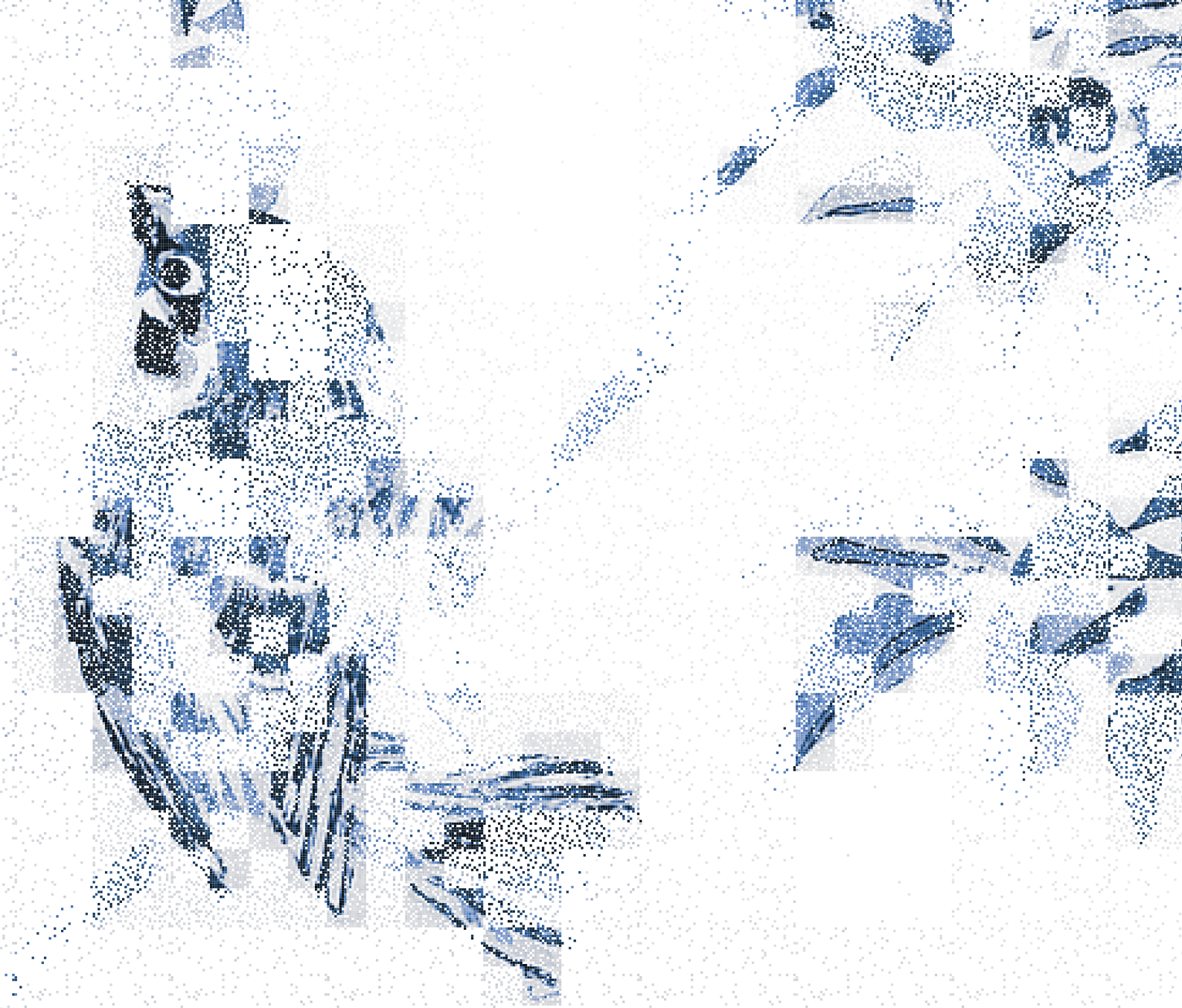


Connecting diagrams with tensor networks

Novel algorithms for many-body quantum systems

Marc K. Ritter

München, 23. Juni 2025

Connecting diagrams with tensor networks

Novel algorithms for many-body quantum systems

Marc K. Ritter

Dissertation
an der Fakultät für Physik
der Ludwig-Maximilians-Universität
München

vorgelegt von
Marc K. Ritter
aus Berlin

München, 23. Juni 2025

Erstgutachter: Prof. Dr. Jan von Delft
Zweitgutachter: Prof. Dr. Fabian Bohrdt geb. Grusdt
Datum der mündlichen Prüfung: 22. Juli 2025

Zusammenfassung

In der Vielteilchenquantenmechanik bezeichnet der Begriff des *Fluchs der Dimensionalität* das exponentielle Wachstum des Hilbertraums mit der Systemgröße. Numerische Methoden, die Quantenvielteilchensysteme simulieren, können anhand ihres Umgangs mit diesem Problem zwei Klassen zugeteilt werden: (A) Manche Methoden verwenden einen niedrigdimensionalen Ansatz, der aufgrund physikalischer Überlegungen gewählt wird. Ein Beispiel dafür sind Tensornetzwerkmethoden. (B) Andere Methoden umgehen dieses Problem, indem sie es vermeiden, quantenmechanische Zustände explizit darzustellen. Diese Arbeit beschäftigt sich mit Methoden beider Art.

Die Parquet-Gleichungen und die Funktionalrenormierungsgruppe, die im ersten Teil dieser Arbeit eingeführt werden, sind der Klasse (B) zuzuordnen. Statt expliziter Behandlung der Wellenfunktionen verwenden diese Methoden exakte Zusammenhänge zwischen Korrelatoren, wodurch sie im Vergleich zur Klasse (A) weniger empfindlich gegenüber der Dimensionalität des Systems und unbeeinflusst vom Wachstum der Verschränkung mit Systemgröße sind. Dadurch ist es möglich, die Methode der mehrschleifigen Pseudofermionen-Funktionalrenormierungsgruppe auf das J_1–J_2–J_3–Heisenbergmodell auf einem kubischen Gitter anzuwenden. Die Implementierung der Mehrschleifen-Gleichungen erfordert einige Verbesserungen des vorherigen Stands der Technik, die ebenfalls im ersten Teil vorgestellt werden.

Der zweite Teil der Arbeit befasst sich mit Tensornetzwerkmethoden. Diese gehören Klasse (A) an, da sie einen Ansatz für die Wellenfunktion wählen, etwa einen Matrixproduktzustand. Dieser Ansatz lässt sich auf beliebige Funktionen vieler Variablen verallgemeinern, und wird dann Tensorzug genannt. Tensorzüge erzeugen wir mittels der Tensor Cross Interpolation, eines Algorithmus des aktiven maschinellen Lernens, der die Zielfunktion nur stichprobenartig auswertet. Die Anzahl der benötigten Funktionsauswertungen skaliert hier mit der Größe der komprimierten Darstellung, die viel kleiner sein kann als die unkomprimierte Darstellung auf einem dichten Gitter. So wird die exponentiell teure Auswertung der Funktion auf allen Gitterpunkten vermieden. Besonders nützlich ist die Verbindung der Tensorzugdarstellung mit der Quantics-Darstellung, in der die Funktion durch die binären Ziffern der Funktionsargumente parametrisiert wird. Falls die Funktion kompressibel ist, skaliert der Rechenaufwand für die Erzeugung und Verarbeitung ihrer komprimierten Darstellung logarithmisch mit der Größe des Gitters. Operationen mit Funktionen in Tensorzugdarstellung, beispielsweise Multiplikation, Fourier-Transformation oder Faltung, können mit etablierten Tensornetzwerkmethoden aus der Vielteilchenquantenmechanik ausgewertet werden.

Somit ist es möglich, ganze Simulationsalgorithmen aus der Vielteilchenquantenmechanik in der Quantics-Tensorzugdarstellung durchzuführen. Der dritte Teil dieser Arbeit stellt eine so implementierte Selbstkonsistenziteration zum Lösen der Parquetgleichungen vor, und verbindet damit die Ansätze der

Klassen (A) und (B). Anwendung auf das Hubbard-Atom und das Anderson-Störstellenmodell zeigen, dass dieser Algorithmus so schnell konvergiert wie eine Implementierung mit dichten Gittern, dabei aber sehr viel weniger Arbeitsspeicher benötigt. Dieses Ergebnis zeigt auf, wie komplexere Modelle mit mehreren Orbitalen und Impulsabhängigkeiten mittels der Parquetgleichungen simuliert werden können, was bisher unerreicht ist.

Abstract

In the field of many-body quantum physics, the term *curse of dimensionality* refers to the exponential growth of the Hilbert space with system size. Numerical methods for simulating quantum many-body systems can be divided into two classes according to how they approach this problem: some methods (A) rely on a low-dimensional *ansatz* chosen for physical reasons, such as in tensor network methods, while others (B) avoid explicit representation of quantum mechanical states entirely, thus sidestepping the problem. This thesis is about methods in both classes.

The parquet equations and the functional renormalization group, introduced in the first part, both belong to class (B). Instead of computing wave functions, they rely on exact relations between correlators, which makes them less sensitive to the dimensionality of the system and agnostic of the scaling of the entanglement entropy compared to many methods of class (A). This enables us to apply the multiloop pseudofermion functional renormalization group scheme to the J_1–J_2–J_3 Heisenberg model on a cubic lattice. Implementing the multiloop equations requires numerous technical improvements over the previous state of the art, which are shown as well.

The second part of the thesis is based on tensor network methods. They belong to class (A), as they choose an ansatz, such as a matrix product state, for the wave function. This ansatz can be generalized to represent general functions of many variables, and is then called a tensor train. To construct a tensor train, we use the tensor cross interpolation algorithm, which samples the target function in an active machine learning scheme. The number of samples scales with the size of the compressed representation, which may be much smaller than the full tensor. Thus, the exponential cost of generating all components of the full tensor is avoided. The tensor train format is particularly useful when combined with the quantics representation, where a function is parameterized in a binary representation of its variables. Provided a function is compressible, the corresponding tensor train can then be constructed and operated on at a logarithmic cost in the number of discretization points. Operations on functions in tensor train representation, such as multiplication, Fourier transform, and convolution, can be evaluated using known tensor network algorithms that were originally formulated for many-body quantum physics.

Thus, quantics tensor trains offer enough versatility to perform entire many-body physics algorithms within this format. The third part of this thesis presents a self-consistent parquet solver implemented in this way, thus combining approaches of class (A) and (B). Benchmarks on the Hubbard atom and single-impurity Anderson models show that the quantics tensor trains converge as quickly as a solver based on dense grids, with much smaller memory consumption. This puts simulation of more complex models including multiple orbitals and momentum dependence with the parquet equations, which had hitherto been unfeasible, within reach.

Acknowledgments

This PhD thesis is the result of several collaborative projects that would not have been possible without the contributions and support of numerous individuals. I am deeply grateful to my supervisor Jan von Delft, for the opportunity to pursue a doctorate in his group, for invaluable guidance since the beginning of my academic career, and for unwavering support during difficult times. He has shown me the beginning of this path, and I am curious to follow it further and experience what it has to offer. I have learned many things from him, not least many insights into effectively communicating research.

I would like to thank those who contributed to the various projects undertaken during my doctorate. Julian Thönniß and Fabian Kugler were instrumental in setting up the pffRG for spin systems, and together with Matthias Punk first sparked my interest in quantum magnetism. As this project developed, I had many inspiring discussions with Benedikt Schneider, Björn Sbierski, Gün Günal, Marcel Gievers, and Nepomuk Ritz. Many insights were gained in the collaboration with the Cologne-Würzburg pffRG project, namely Dominik Kiese, Tobias Müller, Ronny Thomale, and Simon Trebst.

The project on tensor trains was initiated by Yuriel Núñez Fernández and Xavier Waintal, and Hiroshi Shinaoka and Markus Wallerberger contributed many insights on the quantics representation as well as the idiosyncrasies of the Julia programming language. I would like to thank them not only for their scientific input, but also for the exceptionally productive and enjoyable collaboration. Olivier Parcollet likewise contributed a great amount to this project, and I am indebted to him for welcoming me at the Flatiron Institute during my pre-doctoral stay. My time in New York widened my perspective on research in our field, not least through many inspiring discussions with Miles Stoudenmire, Jason Kaye, Fabian Kugler, and Dominik Kiese.

For the great collaboration on the QTT diagrammatics projects, I thank Stefan Rohshap, Anna Kauch, Hiroshi Shinaoka and Markus Wallerberger. It is always a great pleasure to go to Vienna, and I had many inspiring discussions there with Karsten Held, Samuel Badr, and Frederic Bippus. I also thank Markus Frankenbach, Nepomuk Ritz, and Gianluca Grosso for the productive collaboration on ongoing projects.

On a personal note, I am deeply grateful to my friends Sasha, Marcel, and Ming, for their warm friendship, steady support, and their faith in my abilities. Their strength enabled me to continue in the difficult phases of this doctorate. I would also like to thank my parents, particularly for their support during Covid times.

I would like to thank everyone at the von Delft chair in Munich for the great environment and pleasant atmosphere, especially my current and former office mates Sasha Kovalska, Marcel Gievers, Ming Huang, Carlo Bellinati, Mathias Pelz and Jheng-Wei Li. Special thanks go to Jan, Sasha, Nepomuk, and Benedikt, who provided valuable feedback on this manuscript.

Publications

This dissertation is based on the following articles:

P1 *Benchmark calculations of multiloop pseudofermion fRG*,
Marc K. Ritter, Dominik Kiese, Tobias Müller, Fabian B. Kugler, Ronny Thomale, Simon Trebst, and Jan von Delft,
The European Physical Journal B **95**, 102 (2022),
doi:10.1140/epjb/s10051-022-00349-2.
Reprinted on pages 14–28.

P2 *Quantics tensor cross interpolation for high-resolution parsimonious representations of multivariate functions*,
Marc K. Ritter, Yuriel Núñez Fernández, Markus Wallerberger, Jan von Delft, Hiroshi Shinaoka, and Xavier Waintal,
Physical Review Letters **132**, 056501 (2024),
doi:10.1103/PhysRevLett.132.056501.
Reprinted on pages 36–43.

P3 *Learning tensor networks with tensor cross interpolation: new algorithms and libraries*,
Yuriel Núñez Fernández*, Marc K. Ritter*, Matthieu Jeannin, Jheng-Wei Li, Thomas Kloss, Thibaud Louvet, Satoshi Terasaki, Olivier Parcollet, Jan von Delft, Hiroshi Shinaoka, and Xavier Waintal,
SciPost Physics **18**, 104 (2025),
doi:10.21468/SciPostPhys.18.3.104.
Reprinted on pages 43–118.
* Equal contributions (see page 54 of publication 3).

P4 *Two-particle calculations with quantics tensor trains – solving the parquet equations*,
Stefan Rohshap, Marc K. Ritter, Hiroshi Shinaoka, Jan von Delft, Markus Wallerberger, and Anna Kauch,
Physical Review Research **7**, 023087,
doi:10.1103/PhysRevResearch.7.023087.
Reprinted on pages 120–142.

