates the entanglement entropy of the Ising ground state along the orbital space cut at $m = 0$, and also the entanglement spectrum in the half-filled $l_z = 0, 1$ and both $\mathbb{Z}_2$ sectors.
- `ising_generator.jl` examines the quality of conformal symmetry at $N_m = 12$ by examining the matrix elements of conformal generators $P^z + K^z$ and compare the states $(P^z + K^z)|\Phi\rangle$ with the CFT expectations. This example reproduces Figure 7 in Ref. [14].

- `defect_spectrum.jl` calculates the spectrum of magnetic line defect in 3d Ising model in $l_z = 0, P = \pm 1$ and $l_z = 1$ sectors, calibrated by bulk T. This example reproduces Table I in Ref. [6].

- `defect_correlator.jl` calculates the 1-pt function σ and 2-pt function $\sigma\hat{\phi}$ of magnetic line defect in 3d Ising model. The normalisation of the correlators require extra bulk data. This example reproduces Figure 4 in Ref. [6].

- `defect_changing.jl` calculates the spectrum of the defect creation and changing operators of the magnetic line defect in 3d Ising model. This example reproduces Table 2 and Figure 5 in Ref. [9].

- `defect_overlap.jl` calculates the g-function of magnetic line defect in 3d Ising model using the ovelaps between the bulk, defect ground state and the lowest defect-creation state. This example reproduces Figure 6 in Ref. [9].

- `cusp_dim.jl` calculates the scaling dimension of the cusp of the magnetic line defect in 3d Ising model as a function of the angle θ. This example reproduces Table 2, upper panel in Ref. [11].

- `surface_ordinary_spectrum.jl` calculates the spectrum of ordinary surface CFT in 3d Ising model calibrated by surface displacement operator D in the orbital boundary scheme. This example reproduces Figures 3 and 4 in Ref. [12].

- `surface_normal_spectrum.jl` calculates the spectrum of normal surface CFT in 3d Ising model calibrated by surface displacement operator D in the orbital boundary scheme. This example reproduces Figure 5 in Ref. [12].

- `o3_wf_spectrum.jl` calculates the spectrum of O(3) Wilson-Fisher CFT using the bilayer Heisenberg model. This example reproduces Table I and Figure 2 in Ref. [8].

- `so5_spectrum.jl` calculates the spectrum of SO(5) DQCP on fuzzy sphere. This example reproduces Table II in Ref. [4].

- `sp3_spectrum.jl` calculates the spectrum of Sp(3) CFT on fuzzy sphere. This example reproduces Table I in Ref. [16].

- `ising_frac_fermion.jl` calculates the spectrum of 3d Ising model on fuzzy sphere for fermions at fractional filling $\nu = 1/3$. This example reproduces Figure 10 in Ref. [17].

- `ising_frac_boson.jl` calculates the spectrum of 3d Ising model on fuzzy sphere for bosons at fractional filling $\nu = 1/2$ with the module Fuzzifino. This example reproduces Figure 12a, b in Ref. [17].
- `ising_spectrum_krylov.jl` calculates the spectrum of 3d Ising model on fuzzy sphere by calling the eigsolve function in KrylovKit.jl instead of Arpack.
- `ising_spectrum_cuda.jl` calculates the spectrum of 3d Ising model on fuzzy sphere for one sector by performing the sparse matrix multiplication on CUDA.

A Data structures in exact diagonalisation

In this appendix, we describe the two data structures that are used in the exact diagonalisation (ED) process, *viz.* Lin table for the reverse look-up process that returns the index from the binary string and the compressed sparse column (CSC) that is used to store the sparse matrices.

A.1 Construction of Lin table

In this section, we describe the construction of Lin table which is used for a reverse look-up process that returns the index i from the binary string $conf_i$ described in Section 7.2.