P5 *Computing and compressing local vertex functions in imaginary and real frequencies from the multipoint numerical renormalization group using quantics tensor cross interpolation*,
Markus Frankenbach, Marc K. Ritter, Mathias Pelz, Nepomuk Ritz, Jan von Delft, and Anxiang Ge,
to be submitted to Physical Review Research,
doi:10.48550/arXiv.2506.13359.
Reprinted on pages 142–161.

These publications are reprinted in their original form, resized to fit ISO A4 format.

Contents

Chapter One

Introduction

'More is different', wrote Anderson in his seminal essay against reductionism [1]. What he expressed very succinctly is the principle of emergence: A system comprised of many parts may behave in ways that are very different to the behavior one of its constituents shows in isolation. Such emergent phenomena in systems comprised of quantum particles are the main object of study of quantum many-body physics. In some particularly fascinating cases, an emergent macroscopic phenomenon necessarily requires the underlying, microscopic constituents to be quantum objects. Thus, the quantum mechanical nature of atoms and electrons leaves a uniquely identifiable, macroscopic fingerprint. Well-known instances of such emergent phenomena in condensed matter physics are superconductivity and magnetism.

A paradigmatic example from field of quantum magnetism are quantum spin liquids (QSL) [2]. It has been hypothesized for a long time that certain systems of quantum spins may show interesting quantum mechanical entanglement in the ground state. This can be induced by frustrated interactions, i.e. interactions that do not permit a classical state to be locally optimal everywhere at the same time. The prototypical example is an antiferromagnetic interaction on a triangular lattice: there is no configuration of classical spins on a triangle in which all neighbors are antiparallel. Although the ground state of this model in particular turns out not to be a QSL, slight variations of the interactions or the lattice do induce QSL physics. Characteristic for QSL are long-range entanglement between spins, and collective excitations described by an emergent lattice gauge theory. When testing real materials for QSL states, it is very difficult to probe these properties. It is therefore necessary to identify model systems that host a stable QSL phase, and to predict observables in that phase on the theory side.

Precisely the characteristic properties of a QSL system also pose difficulties when studying these systems with numerical methods. Long-range correlations and entanglement are problematic for both exact diagonalization and tensor network methods. The former are very limited in the system sizes that can be

accessed [3–5], the latter rely on area-law scaling of the entanglement entropy for efficient representation of the wave function [6–8]. In Monte Carlo methods, frustrated interactions often induce a sign problem, leading to extremely slow convergence of results with the number of Monte Carlo samples [9, 10]. These difficulties can be circumvented using methods that rely on exact relations between correlators, such as the parquet equations [11, 12] and the closely related functional renormalization group [13, 14]. Chapter 2 shows how to set up these schemes for a lattice of quantum spins, and an application to the J_1–J_2–J_3 Heisenberg model on the cubic lattice in publication [P1].

Though the correlators are much simpler to represent than full wave functions, four-point and higher correlators still have complicated dependencies on multiple frequency, momentum, spin or orbital variables in general. For the Heisenberg model considered in publication [P1], the four-point vertex has one site index, one spin index, and three frequency indices. There, the vertex was parameterized on a specially designed adaptive discretization grid. When implementing these methods for other models, considerable effort is required to design such specific solutions. A more flexible and adaptable approach to representing correlators is therefore highly desirable.

Here, we take inspiration from a different class of numerical methods in condensed matter physics: tensor networks. The celebrated density matrix renormalization group (DMRG) algorithm [15, 16] relies on a matrix product state representation of quantum states [8]. This efficient representation exploits the area-law entanglement structure of the state: Partitioning the system into two subsystems, the entanglement entropy obtained from the reduced density matrix of one subsystem scales with the area of the partition, not the volume of the subsystem [8]. The underlying linear algebra structure can be applied more generally to represent functions of many variables, also if they are not wave functions in a Hilbert space.

In the applied mathematics community, this structure is called a *tensor train* (TT) [17]. The most obvious way to construct a TT is by repeated singular value decomposition of a dense tensor representation of a function. The amount of memory necessary to represent this dense tensor scales exponentially with the number of tensor legs, even in cases where it is strongly compressible. This curse of dimensionality can be avoided by constructing the TT with the *tensor cross interpolation* (TCI) algorithm, which optimizes the TT iteratively based on samples of the original tensor [17]. Being an active machine learning algorithm, the number of samples required may be much smaller than the size of the original tensor, depending on its structure [P2, P3, 18–22]. With an analogous shift of perspective, many of the established tensor network algorithms for wave functions can be generalized to operations on tensor train representations of functions [P3]. The aforementioned DMRG, for instance, can then be considered an algorithm to find the lowest eigenvalue and the corresponding eigenvector of a Hermitian linear operator.

When dealing with continuous functions, there are many ways of converting the function to a tensor with discrete indices. For functions of many variables with moderate resolution requirements in each variable, the domain may be discretized in each variable separately, and the tensor indices defined as simply enumerating the discretization points. For functions of few variables, where high resolution is required in each variable, it is advantageous to use the so-called *quantics* representation, where each index corresponds to a binary digit of the discretized variable [18]. If the function in question is compressible in this representation, combining quantics and TCI can generate a corresponding TT at logarithmic cost with respect to the number of discretization points [P2, 18]. Functions that require a very fine resolution, yet are compressible due to their structure, are very common in physics, engineering, and other fields applying mathematical representations, such as turbulence in hydrodynamics [23–26], options pricing in financial mathematics [27], and orbitals in quantum chemistry [28]. These representations are presented in chapter 3, including detailed description of the involved algorithms on a technical level and various applications in two publications [P2, P3].

Correlators in quantum many-body physics, such as the 4-point vertex introduced earlier, are also representable as compressed quantics TT [29–32]. In combination with tensor network algorithms that correspond to operations such as basis transforms, integrals, and convolutions, the entire parquet formalism can be implemented in a TT-based solver, as shown in chapter 4 and publication [P4]. A step beyond the parquet approximation is to use a local vertex computed using dynamical mean-field theory (DMFT) as an input to the parquet equations, in a scheme known as DΓA [33, 34]. DΓA can equally be regarded as a diagrammatic extension of DMFT. Publication [P5] presents an implementation of the multipoint numerical renormalization group (mpNRG) scheme [35–37], which directly obtains the vertex in QTT format.

Chapter Two

Functional renormalization group methods for spin systems

2.1 Introduction

Condensed matter systems showing emergent macroscopic excitations that can only be described as collective quantum mechanical behavior are one of the fascinating consequences of the inherently quantum mechanical nature of electrons. One example for such systems are the proposed quantum spin liquid (QSL) phases in quantum magnetism. Simulating QSL systems is challenging due to several of its characteristic properties: The frustrated interactions that are typical for these systems induce a sign problem in quantum Monte Carlo methods, which leads to slow convergence [3, 9, 10]. The entanglement entropy of QSL states scales with volume, not surface area. This poses a problem for the density matrix renormalization group (DMRG), since it relies on area-law scaling of entanglement for its compressed representation of states [6, 8]. One way to sidestep these problems is to work with exact relations of correlators, and thereby avoid explicit representation of the wave function. This is the general strategy of the parquet approach [11, 12], and the closely related functional renormalization group (fRG) [13, 14].

This chapter introduces the parquet equations in Sec. 2.2 and uses them as a starting point to derive fRG for in Sec. 2.3. Different ways of applying fRG to spin systems, with emphasis on the pseudofermion fRG (pffRG) are presented in Sec. 2.4. Thereafter, publication [P1] explains how to implement pffRG in an error-controlled way to obtain reproducible fRG flows. It is followed by a discussion of the strengths and limitations of the approach in Sec. 2.6.

2.2 Parquet equations

2.2.1 Fermionic action and correlators

Consider a fermionic action

$$S[\bar{\psi},\psi] = -\sum_{1',1} \bar{\psi}_{1'} \left[G_0\right]^{-1}(1',1)\,\psi_1 - \tfrac{1}{4}\sum_{1',2',1,2} \Gamma_0(1',2',1,2)\,\bar{\psi}_{1'}\bar{\psi}_{2'}\psi_2\psi_1, \quad (2.1)$$

where $\bar{\psi}$ and ψ are Grassmann variables and the indices $1', 2', 1, 2$ subsume parameters such as imaginary time or frequency, position or momentum arguments, and orbital indices. Here, the model is defined through the bare propagator G_0 and the bare two-particle interaction Γ_0, closely related to the interaction term in the Hamiltonian. In graphical notation, these are represented as a dashed line and a filled square, respectively:

$$G_0(1',1) = \quad ; \qquad \Gamma_0(1',2',1,2) = \quad .$$

Time-ordered correlators can be expressed through a path integral

$$\langle \bar{\psi}_{1'} \ldots \bar{\psi}_{n'} \psi_1 \ldots \psi_n \rangle = \frac{\int D[\bar{\psi},\psi]\, \bar{\psi}_{1'} \ldots \bar{\psi}_{n'}\, \psi_1 \ldots \psi_n\, e^{-S[\bar{\psi},\psi]}}{\int D[\bar{\psi},\psi]\, e^{-S[\bar{\psi},\psi]}}. \quad (2.2)$$

The two simplest such correlators are

1. the two-point correlator,

$$\langle \bar{\psi}_{1'} \psi_1 \rangle \equiv G(1',1) \equiv \quad , \quad (2.3)$$

   denoted as an arrow connecting its two indices $1', 1$ in diagrammatic notation, and
2. the four-point correlator,

$$\langle \bar{\psi}_{1'} \bar{\psi}_{2'} \psi_1 \psi_2 \rangle \equiv G^{(4)}(1',2',1,2) \equiv \quad , \quad (2.4)$$

   denoted as an *astroid* or *flying squirrel* shape in diagrammatic notation, with its four indices $1', 2', 1$, and 2 on its four corners [38].

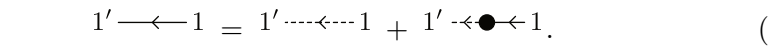
These correlators are the building blocks of a set of self-consistent relations, the so-called parquet equations [11, 12], which we will recapitulate in this section. To this end, we decompose them into trivial and correlated parts. Dyson's equation decomposes the propagator G into the bare propagator G_0 and a part related to the self energy Σ,

$$G(1',1) = G_0(1',1) + \sum_{2,3} G_0(1',2)\,\Sigma(2,3)\,G(3,1). \quad (2.5)$$

In diagrammatic notation, Σ is denoted with a circle,

$\Sigma(1', 1) \;=$. (2.6)

Thus, Dyson's equation can be expressed as

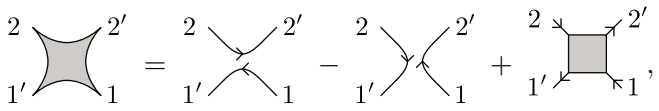
 . (2.7)

Points at which two correlators meet denote a contraction over an index: a frequency integral, a summation over Matsubara indices, or other summations and integrals as appropriate for the respective variable.

The four-point correlator can be decomposed into three parts in a so-called tree expansion,

$$G^{(4)}(1', 2', 1, 2) = G(1', 1)\, G(2', 2) - G(1', 2)\, G(2', 1) + \sum_{3,4,5,6} G(1', 3) G(2', 4)\, \Gamma(3, 4, 5, 6)\, G(5, 1)\, G(6, 2), \quad (2.8a)$$

which defines the four-point vertex Γ as the fully connected part of the four-point correlator, with amputated legs. In graphical notation, this is

 , (2.8b)

where the vertex Γ is denoted as a square.

2.2.2 Two-particle reducibility and the parquet equation

The vertex Γ can be expanded further in a perturbation series in the bare interaction Γ_0, which can be depicted using diagrams. We now organize all diagrams in this series according to their two-particle reducibility: A diagram is two-particle reducible if cutting two propagators results in two disconnected parts.[1] Each of the two resulting disconnected parts is attached to a pair of external legs, and there are three distinct divisions of the four legs into two pairs. We group each diagram into one of three classes based on this property [12, 39].

a-channel: Cutting two *anti-parallel* lines of an a-channel diagram results in disconnected parts connected to external legs $(1', 2)$ and $(2', 1)$, respectively. The sum of all diagrams in the a-channel is the *a-reducible vertex*, γ_a.

p-channel: Similarly, cutting two *parallel* lines of an p-channel diagram results in disconnected parts connected to external legs $(1', 2')$ and $(1, 2)$, and the sum of these diagrams is the p-reducible vertex, γ_p.

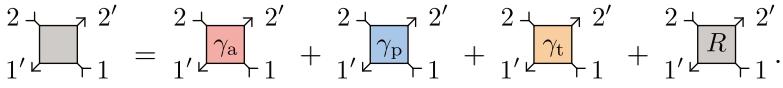
[1] In discrete mathematics terms, a two-particle reducible diagram is a 2-edge-connected graph.

t-channel: Diagrams reducible in the t-channel are disconnected when cutting two *transverse* lines, and the disconnected parts are connected to external legs $(1', 1)$ and $(2', 2)$, respectively. Their sum is the t-reducible vertex, γ_t.

No diagram can be two-particle reducible in more than one channel, but there are diagrams that are not reducible in any of the three channels [12]. These are called *two-particle irreducible*, and their sum is denoted R. Since the classification by reducibility assigns each diagram to exactly one class, the sum of all three reducible vertices and R then recovers the full vertex Γ,

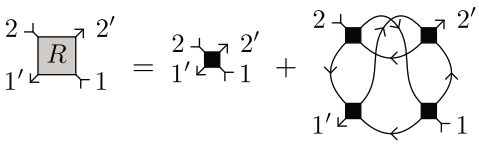
$$\Gamma = \gamma_a + \gamma_p + \gamma_t + R, \tag{2.9a}$$

or, in graphical notation,



(2.9b)

This equation is known as the *parquet decomposition* or as *the* parquet equation. The two-particle irreducible vertex R contains the bare vertex Γ_0, and further diagrams starting with the so-called *envelope diagram* at fourth order in the interaction [39]:



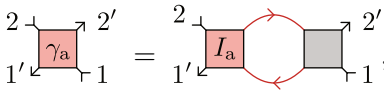
(2.10)

2.2.3 Bethe–Salpeter equations

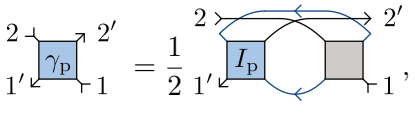
Each of the three reducibility classes $r \in \{\mathrm{a}, \mathrm{p}, \mathrm{t}\}$ in the parquet equation fulfills a corresponding Bethe–Salpeter equation (BSE) of the form

$$\gamma_r = \Gamma \circ \Pi_r \circ I_r, \tag{2.11}$$

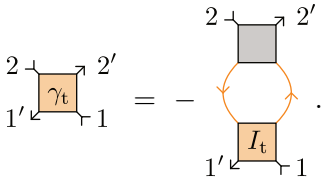
where Π_r is a propagator bubble in channel r, $I_r = \Gamma - \gamma_r$, and the products $\circ$ are to be understood as channel-specific matrix products [40]. In graphical notation,



(2.12a)



(2.12b)

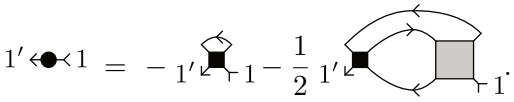


(2.12c)

The propagator bubble in each equation corresponds to a frequency integral or Matsubara frequency sum, and evaluation of these integrals or sums accounts for the majority of computational effort required to solve the parquet equations.

2.2.4 Schwinger–Dyson equation

The self energy Σ is related to the vertex Γ via the Schwinger–Dyson equation,



(2.13)

Thus, the self energy Σ can be obtained from a known vertex Γ, and in turn, the propagator G obtained from Σ via Dyson's equation (2.5). This newly obtained propagator can then be inserted into the Bethe–Salpeter equations (2.12) and the parquet decomposition (2.9) to obtain a new vertex Γ, thus forming a set of exact self-consistent relations known as the parquet equations. This set is complete except for one quantity: the irreducible vertex R.

2.2.5 The irreducible vertex R

The irreducible vertex R cannot be obtained from the parquet equations, and is considered an external input. One common approximation is the so-called *parquet approximation*, which sets

$$R = \Gamma_0, \tag{2.14}$$

thereby neglecting all non-trivial irreducible diagrams, starting with the envelope diagram. Alternatively, it is possible to use an external input for R, such as a local vertex obtained from DMFT, in an approach known as *dynamical vertex approximation* (DΓA) [34, 41]. Thus, the parquet approach can be used as a diagrammatic extension of DMFT, similar to DMFT+GW [42] and related approaches [33].

With an input for R, a self-consistent solution to the parquet equations can be found in principle. In practice, it is a often very difficult to converge to a solution. Even if convergence is reached, there is no guarantee that the solution is physical. Indeed, it is known that self-consistent systems of diagrammatic equations may have multiple branches, which allows convergence to unphysical solutions [43]. Recently, some techniques to force convergence to the physical branch have been developed, which are outside the scope of this work [44].

2.3 Functional renormalization group

2.3.1 Introduction

Instead of solving the parquet equations using a self-consistency iteration or a root-finding algorithm, it is possible to use the known solution of a simpler

reference system, and then transport that solution to the target system. This is the idea of the *functional renormalization group* (fRG) approach. There, a flow parameter Λ interpolates between a reference and a target system. Taking the derivative with respect to Λ of the parquet equations generates a set of equations, the so-called *flow equations*, that describe how the solution evolves with Λ. Integrating these equations then allows us to transport solutions from the reference to the target system.[2]

This is a somewhat unorthodox way of motivating fRG. Usually, the fRG is derived from a generating functional approach [14], which yields equivalent flow equations except for so-called multiloop terms. Historically, the multiloop terms were introduced as extensions of the fRG, with the goal of enforcing certain properties that the exact solution is known to fulfill, among them the parquet equations and the Mermin–Wagner theorem [45, 46].

2.3.2 Flow parameter and regulator

There are many ways of introducing a flow parameter Λ to the parquet equations; in this thesis, it is chosen to be an artificial energy cutoff in the bare Green's function

$$G_0^\Lambda(\omega) = \Theta^\Lambda(\omega)\, G_0(\omega) \tag{2.15}$$

where we choose a smooth cutoff function for the *regulator* Θ,

$$\Theta^\Lambda(\omega) = 1 - e^{-\omega^2/\Lambda^2}. \tag{2.16}$$

It is worth noting that there is considerable freedom in these choices. Others have used different regulators, including the sharp heaviside function $\Theta^\Lambda(\omega) = \theta(\omega - \Lambda)$ [47]. The flow parameter does not necessarily have to be an energy cutoff; other parameters, such as an artificial momentum cutoff or even physical parameters such as the temperature have been used as flow parameters in other contexts [48, 49]. The interpretation of an fRG flow as an interpolation between a reference and target system is perhaps most intuitive if a physical parameter is used as flow parameter. In combination with some approximations to the fRG flow, the choice of regulator may in general affect the physical observables at the end of the fRG flow. This is a major limitation of this method, which can be mitigated in part using the multiloop flow described below.

With a cutoff-dependent propagator, the self-energy Σ, the full propagator G, as well as the two-particle quantities Γ and γ_r become cutoff-dependent as well. If $\Lambda \to \infty$, i.e. Λ is much larger than all energy scales in the system, all quantities are equal to those of the reference system. For our choice of regulator (2.16), the bare propagator vanishes. This in turn implies that all

[2] The more recent *finite difference parquet* approach by Lihm et al. [37] implements this idea more directly, by considering the difference between the two-particle quantities describing a reference and target system.

two-particle reducible vertices γ_r vanish, and therefore $\Gamma = R$, which is equal to Γ_0 in the parquet approximation. This is the starting point of the fRG flow. An alternative is to use a local vertex and self energy obtained from DMFT as reference system, in an approach known as DMF^{2}RG [50].

2.3.3 Flow equations

The flow equations specify how to transport this solution to smaller values of Λ. These equations are obtained by inserting the propagator G^Λ into the parquet equations (2.9), (2.12) and (2.13), then taking the derivative with respect to Λ. We denote these derivatives by adding a dot to the corresponding quantity, i.e., $\dot{X} = \frac{\partial}{\partial\Lambda} X$. After some algebraic manipulation (see Refs. [13, 51]), we obtain the flow equations for the reducible vertices as a series of contributions ordered by loop order ℓ:

$$\dot{\gamma}_\mathrm{r} = \sum_{\ell=1}^{\infty} \dot{\gamma}_\mathrm{r}^{(\ell)}. \tag{2.17a}$$

The first term is the one-loop contribution, which correpsonds to the flow equations in fRG derived from the generating functional approach [14],

$$\dot{\gamma}_\mathrm{r}^{(1)} = \Gamma \circ \dot{\Pi}_r \circ \Gamma. \tag{2.17b}$$

Again, the matrix products should be understood as sums over shared indices in accordance with channel r. All further terms are defined recursively in terms of lower order contributions,

$$\dot{\gamma}_\mathrm{r}^{(2)} = \dot{\gamma}_{\bar{\mathrm{r}}}^{(1)} \circ \Pi_r \circ \Gamma + \Gamma \circ \Pi_r \circ \dot{\gamma}_{\bar{\mathrm{r}}}^{(1)}, \quad \text{where } \dot{\gamma}_{\bar{r}}^{(\ell)} = \textstyle\sum_{r' \neq r} \dot{\gamma}_{r'}^{(\ell)}, \tag{2.17c}$$

$$\dot{\gamma}_\mathrm{r}^{(\ell)} = \dot{\gamma}_{\bar{\mathrm{r}}}^{(\ell-1)} \circ \Pi_r \circ \Gamma + \Gamma \circ \Pi_r \circ \dot{\gamma}_{\bar{\mathrm{r}}}^{(\ell-2)} \circ \Pi_r \circ \Gamma + \Gamma \circ \Pi_r \circ \dot{\gamma}_{\bar{\mathrm{r}}}^{(\ell-1)}, \ell \geq 3. \tag{2.17d}$$

Whereas the series in Eq. (2.17a) is infinite, it has to be truncated at some finite order ℓ_max in practice. These terms can be evaluated efficiently in order from $\ell = 1$ to ℓ_max, inserting results for smaller ℓ in the equation for larger ℓ.

Analogously, the self energy flow is obtained from a derivative of the Schwinger–Dyson equation. After some algebra, this can be shown to be [51]

$$\begin{aligned}\dot{\Sigma}(1',1) = &-\sum_{2,3} \Gamma(1',2;1,3)\, \dot{G}(3,2) \overbrace{-\sum_{2,3} \dot{\gamma}_{\bar{t},C}(1',2;1,3)\, G(3,2)}^{\dot{\Sigma}_{\bar{t}}(1',1)} \\ &- \sum_{2,3,4,5} \Gamma(1',2;1,5)\, G(3,2)\, \dot{\Sigma}_{\bar{t}}(4,3)\, G(5,4),\end{aligned} \tag{2.18}$$

where $\dot{\gamma}_{\bar{t},C} = \sum_{r\in\{\mathrm{a,p}\}} \Gamma \circ \Pi_r \circ \dot{\gamma}_r \circ \Pi_r \circ \Gamma$. The standard (one-loop) fRG flow contains only the first term, and the other two terms are only non-zero for $\ell_\mathrm{max} \geq 3$.

Integrating the flow equations (2.17) and (2.18) from the large initial value of Λ to some final $\Lambda \to 0$ then gives a solution for the physical model.[3] If there is a phase transition to an ordered phase at some value of Λ_{crit}, this will induce a divergence in the corresponding susceptibility. The continuous fRG flow can only approach Λ_{crit} asymptotically. Close to the transition, the diverging components of generalized two-particle susceptibilities of the form $\chi \sim \langle \bar{\psi}\bar{\psi}\psi\psi \rangle$ correspond to the order parameter of the phase behind the transition point. This allows identifying the ordered phase without explicitly crossing the transition.[4] There is also an alternative approach described in Ref. [55], where a Hubbard–Stratonovich transform corresponding to the unstable susceptibility component allows continuation of the transformed flow into the ordered phase.

2.3.4 Limitations and scope of applicability

In practice, it is not always possible to reach convergence in loop order ℓ_{max} due to excessive requirements on memory and computation time. Even at convergence, there are some more general limitations to this approach. As we remarked earlier, the parquet equation may have multiple solution branches, and a solution obtained from fRG is not guaranteed to be on the physical branch. If the parquet approximation, i.e. $R = \Gamma_0$, is used, fRG neglects some diagrams starting at fourth order in the interaction, and this truncation of the diagrammatic series is only justified at small interaction strengths. At strong interactions, fRG results may not reproduce the correct behavior [56]. It is also not clear whether truncating at higher loop orders $\ell_{\text{max}} > 1$ is generally beneficial: for the x-ray edge problem, Diekmann and Jakobs [57] argue that the one-loop flow contains exactly the leading order of divergences in the particle-hole susceptibility, and all further loop orders constitute comparatively arbitrary truncations of the diagrammatic series. On the other hand, Gievers [38] recently demonstrated that subleading logarithmic corrections are necessary to reproduce certain power laws in the particle-hole susceptibility. For other models, where no such arguments are known, the general philopsophy is to fulfill as many exact relations as possible by including these correction terms to the extent that is feasible at a tolerable numerical cost.

The strength of fRG, on the other hand, mainly lies in a great flexibility regarding the systems it can be applied to. Symmetries in the Hamiltonian directly induce symmetries in the vertices, which can be exploited for symmetry

[3] In principle, the final value of the flow parameter is $\Lambda = 0$. In some cases, such as for the pseudofermion Hamiltonian discussed later in this chapter, there is a divergence $\sim 1/\Lambda$ in some susceptibilities, which prevents reaching exact $\Lambda = 0$. In this case, a small value of Λ below all other energy scales of the system is sufficient to obtain all relevant physics.

[4] There are phases with higher-order order parameters that cannot be identified using divergences in a two-particle susceptibility, such as the nematic order parameter in Ref. [52]. These phases can be identified indirectly by adding perturbations to the Hamiltonian, and observing the effect of these perturbations on the two-particle susceptibilities [53, 54].

reduction. For the pseudofermion systems discussed further below, this means large lattices are accessible at comparatively small computational cost [58]. fRG does not impose any inherent restriction on the dimensionality of the system, unlike tensor network methods. Two- and three-dimensional lattices are thus accessible, though at an increased computational cost. Frustrated interactions, which induce sign problems in Quantum Monte Carlo methods, are unproblematic as well. Since wave functions are not represented explicitly, the method does not have specific requirements on their structure, such as the area-law scaling of the entanglement that DMRG relies on. This means fRG can be applied to systems at quantum critical points, or indeed in phases where a volume-law scaling is present. Quantum spin liquids are perhaps the most prominent example of such wave functions [2].

2.4 Functional renormalization group for spin systems

Given these strengths, it seems natural to study frustrated spin systems using fRG, targeting proposed quantum spin liquid phases. This is the idea of the pseudofermion fRG (pffRG), which was originally formulated by Reuther and Wölfle [58] (see also the review in Ref. [59]). We consider Hamiltonians of spin-1/2 systems with two-spin interactions of the form

$$H = \sum_{i,j} \sum_{\mu,\nu\in\{\mathrm{x,y,z}\}} J_{ij}^{\mu\nu} S_i^\mu S_j^\nu, \tag{2.19}$$

where S_i^μ is the μ component of the spin operator at site i. Since our derivation of the fRG flow relies on a functional integral,[5] we represent the spin operators in terms of pseudofermions $f_{i\alpha}$ using Abrikosov's construction [61],

$$S_i^\mu = \tfrac{1}{2} \sum_{\alpha\beta} \sigma_{\alpha\beta}^\mu f_{i\alpha}^\dagger f_{i\beta}, \tag{2.20}$$

where σ^μ is the μth Pauli matrix. This is a faithful representation of the spin operator if

$$\sum_\alpha f_{i\alpha}^\dagger f_{i\alpha} = 1 \ \forall i. \tag{2.21}$$

This constraint is necessary to restrict local Hilbert space of the pseudofermions, which has dimension 4, to the two-dimensional subspace of states that corresponds to the physical spin states. In this representation, the Hamiltonian is

$$H = \tfrac{1}{4} \sum_{i,j} \sum_{\mu,\nu\in\{\mathrm{x,y,z}\}} J_{ij}^{\mu\nu} \sum_{\alpha\beta\gamma\delta} \sigma_{\alpha\beta}^\mu \sigma_{\gamma\delta}^\nu f_{i\alpha}^\dagger f_{i\beta} f_{j\gamma}^\dagger f_{j\delta}. \tag{2.22}$$

[5] There is an alternative derivation of the fRG formalism for spin systems that does not rely on a functional integral, and encodes the spin algebra into more complicated initial conditions [60].

We construct an fRG flow in the imaginary frequency Matsubara formalism at zero temperature, using the same procedure as for usual fermions. Because there is no kinetic term, the fermions are immobile and the bare propagator is simply

$$G_0(i_1', i\omega_1', \alpha_1'; i_1, i\omega_1, \alpha_1) = (i\omega_1)^{-1}\delta_{i_1' i_1}\delta_{\alpha_1'\alpha_1}\delta(i\omega_1' - i\omega_1). \tag{2.23}$$

Physical observables are obtained at the end of the flow, at $\Lambda \to 0$, by translating the observable's expectation value to expectation values of the auxiliary operators.

Enforcing the pseudofermion constraint (2.21) is, in theory, possible using a method by Popov and Fedotov, effectively introducing a purely imaginary term into the action [62, 63]. The parquet approximation introduces inaccuracies to the fRG flow starting at fourth order in the interaction, which leads to a violation of the exact constraint by approximately 30%–40% at the end of the flow [64]. In our implementation, the constraint is only enforced on average, i.e., $\langle \sum_\alpha f_{i\alpha}^\dagger f_{i\alpha} \rangle = 1$. This generally leads to a similar violation of the exact constraint by approximately 30%–40% at the end of the flow [64, 65].