Each binary string $conf_i$ is devided into a left part $l(conf_i)$ from the N_{or}-th bit to the N_o-1-th bit and a right part $r(conf_i)$ from the 0-th bit to the $N_{or}-1$-th bit from the right. The cut N_{or} is typically taken as $N_o/2$. Two arrays lid and rid with length $2^{N_o-N_{or}}$ and $2^{N_{or}}$ are constructed, such that

$$i = lid_{l(conf_i)} + rid_{r(conf_i)} \tag{A.1}$$

As a simplest example for this, consider the configurations on 6 sites filled with 3 particles. The construction is listed in Table 2.

c	lid_c	rid_c	i	$conf_i$	l_i	r_i	lid_{l_i}	rid_{r_i}	i	$conf_i$	l_i	r_i	lid_{l_i}	rid_{r_i}
0	1	0	1	000111	0	7	1	0	11	100011	4	3	11	0
1	2	0	2	001011	1	3	2	0	12	100101	4	5	11	1
2	5	1	3	001101	1	5	2	1	13	100110	4	6	11	2
3	8	0	4	001110	1	6	2	2	14	101001	5	1	14	0
4	11	2	5	010011	2	3	5	0	15	101010	5	2	14	1
5	14	1	6	010101	2	5	5	1	16	101100	5	4	14	2
6	17	2	7	010110	2	6	5	2	17	110001	6	1	17	0
7	20	0	8	011001	3	1	8	0	18	110010	6	2	17	1
			9	011010	3	2	8	1	19	110100	6	4	17	2
			10	011100	3	4	8	2	20	111000	7	0	20	0

Table 2. An example of explicit construction of the Lin table in the configurations on 6 sites filled with 3 particles. Here we use a shorthand notation $l_i = l(conf_i)$ and $r_i = r(conf_i)$

A.2 Compressed sparse column (CSC) sparse matrix

In this section, we describe the storage of sparse matrices in the format of compressed sparse column (CSC), which is used in Section 7.5.

In the CSC format, the matrix is stored in three arrays. We first index elements from 1 to the number of elements N_{el} in the ascending order from the column to column, and in each column from row to row.

1. The array *colptr* records the column index of each element. It is of length $N + 1$ where N is the number of columns. The first N items record the index of the first element in each column. The last item records $N_{\text{el}} + 1$, so that the elements that belong to the i-th column have index from $colptr_i$ to $colptr_{i+1} - 1$.
2. The array *rowid* records the row index of each element and is of length N_{el}.
3. The array *elval* records the value of each element and is of length N_{el}.

We give a simple example in Table 3.

$$M = \begin{pmatrix} 10 & 40 & 0 & 0 \\ 20 & 0 & 70 & 80 \\ 0 & 0 & 0 & 90 \\ 30 & 50 & 0 & 100 \\ 0 & 60 & 0 & 0 \end{pmatrix}$$

i	1	2	3	4	5	6	7	8	9	10
$colptr_i$	1	4	7	8	11					
$rowid_i$	1	2	4	1	4	5	2	2	3	4
$elval_i$	10	20	30	40	50	60	70	80	90	100

Table 3. An example of the storing a 4×5 matrix into the CSC format.

B Tutorial code

In Sections 7 and 8, we demonstrate the usage of FuzzifiED interfaces with an example that

1. calculates the lowest eigenstates in the symmetry sector $L^z = 0$ and $(\mathcal{P}, \mathcal{Z}, \mathcal{R}) = (+,+,+)$ (for DMRG, $\mathcal{Z} = +$ only),
2. measures their total angular momenta (for DMRG, the ground state only), and
3. calcultes the OPE coefficient $f_{\sigma\sigma\epsilon} = \langle\sigma|n^z_{00}|\epsilon\rangle / \langle\sigma|n^z_{00}|0\rangle$.

In this Appendix, we collect the full version of the codes, including

1. the ED code that uses only the core functions,
2. the ED code that uses the built-in models,
3. the DMRG code that converts the format into ITensor,
4. the DMRG code that uses the EasySweep extension.

B.1 ED using core functions

```
using FuzzifiED
using WignerSymbols

let

nf = 2
nm = 12
no = nf * nm
qnd = [
    QNDiag(fill(1, no)),
    QNDiag([ (o - 1) ÷ nf * 2 - (nm - 1) for o = 1 : no ])
]
cfs = Confs(no, [nm, 0], qnd)

qnf = [
    QNOffd([ isodd(o) ? o + 1 : o - 1 for o = 1 : no], true, ComplexF64[
        isodd(o) ? -1 : 1 for o = 1 : no]),
    QNOffd([ isodd(o) ? o + 1 : o - 1 for o = 1 : no]),
    QNOffd([ isodd(o) ? no - o : no + 2 - o for o = 1 : no], ComplexF64(-1)
        .^ (collect(0 : nm * nf - 1) .÷ nf))
]
bs = Basis(cfs, [1, 1, 1], qnf)