There is alternative way to reformulate the spin Hamiltonian for constructing a functional integral, which uses Majorana fermions as auxiliary particles [66]. A similar doubling of the Hilbert space is resolved differently there, in that the local space with dimension 4 is split into two disconnected copies of a physical two-dimensional Hilbert space [66]. At low temperatures, this artificial degeneracy, coupled with the truncation of the diagrammatic series in the fRG flow equations, causes unphysical divergence to appear in the correlators.

For our zero-temperature study of QSLs in the J_1–J_2–J_3 model on the cubic lattice, we therefore rely on pseudofermion fRG. Publication [P1] in this chapter is a detailed description of the technical aspects of implementing pseudofermion fRG for spin systems at zero temperature. In particular, it shows how stable and reproducible fRG flows can be reached through careful use of error-controlled numerical algorithms.

2.5 Publication 1: Benchmark calculations of multiloop pseudofermion fRG

In this section, the following publication is reprinted:

P1 *Benchmark calculations of multiloop pseudofermion fRG*,
Marc K. Ritter, Dominik Kiese, Tobias Müller, Fabian B. Kugler, Ronny Thomale, Simon Trebst, Jan von Delft,
The European Physical Journal B **95**, 102 (2022),
doi:10.1140/epjb/s10051-022-00349-2.
Reprinted on pages 14–28.

Eur. Phys. J. B (2022) 95:102
https://doi.org/10.1140/epjb/s10051-022-00349-2

THE EUROPEAN PHYSICAL JOURNAL B

Regular Article - Solid State and Materials

Benchmark calculations of multiloop pseudofermion fRG

Marc K. Ritter[1,a], Dominik Kiese[2], Tobias Müller[3], Fabian B. Kugler[4], Ronny Thomale[3], Simon Trebst[2], and Jan von Delft[1]

[1] Arnold Sommerfeld Center for Theoretical Physics, Center for NanoScience, and Munich Center for Quantum Science and Technology, Ludwig-Maximilians-Universität München, 80333 Munich, Germany
[2] Institute for Theoretical Physics, University of Cologne, 50937 Cologne, Germany
[3] Institute for Theoretical Physics, University of Würzburg, Am Hubland, 97074 Würzburg, Germany
[4] Department of Physics and Astronomy, Rutgers University, Piscataway, NJ 08854, USA

Received 30 March 2022 / Accepted 10 May 2022 / Published online 1 July 2022

Abstract. The pseudofermion functional renormalization group (pffRG) is a computational method for determining zero-temperature phase diagrams of frustrated quantum magnets. In a recent methodological advance, the commonly employed Katanin truncation of the flow equations was extended to include multiloop corrections, thereby capturing additional contributions from the three-particle vertex (Thoenniss et al. https://arxiv.org/abs/2011.01268; Kiese et al. https://arxiv.org/abs/2011.01269). This development has also stimulated significant progress in the numerical implementation of pffRG, allowing one to track the evolution of pseudofermion vertices under the renormalization group flow with unprecedented accuracy. However, cutting-edge solvers differ in their integration algorithms, heuristics to discretize Matsubara frequency grids, and more. To lend confidence in the numerical robustness of state-of-the-art multiloop pffRG codes, we present and compare results produced with two independently developed and algorithmically distinct solvers for Heisenberg models on three-dimensional lattice geometries. Using the cubic lattice Heisenberg (anti)ferromagnet with nearest and next-nearest neighbor interactions as a generic benchmark model, we find the two codes to quantitatively agree, often up to several orders of magnitude in digital precision, both on the level of spin-spin correlation functions and renormalized fermionic vertices for varying loop orders. These benchmark calculations further substantiate the usage of multiloop pffRG solvers to tackle unconventional forms of quantum magnetism.

1 Introduction

A fascinating phenomenon in the study of frustrated quantum magnets is the interplay of unconventional forms of magnetic order and the possible emergence of quantum spin liquid states near zero temperature [3]. The successful description of such low-energy states of quantum spin systems has, however, remained challenging, especially in the presence of competing interactions, geometric frustration, and in higher spatial dimensions.

Since its inception more than a decade ago [4], the pseudofermion functional renormalization group (pffRG) has become a powerful and flexible approach to map out the zero-temperature phase diagrams of various quantum spin models, both in two [4–20] and three spatial dimensions [16,21–29]. Although the problem obtained after representing the spin operators by complex fermions is treated approximately, one of the striking features of pffRG is its ability to track competing instabilities in different interaction channels, allowing one to discriminate putative spin-liquid phases from long-range ordered magnetic ground states. This ability can be traced back [30,31] to the inclusion of leading-order $1/S$ and $1/N$ diagrams (the former promoting classical magnetic order, the latter quantum fluctuations), which are treated on equal footing in pffRG by means of the routinely employed Katanin truncation [32].

Recently, the multiloop truncation scheme of the infinite hierarchy of fRG flow equations [33–35], previously used in the context of the Hubbard [36,37] and Anderson impurity model [38], was applied to the zero-temperature pffRG by some of us [1,2]. The convergence in the number of loops over a wide range of energy scales attested to the inner consistency of the pffRG method, despite being used in the strong-coupling limit. These developments were accompanied and facilitated by substantial improvements of the numerical implementation that remedy many shortcomings of previous studies. Yet, some of these advances, such as the employed integration routines and adaptive Matsubara frequency grids [1,2], rely on certain numerical heuristics, affecting, e.g., the minimal grid spacing and largest Matsubara frequencies considered. Therefore, quanti-

[a] e-mail: ritter.marc@physik.uni-muenchen.de (corresponding author)

tative agreement between different implementations is, although highly desired, not guaranteed *a priori*.

In the present work, we provide evidence for the numerical robustness of pffRG by benchmarking two independent state-of-the-art solvers, one provided by a research group at LMU Munich (dubbed code #1 in the following), and one by a Cologne–Würzburg collaboration (denoted by code #2) with an open-source release [39]. As a test case, we consider ferro- and antiferromagnetic Heisenberg models on the simple cubic lattice and compare our results both on the level of renormalized couplings (i.e. fermionic vertex functions) as well as for the (post-processed) spin-spin correlation functions.

The remainder of the paper is structured as follows. We begin by providing a brief overview of the multiloop pffRG in Sect. 2. This is followed by an in-depth comparison of the numerical results produced by the two codes at hand in Sect. 3. Finally, in Sect. 4, technical aspects of the implementation, such as the choice of frequency grids, integration routines and differential equation solvers are discussed, with special emphasis devoted to their influence on the numerical stability and accuracy of the two codes.

2 Multiloop pseudofermion fRG

Within the pffRG approach, one can study generic spin-1/2 Hamiltonians with bilinear spin couplings, i.e.,

$$\mathcal{H} = \tfrac{1}{2}\sum_{ij} J_{ij}^{\mu\nu} S_i^\mu S_j^\nu \,. \tag{1}$$

Here, the spin operators S_i^μ live on the sites i of an arbitrary lattice, and the exchange matrices $J_{ij}^{\mu\nu}$ are assumed to be real. The spin operators are represented in terms of complex pseudofermions $f_{i\alpha}^{(\dagger)}$ with $\alpha \in \{\uparrow,\downarrow\}$ as

$$S_i^\mu = \tfrac{1}{2}\sum_{\alpha,\beta} f_{i\alpha}^\dagger \sigma_{\alpha\beta}^\mu f_{i\beta} \,, \tag{2}$$

where $\sigma_{\alpha\beta}^\mu$ for $\mu \in \{x,y,z\}$ are the Pauli matrices. This yields a purely quartic Hamiltonian which can be treated by established functional RG techniques.

Note that the pseudofermion representation of the spin algebra comes with an artificial enlargement of the local Hilbert space dimension, which must be dealt with by an additional particle number constraint $\sum_\alpha f_{i\alpha}^\dagger f_{i\alpha} = 1$ on every lattice site. In practice, this constraint is not enforced, but holds on average due to particle-hole symmetry [1,2,4]. Nevertheless, the influence of fluctuations can be quantitatively gauged by explicitly computing the variance of the number operator, which can be expressed through the equal-time spin-spin correlation function $\langle S_i^\mu S_i^\mu\rangle$ [1]. Although fluctuations are not fully suppressed, even if a local level repulsion term $AS_i^\mu S_i^\mu$ (with $A<0$) is employed, recent studies [1,19,23,30] pointed out that observables extracted from pffRG flows are qualitatively unaffected by the unphysical Hilbert space sectors.

An alternate decomposition of the spin operators into Majorana instead of Abrikosov fermions allows one to circumvent the problem of unphysical states in the fermionic representation at the cost of redundant copies of physical Hilbert-space sectors [40]. For moderately high temperatures, the latter approach was recently shown to enable an accurate calculation of thermodynamic observables [41], such as the free energy and specific heat. However, the approach was also found to suffer from unphysical divergencies when approaching the $T \to 0$ limit, which we consider here (for the Abrikosov fermion decomposition).

Since kinetic contributions are absent in the pseudofermion representation of Eq. (1), the free propagator assumes the simple form

$$G_0(1'|1) = (i\omega_1)^{-1}\delta_{i_{1'}i_1}\delta_{\alpha_{1'}\alpha_1}\delta(\omega_{1'}-\omega_1)\,, \tag{3}$$

diagonal in all indices. To successively integrate out high-energy modes and thus provide an effective low-energy description of a given model, a cutoff parameter, here denoted as Λ, is introduced in the bare propagator. The fRG equations then govern the flow of the n-particle vertices from the UV limit $\Lambda \to \infty$, where the regularized bare propagator vanishes, to the infrared limit $\Lambda \to 0$, where one recovers the physical theory. As such, there is a certain degree of freedom in the cutoff implementation. A popular choice for the regulator in pffRG is a Heavyside step function, which sharply suppresses frequency contributions $|\omega| < \Lambda$. This choice is very useful for analytical treatments of pffRG in the large-S and large-N limit, where the flow equations can be solved exactly and reproduce mean-field gap equations [30,31]. However, if numerical calculations are employed away from these limits, a non-analytic regulator spoils the smoothness of the right-hand side of the flow equations, and therefore limits the applicability of higher-order integration routines. For this reason, we consider a smooth regulator

$$R^\Lambda(\omega) = 1 - e^{-\omega^2/\Lambda^2}\,, \tag{4}$$

throughout this manuscript, and implement the cutoff as $G_0^\Lambda(\omega) = R^\Lambda(\omega)G_0(\omega)$, with $G_0(\omega) \equiv (i\omega)^{-1}$.

To make the infinite hierarchy of fRG flow equations amenable to further calculations, a truncation is necessary. Usually, this is done by neglecting all n-particle vertices of $n=3$ and higher [32]. However, to capture the physics of interest in pffRG, one must already go beyond that using the Katanin truncation, which feeds the Λ derivative of the self-energy Σ^Λ back into the flow of the two-particle vertex Γ^Λ [4]. Within this truncation, the flow equations schematically read

$$\frac{d}{d\Lambda}\Sigma^\Lambda = -\left[\Gamma^\Lambda \circ S^\Lambda\right]_\Sigma \,, \tag{5}$$

$$\frac{d}{d\Lambda}\Gamma^\Lambda = \sum_c \dot{\gamma}_c^\Lambda = -\sum_c [\Gamma^\Lambda \circ \partial_\Lambda (G^\Lambda \times G^\Lambda) \circ \Gamma^\Lambda]_c \,. \tag{6}$$

Here, we introduced the loop function $[\Gamma \circ G]_\Sigma$ and the single-scale propagator $S^\Lambda \equiv -\frac{d}{d\Lambda} G^\Lambda|_{\Sigma^\Lambda = \text{const.}}$. We categorized the contributions to the flow of Γ into three distinct channels c: the particle-particle (s) channel, the direct particle-hole (t) channel, and the crossed particle-hole (u) channel. Each "bubble" term, with the general form $[\Gamma \circ (G \times G') \circ \Gamma']_c$, describes the flow of a two-particle reducible vertex γ_c. As all self-energies, vertices, and related correlators are Λ-dependent, we refrain from writing this dependence explicitly in the following.

The multiloop fRG (mfRG) flow [33–35], recently employed within pffRG [1,2], is an attempt to go beyond the Katanin truncation and capture even more contributions from n-particle vertices with $n \geq 3$. It can be derived from the parquet approximation [42], which self-consistently connects one- and two-particle correlation functions via the Schwinger–Dyson (SDE) and Bethe–Salpeter equations (BSE), and as such the inherent dependence of the $\Lambda \to 0$ fRG result on the specific choice of regulator is eliminated [34]. This approximation includes all those contributions to the flow of the two-particle vertex which can be efficiently calculated, i.e., with the same cost as the one-loop flow in Eqs. (5) and (6). Summarized briefly: To obtain the mfRG flow of γ_c, one iteratively computes multiloop corrections to the one-loop ($\ell = 1$) result, using bubble functions with undifferentiated propagators but differentiated vertices. In a similar fashion, one can recover equivalence to the SDE, by feeding back the so-determined vertex corrections into the self-energy flow.

One of the most important ingredients to achieve sufficient numerical accuracy throughout the multiloop flow is an appropriate treatment of the frequency dependence of the two-particle vertex. In Ref. [43], a parametrization in terms of one bosonic and two fermionic frequencies (the fourth frequency argument is fixed by energy conservation) for each two-particle reducible vertex was put forward. This parametrization captures the non-trivial high frequency asymptotics of the vertices while being numerically efficient. Code #1 uses precisely the proposal of Ref. [43], and the diagrams contributing to each channel are grouped into four asymptotic classes K_n as

$$\begin{aligned}\gamma_c(\omega_c, \nu_c, \nu_c') = {} & K_{1,c}(\omega_c) \\ & + K_{2,c}(\omega_c, \nu_c) + K_{2',c}(\omega_c, \nu_c') \\ & + K_{3,c}(\omega_c, \nu_c, \nu_c') \,,\end{aligned} \tag{7}$$

where we displayed only frequency arguments for brevity. Here, ω_c, ν_c and ν_c', denote the natural frequency arguments for diagrams reducible in channel c (see Ref. [1] for the conventions used). The K_n asymptotically decay to zero in each frequency, allowing one to reduce the necessary number of arguments when summing up the asymptotic classes to obtain γ_c. Code #2 chooses a slightly different approach, by defining asymptotic classes Q_n [44] as

$$\begin{aligned} Q_{1,c}(\omega_c) &= K_{1,c}(\omega_c) \\ Q_{2,c}(\omega_c, \nu_c) &= K_{1,c}(\omega_c) + K_{2,c}(\omega_c, \nu_c) \\ Q_{2',c}(\omega_c, \nu_c') &= K_{1,c}(\omega_c) + K_{2',c}(\omega_c, \nu_c') \\ Q_{3,c}(\omega_c, \nu_c, \nu_c') &= K_{1,c}(\omega_c) \\ &\quad + K_{2,c}(\omega_c, \nu_c) + K_{2',c}(\omega_c, \nu_c') \\ &\quad + K_{3,c}(\omega_c, \nu_c, \nu_c') \,, \end{aligned} \tag{8}$$

with the respective choice of natural frequency arguments outlined in Ref. [2]. Since the K_n classes decay to zero for large frequencies, the Q_n (at least for $n > 1$) are projected to a lower class. For instance, $Q_{3,c}(\omega_c, \nu_c, \nu_c') = Q_{2,c}(\omega_c, \nu_c)$ if $|\nu_c'| \to \infty$. Let us emphasize that both parametrizations contain the same information about the asymptotic structure of the two-particle vertices, as the K_n and Q_n parametrizations can be exactly transformed into each other. For an appropriate choice of numerical frequency grids, both parametrizations are therefore equally valid and differ only in numerical performance. The former approach allows for a more fine-grained adjustment of discrete frequencies to the asymptotic decay of individual classes, while the latter reduces the cost of evoking a two-particle vertex from a summation of up to four classes K_n to loading just a single Q_n.