```

```
ps_pot = [ 4.75, 1. ] * 2.
h = 3.16
tms_hmt = Term[]
for m1 = 0 : nm - 1
    f1 = 0
    o1 = m1 * nf + f1 + 1
    m1r = m1 - s
    for m2 = 0 : nm - 1
        f2 = 1
        o2 = m2 * nf + f2 + 1
        m2r = m2 - s
        for m3 = 0 : nm - 1
            f3 = 1
            o3 = m3 * nf + f3 + 1
            m3r = m3 - s
            m4 = m1 + m2 - m3
            if (m4 < 0 || m4 >= nm) continue end
            f4 = 0
            o4 = m4 * nf + f4 + 1
            m4r = m4 - s
            val = ComplexF64(0)
            for l in eachindex(ps_pot)
                if (abs(m1r + m2r) > nm - l || abs(m3r + m4r) > nm - l) break
                    end
                val += ps_pot[l] * (2 * nm - 2 * l + 1) * wigner3j(s, s, nm -
                    l, m1r, m2r, -m1r - m2r) * wigner3j(s, s, nm - l, m4r,
                   m3r, -m3r - m4r)
            end
            tms_hmt += Terms(val, [1, o1, 1, o2, 0, o3, 0, o4])
        end
    end
    o1x = o1 + 1
    tms_hmt += Terms(-h, [1, o1, 0, o1x])
    tms_hmt += Terms(-h, [1, o1x, 0, o1])
end

```

```
hmt = Operator(bs, tms_hmt)
hmt_mat = OpMat(hmt)
enrg, st = GetEigensystem(hmt_mat, 10)

tms_lz =
    [ begin m = div(o - 1, nf)
        Term(m - s, [1, o, 0, o])
    end for o = 1 : no ]
tms_lp =
    [ begin m = div(o - 1, nf)
        Term(sqrt(m * (nm - m)), [1, o, 0, o - nf])
    end for o = nf + 1 : no ]
tms_lm = tms_lp'
tms_l2 = SimplifyTerms(tms_lz * tms_lz - tms_lz + tms_lp * tms_lm)
l2 = Operator(bs, tms_l2)
l2_mat = OpMat(l2)
l2_val = [ st[:, i]' * l2_mat * st[:, i] for i in eachindex(enrg)]
@show real(l2_val)
st_T = st[:, 3]
st_L2T = l2_mat * st[:, 3]
@show abs(st_L2T' * st_T) ^ 2 / ((st_T' * st_T) * (st_L2T' * st_L2T))

bs1 = Basis(cfs, [ 1, -1, 1 ], qnf)
hmt = Operator(bs1, bs1, tms_hmt ; red_q = 1, sym_q = 1)
hmt_mat = OpMat(hmt)
enrg1, st1 = GetEigensystem(hmt_mat, 10)
stI = st[:, 1]
ste = st[:, 2]
sts = st1[:, 1]

tms_nz00 = Term[]
for m1 = 0 : nm - 1
    o1u = 2 * m1 + 1
    o1d = 2 * m1 + 2
    push!(tms_nz00, Term( 1 / nm, [1, o1u, 0, o1u]))
    push!(tms_nz00, Term(-1 / nm, [1, o1d, 0, o1d]))
```

```
end
nz00 = Operator(bs, bs1, tms_nz00 ; red_q = 1)
@show abs((sts' * nz00 * ste) / (sts' * nz00 * stI))
```

B.2 ED using built-in models

```
using FuzzifiED
sg1 = [  1  0 ;  0  0 ]
sg2 = [  0  0 ;  0  1 ]
sgx = [  0  1 ;  1  0 ]
sgz = [  1  0 ;  0 -1 ]

nm = 12
qnd = [
    GetNeQNDiag(2 * nm),
    GetLz2QNDiag(nm, 2)
]
cfs = Confs(2 * nm, [nm, 0], qnd)

qnf = [
    GetParityQNOffd(nm, 2, [2, 1], [-1, 1]),
    GetFlavPermQNOffd(nm, 2, [2, 1]),
    GetRotyQNOffd(nm, 2)
]
bs = Basis(cfs, [1, 1, 1], qnf)

tms_hmt = SimplifyTerms(
    GetDenIntTerms(nm, 2, 2 .* [4.75, 1.], sg1, sg2) -
    3.16 * GetPolTerms(nm, 2, sgx)
)

hmt = Operator(bs, tms_hmt)
hmt_mat = OpMat(hmt)
enrg, st = GetEigensystem(hmt_mat, 10)
@show enrg

tms_l2 = GetL2Terms(nm, 2)
```

```
l2 = Operator(bs, tms_l2)
l2_mat = OpMat(l2)
l2_val = [ st[:, i]' * l2_mat * st[:, i] for i in eachindex(enrg)]
@show l2_val

st_T = st[:, 3]
st_L2T = l2_mat * st[:, 3]
@show abs(st_L2T' * st_T) ^ 2 / ((st_T' * st_T) * (st_L2T' * st_L2T))

bs1 = Basis(cfs, [1, -1, 1], qnf)
hmt = Operator(bs1, bs1, tms_hmt ; red_q = 1, sym_q = 1)
hmt_mat = OpMat(hmt)
enrg1, st1 = GetEigensystem(hmt_mat, 10)
stI = st[:, 1]
ste = st[:, 2]
sts = st1[:, 1]

obs_nz = GetDensityObs(nm, 2, sgz)
tms_nz00 = SimplifyTerms(GetComponent(obs_nz, 0.0, 0.0))
nz00 = Operator(bs, bs1, tms_nz00 ; red_q = 1)
@show abs((sts' * nz00 * ste) / (sts' * nz00 * stI))
```

B.3 DMRG using format conversion into ITensor

```
using FuzzifiED
using ITensors, ITensorMPS
FuzzifiED.ElementType = Float64
sgx = [  0  1 ;  1  0 ]
sgz = [  1  0 ;  0 -1 ]

nm = 12
nf = 2
no = nm * nf

sites = GetSites([
    GetNeQNDiag(nm * nf),
    GetLz2QNDiag(nm, nf),
```

```
    GetZnfChargeQNDiag(nm, nf)
])
ps_pot = [4.75, 1.] ./ 2
tms_hmt = SimplifyTerms(
    GetDenIntTerms(nm, 2, ps_pot) -
    GetDenIntTerms(nm, 2, ps_pot, sgx) -
    3.16 * GetPolTerms(nm, nf, sgz)
)
os_hmt = OpSum(tms_hmt)
hmt = MPO(os_hmt, sites)

cf0 = [ isodd(o) ? 1 : 0 for o = 1 : no ]
st0 = MPS(sites, string.(cf0))

EI, stI = dmrg(hmt, st0 ; nsweeps = 10, maxdim = [10,20,50,100,200,500],
    noise = [1E-4,3E-5,1E-5,3E-6,1E-6,3E-7], cutoff = [1E-8])
@show Eg
```

B.4 DMRG with Easy Sweep

```
using FuzzifiED
using ITensors, ITensorMPS, HDF5
using ITensorMPOConstruction
const sgx = [  0  1 ;  1  0 ]
const sgz = [  1  0 ;  0 -1 ]

function MyMPO(os, sites)
    operatorNames = [ "I", "C", "Cdag", "N" ]
    opCacheVec = [ [OpInfo(ITensors.Op(name, n), sites[n]) for name in
        operatorNames] for n in eachindex(sites)  ]
    return MPO_new(os, sites ; basis_op_cache_vec = opCacheVec)
end

nm = 12
nf = 2
no = nm * nf

```

```
17 path = "nm_$(nm)_tmp/"
18 mkpath(path)
19
20 ps_pot = [4.75, 1.] ./ 2
21 tms_hmt = SimplifyTerms(
22     GetDenIntTerms(nm, 2, ps_pot) -
23     GetDenIntTerms(nm, 2, ps_pot, sgx) -
24     3.16 * GetPolTerms(nm, 2, sgz)
25 )
26 qnd = [
27     GetNeQNDiag(no),
28     GetLz2QNDiag(nm, nf),
29     GetZnfChargeQNDiag(nm, nf)
30 ]
31 hmt, sites = GetMPOSites("hmt", tms_hmt, qnd ; path, mpo_method = MyMPO)
32
33 cf0 = [ isodd(o) ? 1 : 0 for o = 1 : no ]
34 st0 = MPS(sites, string.(cf0))
35
36 EI, stI = EasySweep("g", hmt, st0 ; path)
37 Ee, ste = EasySweep("e", hmt, st0 ; path, proj = ["g"])
38
39 cf1 = cf0
40 cf1[1] = 0
41 cf1[2] = 1
42 st1 = MPS(sites, string.(cf1))
43 Es, sts = EasySweep("s", hmt, st1 ; path)
44
45 tms_l2 = GetL2Terms(nm, 2)
46 l2 = GetMPO("l2", tms_l2, sites ; path, mpo_method = MyMPO)
47 l2_val = stI' * l2 * stg
48
49 obs_nz = GetDensityObs(nm, 2, sgz)
50 tms_nz00 = SimplifyTerms(GetComponent(obs_nz, 0.0, 0.0))
51 nz00 = GetMPO("nz00", tms_nz00, sites ; path, mpo_method = MyMPO)
52 f_sse = abs(inner(sts', nz00, ste) / inner(sts', nz00, stI))
```