The central observable computed from the pffRG equations is the flowing spin-spin correlation function,

$$\chi_{ij}^{\mu\nu}(i\omega = 0) = \int_0^\infty d\tau \langle T_\tau S_i^\mu(\tau) S_j^\nu(0) \rangle \,, \tag{9}$$

where we omit indication of the Λ-dependence for brevity. In all models considered here, the interactions in the Hamiltonian are diagonal and SU(2)-symmetric. This leads to spin-spin correlations that are symmetric as well, and we thus define $\chi_{ij} \equiv \chi_{ij}^{xx} = \chi_{ij}^{yy} = \chi_{ij}^{zz}$.

The spin-spin correlations can be used to identify transitions into phases with broken symmetries; there, the flow becomes unstable at some Λ_T and must be stopped. For long-range ordered states, the momentum $\boldsymbol{k}$ for which the structure factor

$$\chi(\boldsymbol{k}, i\omega) = \frac{1}{N_\text{sites}} \sum_{ij} e^{i\boldsymbol{k}\cdot(\boldsymbol{R}_i - \boldsymbol{R}_j)} \chi_{ij}(i\omega) \tag{10}$$

(i.e. the Fourier transform of χ_{ij}) is most dominant gives an indication of the emergent magnetic order, as exemplified in Fig. 1. A smooth flow down to the infrared $\Lambda \to 0$ is, on the other hand, associated with non-magnetic phases, such as spin liquids, dimerized, or plaquette-ordered states.

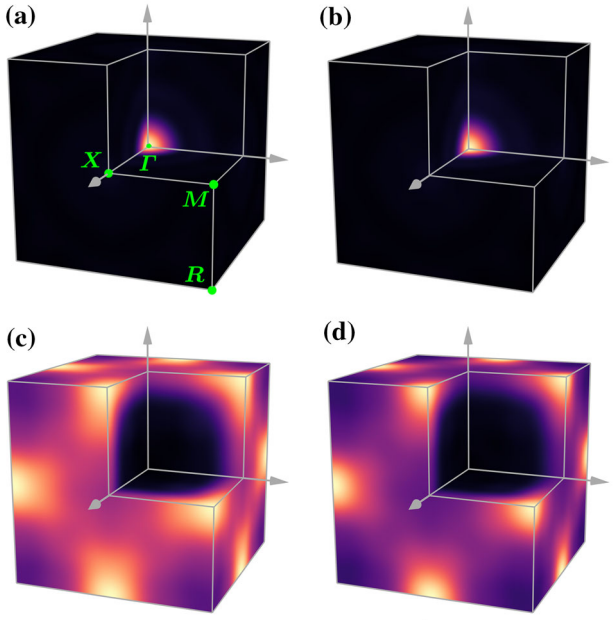


Fig. 1 Momentum-resolved structure factors within the first Brillouin zone of the cubic lattice for (**a**, **b**) the ferromagnetic case at $\Lambda/J = 0.8$ and (**c**, **d**) the paramagnetic case at $\Lambda/J = 0.3$, computed for (**a**, **c**) $\ell = 1$ and (**b**, **d**) $\ell = 3$ using code #2. The ferromagnet shows a sharp peak at the $\boldsymbol{\Gamma}$ point, without visible difference between the two loop orders. The putative paramagnet shows a broadened distribution of spectral weight centered around soft maxima at the $\boldsymbol{M}$ points in $\ell = 1$ calculations, while the structure factor peaks more distinctively for $\ell = 3$, signalling the onset of magnetic order instead

3 Results

To benchmark the two codes, we calculate the spin-spin correlations and pseudofermion vertices of an extended Heisenberg model on the cubic lattice with a maximum correlation length $\xi = 5$ in units of the lattice spacing [1]. The corresponding three-dimensional cluster contains $N = 515$ sites, small enough to efficiently compare the two codes but large enough to produce the (qualitatively) correct physics. The corresponding Hamiltonian with up to third-neighbor interactions (see inset in Fig. 2) reads

$$\mathcal{H} = J_1 \sum_{\langle ij \rangle} S_i^\mu S_j^\mu + J_2 \sum_{\langle\langle ij \rangle\rangle} S_i^\mu S_j^\mu + J_3 \sum_{\langle\langle\langle ij \rangle\rangle\rangle} S_i^\mu S_j^\mu \,, \tag{11}$$

where we fix $J \equiv \sqrt{J_1^2 + J_2^2 + J_3^2}$ as the unit of energy. We focus on two choices of these interaction parameters to highlight differences between fRG flows in different phases:

$$J_1 < 0, \qquad J_2 = 0, \qquad J_3 = 0, \tag{12}$$

$$J_1 > 0, \qquad J_2/J_1 = 0.6, \qquad J_3/J_1 = 0.25, \tag{13}$$

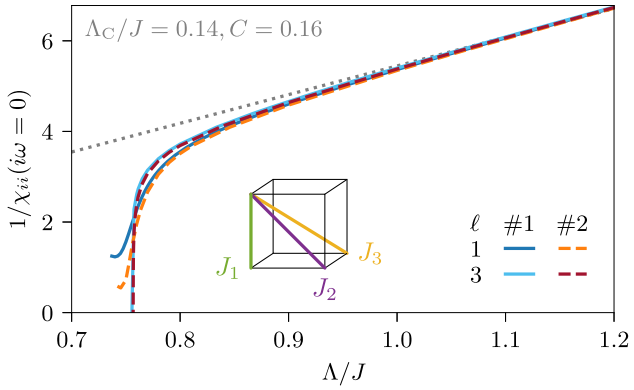


Fig. 2 Inverse spin-spin correlation function for the ferromagnet as a function of Λ. Shown here is a comparison of the $\ell = 1$ and $\ell = 3$ flows obtained from both codes. The dotted line is a Λ^{-1} fit [$\chi_C = CJ/(\Lambda - \Lambda_C)$] to the data at $\Lambda/J \in [1.0, 4.0]$. The transition to a ferromagnetically ordered phase is visible as a sharp downturn away from Curie–Weiss behavior. Inset: Definition of the first, second, and third nearest-neighbor interaction, J_1 (green), J_2 (purple), and J_3 (yellow)

where Eq. (12) yields a nearest-neighbor ferromagnet and the setup of Eq. (13) was previously reported to result in a paramagnetic ground state [21].

Rewriting each spin operator S^μ in the Hamiltonian in terms of pseudofermions leads to an expression proportional to $f^\dagger_{\alpha'} f_\alpha f^\dagger_{\beta'} f_\beta$, with interactions proportional to $\sum_\mu \sigma^\mu_{\alpha'\alpha} \sigma^\mu_{\beta'\beta}$. Exploiting this SU(2) symmetry (the interactions are diagonal and of equal magnitude in every spin direction), the flowing pseudofermion vertex Γ (and each of its two-particle reducible parts γ_c) can be decomposed into a spin component Γ^s, proportional to the latter combination of Pauli matrices, and a density component Γ^d proportional to $\delta_{\alpha'\alpha}\delta_{\beta'\beta}$ [4,45]. Note that the density component, although initially vanishing for any typical spin model, becomes finite away from the UV limit and is essential for tracking the evolution of *all* symmetry-allowed couplings under the RG flow.

3.1 Ferromagnetic phase

With pure nearest-neighbor ferromagnetic interactions, the zero-temperature ground state is intuitively expected to be a ferromagnet. Therefore, in the context of pseudofermion fRG, there should be a transition at some finite $\Lambda_T > 0$ from a paramagnetic regime at large $\Lambda > \Lambda_T$ to the ferromagnetic phase at $\Lambda < \Lambda_T$. Approaching the transition, the spin-spin correlator χ_{ij} is expected to diverge, similar to a finite-temperature phase transition. In this case, a peak will form at the $\boldsymbol{\Gamma}$ point in reciprocal space, as is visible Fig. 1, since the correlations are uniform and positive in a ferromagnet.

Close to the transition, the flow is supposed to visibly deviate from its paramagnetic Curie–Weiss behavior $\chi_{ii} \approx CJ/(\Lambda - \Lambda_C)$ at large $\Lambda \gg \Lambda_T$. For this reason, it is convenient to plot the inverse correlator $1/\chi_{ii}$ as a function of Λ to locate the transition, as

shown in Fig. 2. Here, the $1/\Lambda$ behavior appears as a straight line with slope $1/C$ displaced horizontally by Λ_C/J and the transition to the ferromagnetic phase is visible as a sharp turn down to a smaller inverse correlation function at $\Lambda/J \approx 0.76$. The structure factor at Λ close to Λ_T, shown in Figs. 1 and 3, has a single peak at the $\boldsymbol{\Gamma}$ point, signifying an instability towards ferromagnetic order. This, as well as the Curie–Weiss fit parameters, are consistent across both considered loop orders $\ell = 1, 3$ and both codes, while Λ_T differs slightly.

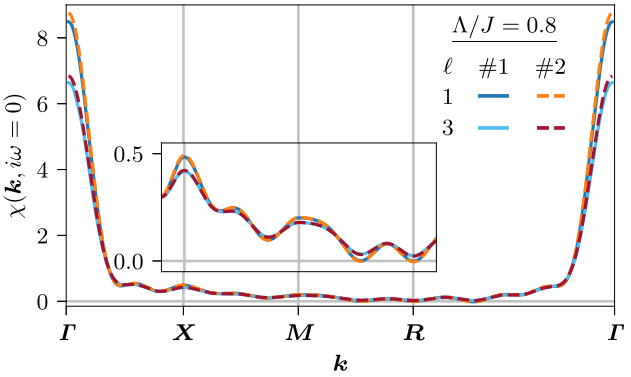


Fig. 3 Structure factor for the ferromagnet along a high-symmetry path of the cubic lattice Brillouin zone. The results are in excellent agreement between both codes, both for $\ell = 1$ and $\ell = 3$, showing dominant ferromagnetic correlations indicated by a sharp peak around the $\boldsymbol{\Gamma}$ point. Inset: Zoom into the path segment connecting the $\boldsymbol{X}$, $\boldsymbol{M}$, and $\boldsymbol{R}$ point

Since both implementations obtain the spin-spin correlations by post-processing the vertices, any discrepancy therein originates from differences in the vertices. Therefore, a more detailed examination of the $1/\chi_{ii}$-deviations between the codes for $\ell = 1$ will follow once the flow of the vertex components has been discussed. Moreover, even if the flows for the χ_{ij} agree perfectly (as, e.g., in the regime $\Lambda > \Lambda_T$), discrepancies in the vertices cannot be fully excluded, as post-processing spin-spin correlations from pseudofermion vertex data amounts to integrating a combination of several propagators and the vertex over two frequencies [1]. Hence, this additional step might hide potential differences in the vertex data.

To investigate this further, we focus on the t-reducible vertex γ_t plotted in Fig. 4 at various values of Λ: Its spin component γ_t^s (second and third column) is responsible for the transition and becomes sharply peaked at small bosonic frequencies $\omega \approx 0$. Its density component γ_t^d (last column) with its extended structures and peaks at non-zero fermionic frequencies ν is particularly difficult to resolve and thus most likely to contain numerical artifacts. Comparing γ_t, as well as the the self-energy Σ between the codes, we find quan-

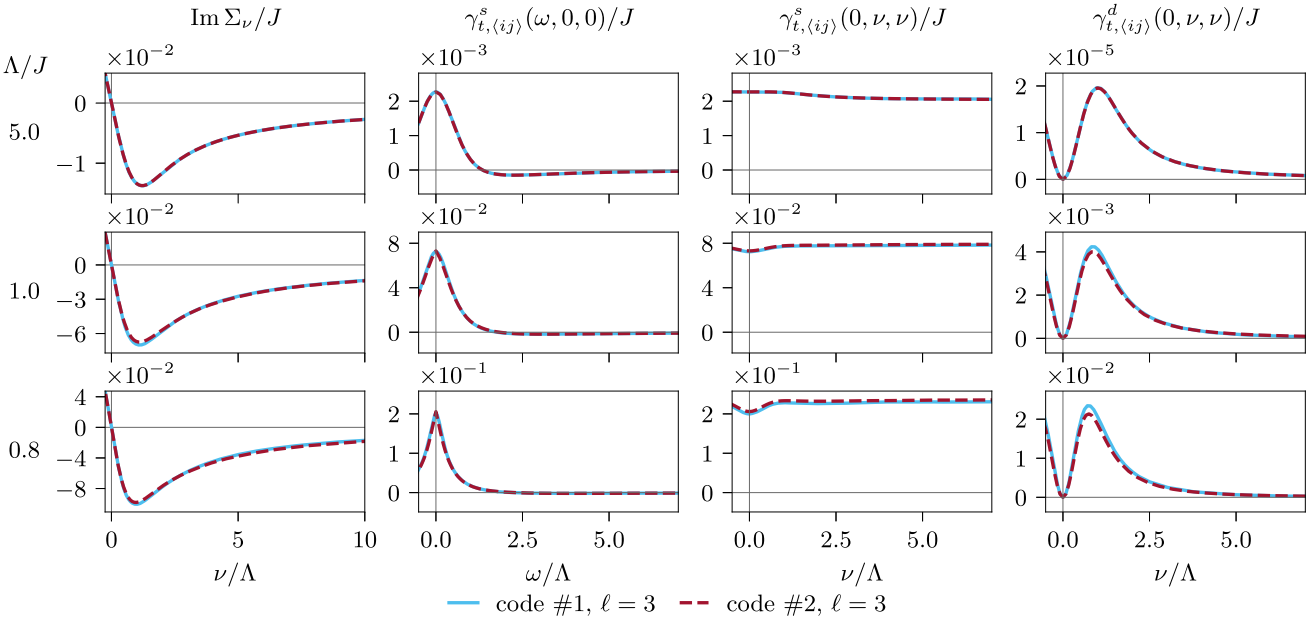


Fig. 4 Frequency structure of self-energy and t-reducible vertex for the ferromagnet at different values of Λ/J for $\ell = 3$ flows. The self-energy is purely imaginary and antisymmetric in frequency space, while all vertex components are real and symmetric along the directions plotted here. We show two cuts through the three-dimensional structure of $\gamma^{\Lambda,\mu}_{t,\langle ij\rangle}(\omega,\nu,\nu')$: A cut along the bosonic frequency axis ω, with both fermionic frequencies set to $\nu = \nu' = 0$, and a cut with equal fermionic frequencies $\nu = \nu'$, where the bosonic frequency was set to $\omega = 0$. The first cut is not shown for γ_t^d as $\gamma^d_{t,\langle ij\rangle}(\omega,0,0) = 0$ due to symmetry [1,2]. The most prominent structure in the t-reducible vertex is a peak around zero bosonic frequency $\omega = 0$ that grows in magnitude and becomes sharper as Λ is decreased. This indicates ferromagnetic correlations that grow stronger as the ordering phase transition is approached. In all components, there is quantitative agreement between the two codes