C Glossaries for interfaces in FuzzifiED

AngModes	GetL2Terms	OpSum
Basis	GetLz2QNDiag	ParticleHole
Confs	GetMPO	QNDiag
Confs	GetMPOSites	QNOffd
CuSparseMatrixCSC	GetNeQNDiag	SBasis
EasySweep	GetPairingMod	SConfs
ElementType	GetPairingObs	SilentStd
FilterComponent	GetPairIntTerms	SimplifyTerms
FilterL2	GetParityQNOffd	SOperator
GetC2Terms	GetPinOrbQNDiag	SparseMatrixCSC
GetComponent	GetPointValue	SphereObs
GetConfId	GetPolTerms	SQNDiag
GetConfWeight	GetQNDiags	SQNOffd
GetDenIntTerms	GetRotyQNOffd	StateDecompMat
GetDensityMod	GetSites	STerm
GetDensityObs	GetZnfChargeQNDiag	STerms
GetEigensystem	Laplacian	StoreComps
GetEigensystemCuda	Libpath	StoreComps!
GetEigensystemKrylov	Libpathino	STransf
GetElectronMod	Matrix	SweepOne
GetElectronObs	NormalOrder	Term
GetEntSpec	NumThreads	Terms
GetFlavPermQNOffd	OpenHelp!	Transf
GetFlavQNDiag	Operator	TruncateQNDiag
GetIntMatrix	OpMat	

[14] Here each symbol represents a specific type : amd is a AngModes, obs is a SphereObs, op is a Operator, sop is a SOperator, mat is a OpMat, qnd is a QNDiag, sqnd is a SQNDiag, qnf is a QNOffd, sqnf is a SQNOffd, trs is a Transf, strs is a STransf, vec is a Vector, # is a number.

Operations

amd + amd[14]

obs + obs

qnd + qnd

tms + tms

stms + stms

* amd

* obs

* qnd

* sqnd

* tms

* stms

amd * amd

obs * obs

qnf * qnf

sqnf * sqnf

tms * tms

stms * stms

mat * vec

op * vec

sop * vec

trs * vec

strs * vec

vec' * mat * vec

vec' * op * vec

vec' * sop * vec

amd'

obs'

tms'

stms'

tms ^ #

stms ^ #

References

[1] 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].

[2] 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].

[3] C. Han, L. Hu, W. Zhu and Y.-C. He, *Conformal four-point correlators of the three-dimensional Ising transition via the quantum fuzzy sphere*, *Phys. Rev. B* **108** (2023) 235123 [arXiv:2306.04681].

[4] Z. Zhou, L. Hu, W. Zhu and Y.-C. He, SO(5) *deconfined phase transition under the fuzzy-sphere microscope: Approximate conformal symmetry, pseudo-criticality, and operator spectrum*, *Phys. Rev. X* **14** (2024) 021044 [arXiv:2306.16435].

[5] B.-X. Lao and S. Rychkov, *3d Ising CFT and exact diagonalization on icosahedron: The power of conformal perturbation theory*, *SciPost Phys.* **15** (2023) 243 [arXiv:2307.02540].

[6] L. Hu, Y.-C. He and W. Zhu, *Solving conformal defects in 3d conformal field theory using fuzzy sphere regularization*, *Nature Commun.* **15** (2024) 9013 [arXiv:2308.01903].

[7] J.S. Hofmann, F. Goth, W. Zhu, Y.-C. He and E. Huffman, *Quantum Monte Carlo simulation of the 3d Ising transition on the fuzzy sphere*, *SciPost Phys. Core* **7** (2024) 028 [arXiv:2310.19880].