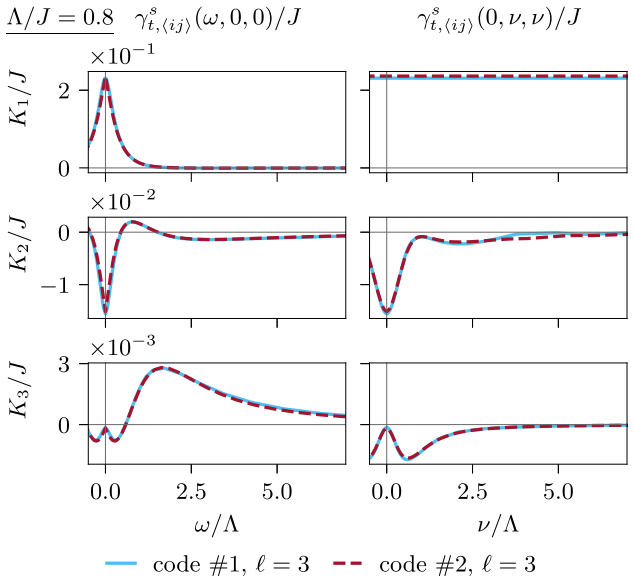


Fig. 5 Decomposition of the $\gamma^s_{t,\langle ij\rangle}(\omega, \nu, \nu')$ vertex for the ferromagnet into asymptotic classes $K_{1,t}, K_{2,t}, K_{3,t}$ (first, second, third row) for the $\ell = 3$ flows at $\Lambda/J = 0.8$. Frequency axes shown here are the same as in Fig. 4. As the flow is close to the ordering phase transition at this value of Λ, strong ferromagnetic correlations are present as a peak around $\omega = 0$ in $K_{1,t}$. The other classes are at least one order of magnitude smaller. In all classes, both codes show quantitative agreement

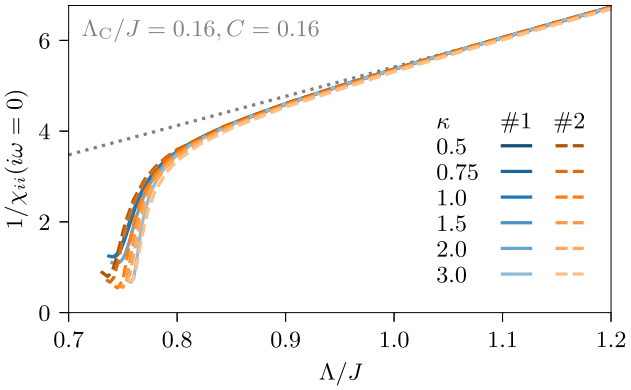


Fig. 6 Flows with rescaled frequency meshes. Comparison of the flow of inverse static on-site spin correlations $1/\chi_{ii}(i\omega = 0)$ obtained using frequency meshes with different scaling factors κ. The dotted line is a Λ^{-1} fit to the data at $\Lambda/J \in [1.0, 4.0]$. For all values of κ, a transition to a ferromagnet is visible as a sharp turn down. The predicted transition point as well as the slope of χ in the region $\Lambda/J < 0.8$ differs, while the behavior at large $\Lambda > J$ remains identical

titative agreement also on this very detailed level of inspection.

As outlined in Sect. 2, both codes use a decomposition of the reducible vertices $\gamma_s, \gamma_t, \gamma_u$ into four asymptotic classes each. The decomposition into asymptotic classes K_n is shown for γ^s_t at $\Lambda/J = 0.8$ in Fig. 5, where we omit $K^s_{2',t}$, as it is equal to $K^s_{2,t}$ by crossing symmetry [1,2]. Note that, while these vertices can directly be extracted from code #1, an additional transformation is applied to the Q_n decomposition of code #2 [see Eq. (8)]. The peak in γ^s_t at small bosonic frequencies in Fig. 4 is found to stem from the K_1 contribution, which is an order of magnitude larger than the other classes. In K_2 and K_3, extended structures with multiple maxima and minima exist. It is thus crucial to use a frequency mesh with enough mesh points in an extended region around the origin to control numerical interpolation errors (see Sect. 4).

Though the codes implement the vertex decomposition differently (see Sect. 2) and use different approaches to build appropriate frequency meshes (see [1,2] for a detailed description), all components of the vertex are consistent with each other. This demonstrates that it is possible to gain control over said interpolation errors by a careful adaptive implementation that places enough mesh points where they are needed.

Since the numerical error incurred by interpolation of the continuous frequency structure from a discrete mesh is particularly relevant whenever sharp structures are present in the vertex, different choices of frequency meshes have strong effects close to phase transitions, where some couplings are expected to diverge. For instance, in the ferromagnetic setup discussed above, the transition was induced by a peak in the spin component of the t-reducible vertex that grows quickly and starts to diverge, as can be seen in the second column of Fig. 4. As the transition is approached, this peak progressively becomes sharper and thus more difficult to resolve using discrete meshes. Thus, minor differences in mesh spacing can induce differences in the flow at the transition, though the qualitative, physical results remain unchanged.

To investigate the effects of changes in the mesh spacing explicitly, we compared results obtained from both codes with artificially modified meshes. Both implementations make use of adaptive frequency grids where, during the flow, the mesh spacing is adjusted according to the frequency structure of the vertex. The simplest way to manipulate the meshes is to rescale them by an artificial scaling factor κ. In Fig. 6, we show the effect of such a rescaling on the $\ell = 1$ flow from Fig. 2. Above $\Lambda/J \approx 0.8$, all frequency structures in the vertex are fairly broad and easy to resolve. Consequently, rescaling the frequency grid has little effect and values $\kappa = 0.5 \ldots 3.0$ result in the same flow and also the same Curie–Weiss fit parameters. Below that point, the flows differ more and more as structures become sharper and ultimately predict slightly different transition points Λ_T/J. Nevertheless, all flows predict a transition to the same ferromagnetic phase, which can be identified by a peak in the structure factor at the $\boldsymbol{\Gamma}$ point.

3.2 Paramagnetic phase

For the second set of parameters, Eq. (13), all interactions up to the third neighbor are antiferromagnetic.

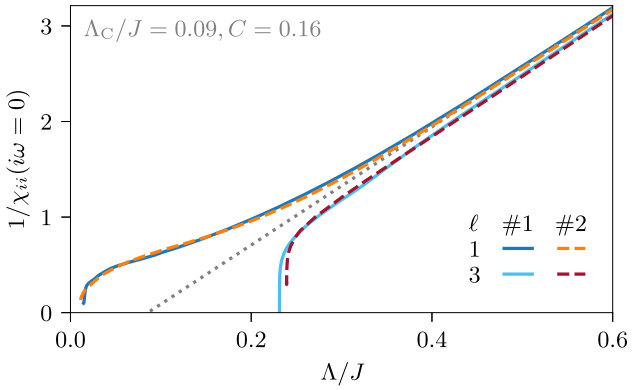


Fig. 7 Inverse spin-spin correlation function for the putative paramagnet as a function of Λ. Shown here is a comparison of the $\ell = 1$ and $\ell = 3$ flow obtained from both codes. The dotted line is a fit of a Λ^{-1} power law to the data at $\Lambda/J \in [1.0, 4.0]$. For $\Lambda/J \geq 0.5$, the Λ^{-1} behavior is followed almost perfectly. At smaller Λ/J, the $\ell = 1$ and $\ell = 3$ flows disagree: The $\ell = 1$ curve smoothly approaches $\Lambda = 0$ (staying above the power law), indicating antiferromagnetic correlations. By contrast, the $\ell = 3$ curve displays a downward cusp, similar to Fig. 2, and thus predicts an ordered state

Consistent with prior work using one-loop fRG [21], both codes find a paramagnetic ground state for $\ell = 1$, indicated by a smooth and regular flow down to $\Lambda = 0$ in Fig. 7.

Remarkably, the $\ell = 3$ data predicts a qualitatively different phase: There is a divergence in the spin correlations at $\Lambda_T/J \approx 0.24$, indicating an ordering transition at a scale roughly three times lower than for the ferromagnetic ordering instability discussed in the previous section. Such a reduced ordering scale is not unexpected for an exchange-frustrated spin system when compared to an unfrustrated one, but sometimes hard to establish.

Probing the structure factor in the vicinity of the divergence reveals a strong enhancement of magnetic correlations compared to the $\ell = 1$ flow, as indicated by sharpened Bragg peaks around the $\boldsymbol{M} = (0, \pi, \pi)$ points in Figs. 1 and 9. These correspond to antiferromagnetic correlations between planes orthogonal to the vector connecting the second nearest-neighbors along diagonals of the faces in the cubic unit cell (shown in purple in Fig. 2). Our result is consistent with earlier observations of long-range $(0, \pi, \pi)$ order neighboring the paramagnetic phase [21]. Yet, the mfRG flows obtained from both codes suggest a rather strong modification of the respective phase boundaries as the coupling parameters investigated here were previously predicted to be deep in the non-magnetic regime.

In the vertex (see Fig. 8) and self-energy, there is again very good quantitative agreement between both codes. At $\Lambda/J = 0.05$, small quantitative differences between code #1 and #2 appear in the density component γ_t^d of the t-reducible vertex, consistent with the earlier remark that it is the most difficult component to resolve well.

The $\ell = 1$ and $\ell = 3$ flows are very similar down to $\Lambda/J \geq 1$. Contributions of $\ell > 1$ terms become significant at $\Lambda/J \approx 1$ and eventually lead to an ordering instability induced by a peak in the γ_t^s component that diverges at $\Lambda/J \approx 0.24$. In contrast to the ferromagnetic case, this peak is negative, indicating anticorrelation. Along the fermionic ν frequency axis, the vertex shows an extended structure with multiple peaks of similar magnitude to the one on the bosonic axis. Since the K_1 class has no fermionic frequency, this means that, remarkably, other classes reach an order of magnitude comparable to K_1, as shown explicitly in Fig. 10. Consequently, vertex structures along fermionic frequency axes, in contrast to the ferromagnetic transition, become sizeable. It is therefore crucial to resolve the full three-dimensional frequency structure in K_3. Though numerically expensive, a large number of mesh points is necessary to ensure sufficient accuracy, as inadequate resolution of features along the fermionic frequency axes can strongly affect the fRG flow. This is even more important for multiloop flows, where interpolation errors might accumulate during the iteration over loop orders.

4 Technical aspects

To conclude our benchmark calculations, we discuss some of the particularly relevant technical aspects (see Table 1) which are needed to obtain confidence that we have sufficient degree of control over numerical errors. In doing so, we will also connect to the existing literature and scrutinize some of the algorithmic approaches which are routinely employed in the pffRG community.

4.1 Frequency grids

Both the self-energy and two-particle vertices are functions of Matsubara frequencies, which are continuous in the zero-temperature limit. A numerical implementation has to sample these functions on a finite grid and interpolate their values in between the sampling points. In many previous works (see e.g. Refs. [4,19,46]), the *same* frequency grid was chosen for the self-energy and all reducible vertices, usually featuring logarithmically increasing distances between adjacent grid points starting from some small but finite frequency. The intention behind such a choice of frequencies was to resolve the structure around zero frequency with high accuracy while coarse-graining high-frequency tails. Moreover, each vertex component was parametrized in terms of the three bosonic transfer frequencies, instead of the channel-specific mixed bosonic-fermionic frequency treatment utilized by codes #1 and #2.

Although most of the structure of the two-particle vertex is indeed centered around zero frequency, its precise extent strongly depends on the cutoff scale Λ (see, e.g., Figs. 4 and 8) and a *static* frequency grid will therefore fail to faithfully resolve the evolution of frequency structures under the fRG flow. Furthermore, multipeak

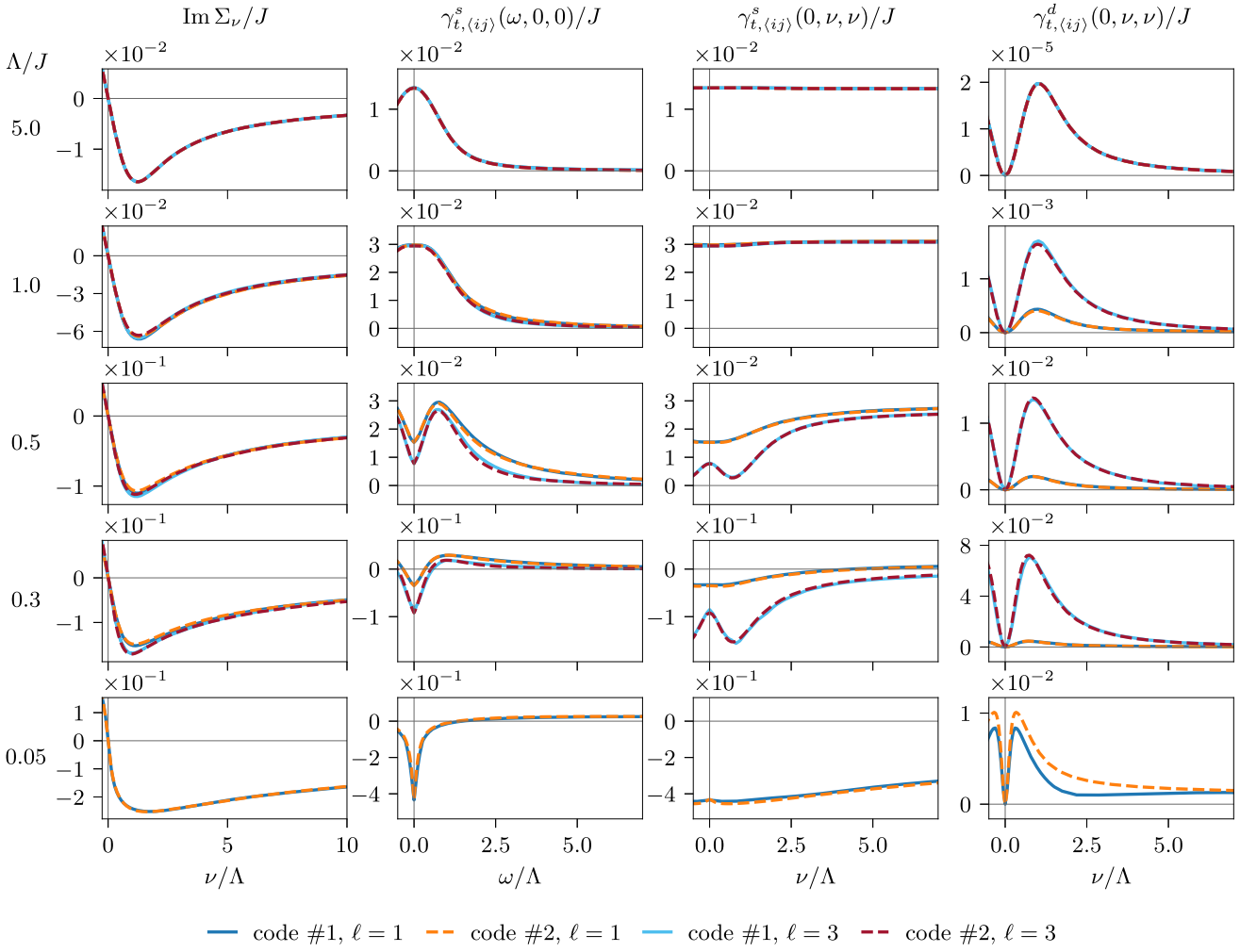


Fig. 8 Frequency structure of self-energy and t-reducible vertex for the putative paramagnet at different values of Λ/J for $\ell = 1$ and 3 flows. As the $\ell = 3$ flow diverges at $\Lambda/J \approx 0.24$, only $\ell = 1$ is shown at $\Lambda/J = 0.05$. The same cuts through the three-dimensional frequency structure of the vertices are shown as in Fig. 4. Again, a peak in the $\gamma^s_{t,\langle ij\rangle}$ component (second column) indicates strong correlations that become stronger as Λ is further decreased. In contrast to the ferromagnetic case, this peak is negative, indicative of antiferromagnetic correlations, and there is a sizeable contribution of γ^s_t for nonzero fermionic frequencies ν, ν' (third column), particularly for $\ell = 3$

structures that are present in several vertex components will in general not be captured by logarithmic sampling.

To address both shortcomings, codes #1 and #2 introduce hybrid frequency meshes using linear spacing around zero frequency augmented by an algebraic (code #1) or logarithmic (code #2) part to capture the high-frequency behavior in the asymptotic classes K_n or Q_n. The parameters of these meshes are then independently rescaled for different vertex components making use of sophisticated scanning routines (see [1,2] for further details).

4.2 Evaluation of bubble integrals

Having fixed the frequency discretization, the evaluation of frequency integrals in loop and bubble functions necessitates the use of a quadrature rule. In earlier implementations, a trapezoidal quadrature was used, with integration points coinciding with the frequency mesh of the vertex. As discussed above, this procedure yields good resolution around the origin of the integration variable. For 1ℓ calculations, the bubble function consists of a single-scale and a full propagator, the former being more strongly peaked than the latter. As the integration variable was usually shifted such that the origin coincided with the more important pole of the single-scale propagator, at least the dominant contribution was accounted for in previous implementations.

In higher loops, however, both propagators enter the bubble on equal footing, necessitating *adaptive* routines to deal with the enriched frequency structure. This is illustrated in Fig. 11, where we compare the results of integrating the bare susceptibility

$$\chi_0^\Lambda(\omega) = \frac{1}{4\pi}\int d\nu\, G_0^\Lambda(\nu+\tfrac{\omega}{2})\, G_0^\Lambda(\nu-\tfrac{\omega}{2})\,,$$

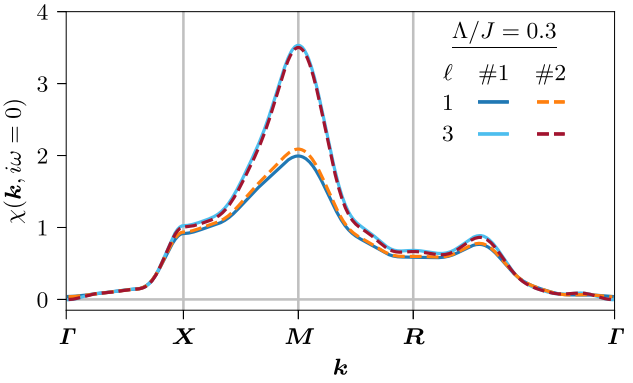


Fig. 9 Structure factor for the paramagnetic setup along a high-symmetry path of the cubic lattice Brillouin zone. The results are in good agreement between both codes, both for $\ell = 1$ and $\ell = 3$, showing that correlations are strongest around the $\boldsymbol{M}$ point. Here, the peak sharpens with increasing loop order, and the $\ell = 3$ flow predicts enhanced long-range correlations

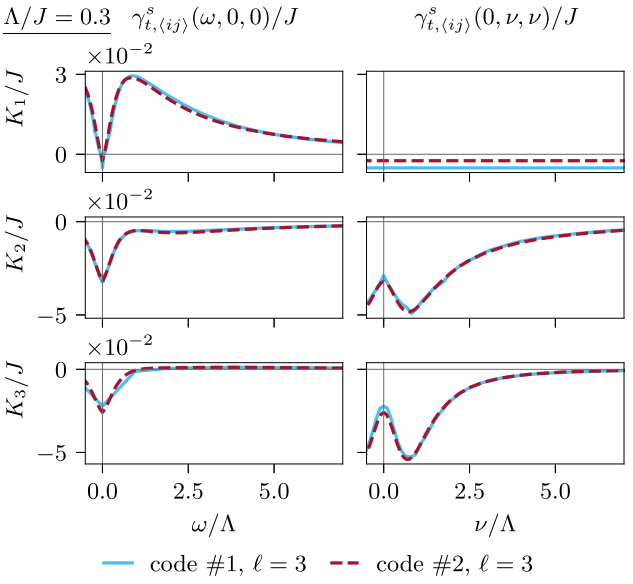


Fig. 10 Decomposition of the $\gamma^s_{t,\langle ij\rangle}(\omega, \nu, \nu')$ vertex in the paramagnetic setup as in Fig. 5, for the $\ell = 3$ flows at $\Lambda/J = 0.3$. Here, all asymptotic classes are of the same order of magnitude, and structures with multiple peaks are present along the fermionic frequency cut (second column)

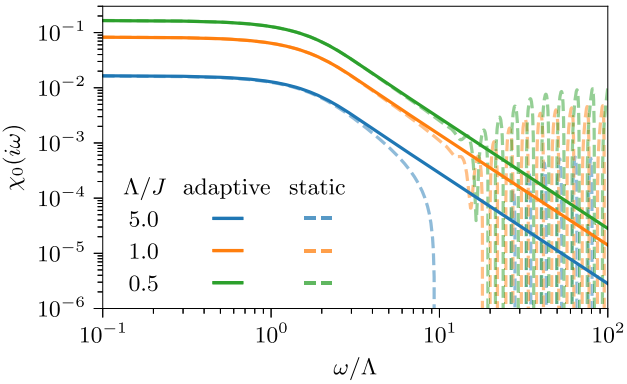


Fig. 11 Evaluation of bubble integrals. Comparison of the bare susceptibility $\chi_0^\Lambda(\omega) = \frac{1}{4\pi}\int d\nu\, G_0^\Lambda(\nu + \frac{\omega}{2})\, G_0^\Lambda(\nu - \frac{\omega}{2})$ obtained numerically via adaptive and static quadrature. The adaptive method utilizes the Simpson rule, while the static method applies a trapezoidal rule to a fixed logarithmic frequency discretization (see main text for more details). For frequencies larger than the scale set by the cutoff Λ, the non-adaptive integration becomes unstable and is plagued by rapid oscillations. By contrast, the adaptive routine yields stable results even beyond the small frequency regime and is therefore crucial to obtain accurate results for the vertex functions and their asymptotic behavior

i.e., the simplest bubble-like integral encountered during the fRG flow. Using trapezoidal quadrature over a fixed set of 60 logarithmically distributed integration points between $\nu_{\min} = 10^{-3}J$ and $\nu_{\max} = 250J$, we find strong deviations for frequencies $\omega/\Lambda \gtrsim 1 \sim 10$ compared to the results produced with the adaptive routine of code #2 (see Ref. [2] for further details). Moreover, at small cutoffs $\Lambda/J \lesssim 1$, the non-adaptive result is plagued by rapid oscillations, rendering it numerically unstable and thus inapplicable. Analytically, an asymptotic falloff with a power law ω^{-2} is expected, and this is reproduced perfectly by the adaptive integrator.

We emphasize that the test case considered here merely constitutes the simplest version of a bubble-like integral computed within the pffRG flow. In general, the propagators in bubble functions are dressed with self-energy insertions and additionally contracted with two-frequency dependent vertices. One should, therefore, expect even larger numerical errors for full fRG calculations that utilize non-adaptive quadrature.

Table 1 Technical summary of the algorithmic choices in code #1 and #2

	Code #1	Code #2
Vertex decomposition	K_1, K_2, K_3	Q_1, Q_2, Q_3
Frequency mesh	Adaptive linear and algebraic	Adaptive linear and logarithmic
Integration rule	Adaptive 21-point Gauss–Kronrod rule	Adaptive Simpson rule + Richardson extrapolation
ODE solver	5th order Cash–Carp	3rd order Bogacki–Shampine

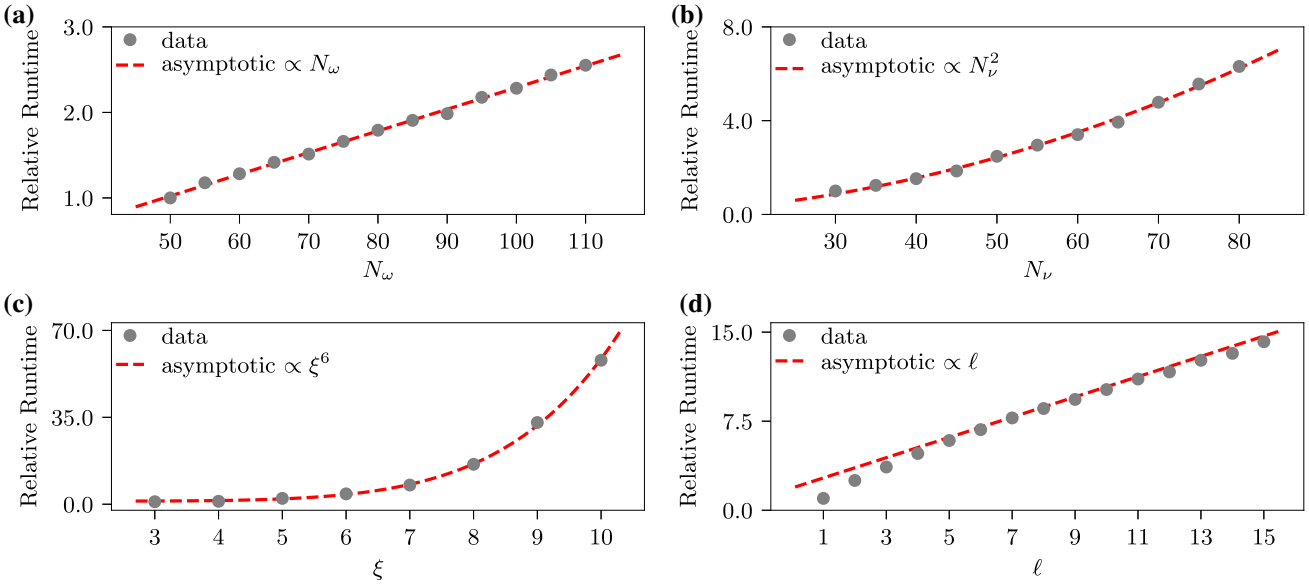


Fig. 12 Scaling of relative runtime with numerical parameters. Median computational runtime of 60 samples of a single calculation of the right-hand side of the flow equation for $\Lambda/J = 1$ relative to the runtime of the fastest computation in each series. Calculations start from a parquet solution to make the code integrate over non-trivial frequency structures. The numerical parameters for all plots are fixed to $N_\omega = 50$, $N_\nu = 30$, $\xi = 4$ and $\ell = 1$, if not varied. The asymptotic behavior expected analytically is achieved in all cases (dashed red lines)

4.3 Flow integration

The integration of the RG flow can, in principle, be performed using any standard solver for ordinary differential equations. While earlier works used an Euler scheme with decreasing step-sizes (see, e.g., Ref. [46]), we employ higher order solvers in the Runge–Kutta family with adaptive step-size control to achieve maximum accuracy while being numerically efficient to operate. It is of particular importance to implement an error-controlling method near ordering instabilities such as the ferromagnetic setup in Sect. 3, as otherwise numerical errors may become unacceptably large even at scales $\Lambda \approx J$.

4.4 Initial condition

The final ingredient to set up the pffRG flow is an appropriate initial condition. In the UV limit $\Lambda \to \infty$, the pseudofermion vertex is given by the bare spin coupling, which, in numerical calculations, is naturally implemented using J as the initial condition at a large but finite value of Λ. The mfRG flow will, by construction, reproduce a solution to the parquet equations [33–35], given an initial condition consistent with them. Therefore, we solve the regularized parquet equations iteratively for an initial scale $\Lambda/J = 5$ and use the resulting self-energy and reducible vertices as a dynamic, i.e., frequency-dependent starting point for the fRG flow [1].

4.5 Scaling analysis

Most of the runtime needed to evaluate the right-hand side of the flow equations is spent calculating the derivative of the high-dimensional two-particle vertex as given in Eq. (6). In comparison, the computation time spent for the self-energy derivative of Eq. (5) is negligible. Consequently, the (asymptotic) computational complexity is given by

$$\mathcal{O}\left(N_\xi^2 \times N_\omega N_\nu^2 \times \ell\right) ,$$

where N_ξ is the number of (symmetry reduced [1,2]) lattice sites, N_ω (N_ν) the number of bosonic (fermionic) frequencies, and ℓ denotes the number of loops. The total number of sites, in turn, is expected to follow a $\mathcal{O}(\xi^d)$ dependence, where ξ is the maximal correlation length considered and d is the spatial dimensionality of the underlying lattice, with $d = 3$ for the simple cubic lattice at hand.

To demonstrate that we indeed reach this asymptotic algorithmic scaling also in numerical implementations we show, in Fig. 12, the median runtime data for 60 evaluations of the right-hand side of the fRG equations obtained using code #2. For the number of bosonic and fermionic frequencies, the expected linear and quadratic behavior, respectively, is achieved over the whole parameter range. Note that, due to the adaptive integration and parallelization used, slight deviations from the theoretical scaling are to be expected. Similarly, the scaling in the maximal correlation length ξ is achieved for the whole parameter range. In the number of loops, the linear scaling sets in at $\ell = 5$, while for smaller ℓ a steeper slope is found. We attribute this behavior to the contributions of higher loops becoming successively smaller, leading to faster converging adaptive loop integrals for given absolute and relative tolerances. That way, the initial overhead of computing two

Table 2 Number of (symmetry reduced) vertex flow equations for Heisenberg models on the cubic lattice as a function of the maximum correlation length ξ. The number of positive frequencies is fixed to 60 (50) for the bosonic (fermionic) Matsubara axis

Max. correlation length ξ	No. flow equations
3	9 183 600
5	24 795 720
7	53 264 880
9	101 019 600
11	167 141 520
13	258 059 160

(three) loop corrections, which require twice (thrice) the number of integrals to be evaluated compared to $\ell = 1$, diminishes with increasing loop number and the analytically expected scaling, linear in ℓ, is recovered.