[8] C. Han, L. Hu and W. Zhu, *Conformal operator content of the Wilson-Fisher transition on fuzzy sphere bilayers*, *Phys. Rev. B* **110** (2024) 115113 [arXiv:2312.04047].

[9] Z. Zhou, D. Gaiotto, Y.-C. He and Y. Zou, *The g-function and defect changing operators from wavefunction overlap on a fuzzy sphere*, *SciPost Phys.* **17** (2024) 021 [arXiv:2401.00039].

[10] L. Hu, W. Zhu and Y.-C. He, *Entropic F-function of 3d Ising conformal field theory via the fuzzy sphere regularization*, arXiv:2401.17362.

[11] G. Cuomo, Y.-C. He and Z. Komargodski, *Impurities with a cusp: general theory and 3d Ising*, *JHEP* **11** (2024) 061 [arXiv:2406.10186].

[12] Z. Zhou and Y. Zou, *Studying the 3d Ising surface CFTs on the fuzzy sphere*, arXiv:2407.15914.

[13] M. Dedushenko, *Ising BCFT from fuzzy hemisphere*, arXiv:2407.15948.

[14] G. Fardelli, A.L. Fitzpatrick and E. Katz, *Constructing the infrared conformal generators on the fuzzy sphere*, arXiv:2409.02998.

[15] R. Fan, *Note on explicit construction of conformal generators on the fuzzy sphere*, arXiv:2409.08257.

[16] Z. Zhou and Y.-C. He, *A new series of 3d CFTs with* Sp(n) *global symmetry on fuzzy sphere*, arXiv:2410.00087.

[17] C. Voinea, R. Fan, N. Regnault and Z. Papić, *Regularizing 3d conformal field theories via anyons on the fuzzy sphere*, arXiv:2411.15299.

[18] M. Fishman, S.R. White and E.M. Stoudenmire, *The ITensor software library for tensor network calculations*, *SciPost Phys. Codeb.* (2022) 4 [arXiv:2007.14822].

[19] S. Rychkov, *EPFL Lectures on Conformal Field Theory in* $D \geq 3$ *Dimensions*, Briefs in Physics, Springer (1, 2016), 10.1007/978-3-319-43626-5, [arXiv:1601.05000].

[20] D. Simmons-Duffin, *The conformal bootstrap*, in *Theoretical Advanced Study Institute in Elementary Particle Physics: New Frontiers in Fields and Strings*, pp. 1–74, 2017, DOI [arXiv:1602.07982].

[21] A.M. Polyakov, *Conformal symmetry of critical fluctuations*, *JETP Lett.* **12** (1970) 381.

[22] J.L. Cardy, *Scaling and Renormalization in Statistical Physics*, Cambridge Lecture Notes in Physics, Cambridge University Press (1996).

[23] J.M. Maldacena, *The large-N limit of superconformal field theories and supergravity*, *Adv. Theor. Math. Phys.* **2** (1998) 231.

[24] A.B. Zamolodchikov, *Irreversibility of the flux of the renormalization group in a 2D field theory*, *JETP Lett.* **43** (1986) 730.

[25] P. Di Francesco, P. Mathieu and D. Sénéchal, *Conformal field theory*, Graduate texts in contemporary physics, Springer, New York, NY (1997), 10.1007/978-1-4612-2256-9.

[26] P.H. Ginsparg, *Applied conformal field theory*, in *Les Houches Summer School in Theoretical Physics: Fields, Strings, Critical Phenomena*, 9, 1988 [arXiv:hep-th/9108028].

[27] A.A. Belavin, A.M. Polyakov and A.B. Zamolodchikov, *Infinite conformal symmetry in two-dimensional quantum field theory*, *Nucl. Phys. B* **241** (1984) 333.

[28] D. Poland, S. Rychkov and A. Vichi, *The conformal bootstrap: Theory, numerical techniques, and applications*, *Rev. Mod. Phys.* **91** (2019) 015002 [arXiv:1805.04405].

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

[30] A. Sterdyniak, A. Chandran, N. Regnault, B.A. Bernevig and P. Bonderson, *Real-space entanglement spectrum of quantum Hall states*, *Phys. Rev. B* **85** (2012) 125308 [arXiv:1110.2810].