As a final remark, we mention that the number of vertex flow equations, another measure of algorithmic complexity, grows rapidly as one increases the maximal correlation length considered for a given lattice model. This is summarized in Table 2.

5 Conclusions

We benchmarked two state-of-the-art codes for solving pseudofermion functional renormalization group equations. Our analysis considered both physical observables, i.e. spin-spin correlation functions and structure factors, as well as fermionic vertex functions (self-energy and two-particle vertex) for ferro- and antiferromagnetic models on the simple cubic lattice.

For the nearest-neighbor ferromagnet, both codes were in quantitative agreement at least until $\Lambda/J \gtrsim 0.76$, where they consistently predicted a breakdown of the RG flow, indicated by a sharp peak (for $\ell = 1$) or a divergence (for $\ell = 3$) in the spin-spin correlations. The energy scale Λ_T associated with this numerical instability slightly differed, which necessitated an in-depth comparison of the influence of the numerical frequency grid on the obtained results. We found that both fRG solvers, due to the emergence of a singular peak in the t reducible vertex functions, become sensitive to the precise mesh spacing and thus predict marginally different critical scales, although the physical conclusion drawn from the RG flow, i.e. the onset of long-range ferromagnetic order, remains the same.

For the antiferromagnetic setup, the $\ell = 1$ results obtained by both codes were in agreement with one another and previous studies [21], predicting a paramagnetic state, signified by a regular RG flow down to the infrared. For $\ell = 3$, similar numerical agreement between the two codes was found. However, the physical results changed qualitatively: the flow of the spin-spin correlator diverged around $\Lambda/J \approx 0.24$, accompanied by sharp Bragg peaks at the $\boldsymbol{M}$ points indicating the formation of antiferromagnetic order at low temperatures. This reinstantiates the importance of including higher loop corrections in pffRG to avoid overestimating the extent of paramagnetic phases and to obtain more accurate predictions of ground states in frustrated quantum magnets.

We also elaborated on the importance of employing adaptive numerical algorithms to obtain robust results at all stages of the flow. More explicitly, there are extended structures with multiple peaks in the three-dimensional frequency dependence of several vertex components. As these structures are sizable, it is crucial to resolve them in an accurate manner. We found fixed logarithmic frequencies to be insufficient for structures not centered at zero frequency, and rely instead on adaptive frequency meshes that have been specifically optimized for pffRG vertices. Furthermore, we demonstrated that the commonly employed quadrature of a trapezoidal rule over a static, logarithmic mesh fails to produce the analytically expected behavior of bare bubble integrations at large frequencies. It is thus unsuitable for providing the essential Matsubara integrals for error-controlled fRG flows. By contrast, the implementations presented and benchmarked here solve these problems using highly accurate, yet efficient adaptive routines (see Table 1). We thus believe that, moving forward, they will be widely used for unbiased calculations of (multiloop) ground-state phase diagrams of frustrated magnets from pffRG.

Acknowledgements We thank L. Gresista, Y. Iqbal, M. Punk, and J. Reuther for useful and stimulating discussions and J. Thoenniss for his pioneering contribution to setting up the Munich multiloop pffRG code. The Cologne group gratefully acknowledges partial support from the Deutsche Forschungsgemeinschaft (DFG)—Projektnummer 277146847—SFB 1238 (project C02), the Munich group from DFG under Germany's Excellence Strategy EXC-2111 (Project No. 390814868), the Würzburg group from DFG through Project-ID 258499086-SFB 1170 and the Würzburg-Dresden Cluster of Excellence on Complexity and Topology in Quantum Matter—ct.qmat Project-ID 390858490-EXC 2147, and F.B.K. from the Alexander von Humboldt Foundation through a Feodor Lynen Fellowship. The numerical simulations were performed on the JURECA Booster and JUWELS cluster at the Forschungszentrum Juelich, the SuperMUC cluster and Linux clusters at the Leibniz Supercomputing Centre, as well as the CHEOPS cluster at RRZK Cologne. This research is also part of the Munich Quantum Valley, which is supported by the Bavarian State Government with funds from the Hightech Agenda Bayern Plus.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data Availability Statement This manuscript has no associated data or the data will not be deposited. [Author's comment: The numerical data generated and analysed in this paper is available from the corresponding author on reasonable request. Furthermore, Code #2 is available as an open-source package [39].]

References

1. J. Thoenniss, M.K. Ritter, F.B. Kugler, J. von Delft, M. Punk. arXiv:2011.01268
2. D. Kiese, T. Müller, Y. Iqbal, R. Thomale, S. Trebst. arXiv:2011.01269
3. C. Lacroix, P. Mendels, F. Mila, *Introduction to Frustrated Magnetism*. Springer Series in Solid-State Sciences (Springer Berlin Heidelberg, 2011). https://doi.org/10.1007/978-3-642-10589-0
4. J. Reuther, P. Wölfle, Phys. Rev. B **81**, 144410 (2010). https://doi.org/10.1103/PhysRevB.81.144410
5. J. Reuther, R. Thomale, Phys. Rev. B **83**, 024402 (2011). https://doi.org/10.1103/PhysRevB.83.024402
6. J. Reuther, P. Wölfle, R. Darradi, W. Brenig, M. Arlego, J. Richter, Phys. Rev. B **83**, 064416 (2011). https://doi.org/10.1103/PhysRevB.83.064416
7. J. Reuther, D.A. Abanin, R. Thomale, Phys. Rev. B **84**, 014417 (2011). https://doi.org/10.1103/PhysRevB.84.014417
8. J. Reuther, R. Thomale, S. Trebst, Phys. Rev. B **84**, 100406 (2011). https://doi.org/10.1103/PhysRevB.84.100406
9. J. Reuther, R. Thomale, S. Rachel, Phys. Rev. B **86**, 155127 (2012). https://doi.org/10.1103/PhysRevB.86.155127
10. J. Reuther, R. Thomale, Phys. Rev. B **89**, 024412 (2014). https://doi.org/10.1103/PhysRevB.89.024412
11. R. Suttner, C. Platt, J. Reuther, R. Thomale, Phys. Rev. B **89**, 020408 (2014). https://doi.org/10.1103/PhysRevB.89.020408
12. J. Reuther, R. Thomale, S. Rachel, Phys. Rev. B **90**, 100405 (2014). https://doi.org/10.1103/PhysRevB.90.100405
13. Y. Iqbal, H.O. Jeschke, J. Reuther, R. Valentí, I.I. Mazin, M. Greiter, R. Thomale, Phys. Rev. B **92**, 220404 (2015). https://doi.org/10.1103/PhysRevB.92.220404
14. Y. Iqbal, W.J. Hu, R. Thomale, D. Poilblanc, F. Becca, Phys. Rev. B **93**, 144411 (2016). https://doi.org/10.1103/PhysRevB.93.144411
15. Y. Iqbal, P. Ghosh, R. Narayanan, B. Kumar, J. Reuther, R. Thomale, Phys. Rev. B **94**, 224403 (2016). https://doi.org/10.1103/PhysRevB.94.224403
16. F.L. Buessen, S. Trebst, Phys. Rev. B **94**, 235138 (2016). https://doi.org/10.1103/PhysRevB.94.235138
17. A. Keleş, E. Zhao, Phys. Rev. Lett. **120**, 187202 (2018). https://doi.org/10.1103/PhysRevLett.120.187202
18. A. Keleş, E. Zhao, Phys. Rev. B **97**, 245105 (2018). https://doi.org/10.1103/PhysRevB.97.245105
19. D. Kiese, F.L. Buessen, C. Hickey, S. Trebst, M.M. Scherer, Phys. Rev. Res. **2**, 013370 (2020). https://doi.org/10.1103/PhysRevResearch.2.013370
20. N. Astrakhantsev, F. Ferrari, N. Niggemann, T. Müller, A. Chauhan, A. Kshetrimayum, P. Ghosh, N. Regnault, R. Thomale, J. Reuther, T. Neupert, Y. Iqbal, Phys. Rev. B **104**, L220408 (2021). https://doi.org/10.1103/PhysRevB.104.L220408
21. Y. Iqbal, R. Thomale, F. Parisen Toldin, S. Rachel, J. Reuther, Phys. Rev. B **94**, 140408 (2016). https://doi.org/10.1103/PhysRevB.94.140408
22. Y. Iqbal, T. Müller, K. Riedl, J. Reuther, S. Rachel, R. Valentí, M.J.P. Gingras, R. Thomale, H.O. Jeschke, Phys. Rev. Mater. **1**, 071201 (2017). https://doi.org/10.1103/PhysRevMaterials.1.071201
23. F.L. Buessen, M. Hering, J. Reuther, S. Trebst, Phys. Rev. Lett. **120**, 057201 (2018). https://doi.org/10.1103/PhysRevLett.120.057201
24. Y. Iqbal, T. Müller, H.O. Jeschke, R. Thomale, J. Reuther, Phys. Rev. B **98**, 064427 (2018). https://doi.org/10.1103/PhysRevB.98.064427
25. Y. Iqbal, T. Müller, P. Ghosh, M.J.P. Gingras, H.O. Jeschke, S. Rachel, J. Reuther, R. Thomale, Phys. Rev. X **9**, 011005 (2019). https://doi.org/10.1103/PhysRevX.9.011005
26. P. Ghosh, T. Müller, F.P. Toldin, J. Richter, R. Narayanan, R. Thomale, J. Reuther, Y. Iqbal, Phys. Rev. B **100**, 014420 (2019). https://doi.org/10.1103/PhysRevB.100.014420
27. A. Revelli, C.C. Loo, D. Kiese, P. Becker, T. Fröhlich, T. Lorenz, M. Moretti Sala, G. Monaco, F.L. Buessen, J. Attig, M. Hermanns, S.V. Streltsov, D.I. Khomskii, J. van den Brink, M. Braden, P.H.M. van Loosdrecht, S. Trebst, A. Paramekanti, M. Grüninger, Phys. Rev. B **100**, 085139 (2019). https://doi.org/10.1103/PhysRevB.100.085139
28. P. Ghosh, Y. Iqbal, T. Müller, R.T. Ponnaganti, R. Thomale, R. Narayanan, J. Reuther, M.J.P. Gingras, H.O. Jeschke, NPJ Quant. Mater. **4**(1), 63 (2019). https://doi.org/10.1038/s41535-019-0202-z
29. S. Chillal, Y. Iqbal, H.O. Jeschke, J.A. Rodriguez-Rivera, R. Bewley, P. Manuel, D. Khalyavin, P. Steffens, R. Thomale, A.T.M.N. Islam, J. Reuther, B. Lake, Nat. Commun. **11**(1), 2348 (2020). https://doi.org/10.1038/s41467-020-15594-1
30. M.L. Baez, J. Reuther, Phys. Rev. B **96**, 045144 (2017). https://doi.org/10.1103/PhysRevB.96.045144
31. F.L. Buessen, D. Roscher, S. Diehl, S. Trebst, Phys. Rev. B **97**, 064415 (2018). https://doi.org/10.1103/PhysRevB.97.064415
32. A.A. Katanin, Phys. Rev. B **70**, 115109 (2004). https://doi.org/10.1103/PhysRevB.70.115109
33. F.B. Kugler, J. von Delft, Phys. Rev. B **97**, 035162 (2018). https://doi.org/10.1103/PhysRevB.97.035162
34. F.B. Kugler, J. von Delft, Phys. Rev. Lett. **120**, 057403 (2018). https://doi.org/10.1103/PhysRevLett.120.057403
35. F.B. Kugler, J. von Delft, New J. Phys. **20**(12), 123029 (2018). https://doi.org/10.1088/1367-2630/aaf65f

36. A. Tagliavini, C. Hille, F.B. Kugler, S. Andergassen, A. Toschi, C. Honerkamp, SciPost Phys. **6**, 9 (2019). https://doi.org/10.21468/SciPostPhys.6.1.009
37. C. Hille, F.B. Kugler, C.J. Eckhardt, Y.Y. He, A. Kauch, C. Honerkamp, A. Toschi, S. Andergassen, Phys. Rev. Res. **2**, 033372 (2020). https://doi.org/10.1103/PhysRevResearch.2.033372
38. P. Chalupa-Gantner, F.B. Kugler, C. Hille, J. von Delft, S. Andergassen, A. Toschi, Phys. Rev. Research **4**, 023050 (2022). https://doi.org/10.1103/PhysRevResearch.4.023050
39. `PFFRGSolver.jl` repository. https://github.com/dominikkiese/PFFRGSolver.jl
40. N. Niggemann, B. Sbierski, J. Reuther, Phys. Rev. B **103**(10) (2021). https://doi.org/10.1103/PhysRevB.103.104431
41. N. Niggemann, J. Reuther, B. Sbierski, SciPost Phys. **12**, 156 (2022). https://doi.org/10.21468/SciPostPhys.12.5.156
42. B. Roulet, J. Gavoret, P. Nozières, Phys. Rev. **178**, 1072 (1969). https://doi.org/10.1103/PhysRev.178.1072
43. N. Wentzell, G. Li, A. Tagliavini, C. Taranto, G. Rohringer, K. Held, A. Toschi, S. Andergassen, Phys. Rev. B **102**, 085106 (2020). https://doi.org/10.1103/PhysRevB.102.085106
44. G. Li, N. Wentzell, P. Pudleiner, P. Thunström, K. Held, Phys. Rev. B **93**, 165103 (2016). https://doi.org/10.1103/PhysRevB.93.165103
45. F.L. Buessen, V. Noculak, S. Trebst, J. Reuther, Phys. Rev. B **100**, 125164 (2019). https://doi.org/10.1103/PhysRevB.100.125164
46. F.L. Buessen, A functional renormalization group perspective on quantum spin liquids in three-dimensional frustrated magnets. Ph.D. thesis, University of Cologne (2019). https://kups.ub.uni-koeln.de/9986/

2.6 Discussion and outlook on pseudofermion fRG

This section discusses pseudofermion fRG from a more general perspective based on our results that have been published in publication [P1], the preprint of Ref. [65], my master's thesis [67], as well as Gün Günal's master's thesis [68]. A general review is available in Ref. [59].

As mentioned before, the pffRG offers great versatility concerning the systems it can be applied to. It is often used to map out phase diagrams of models that are problematic for other methods, notably frustrated systems hosting putative QSL phases, such as Heisenberg models with antiferromagnetic interactions on the Kagome [65] and pyrochlore lattices [67]. The lack of a kinetic term in the pseudofermion Hamiltonian (2.22) corresponds to an infinite interaction limit, $U/t \to \infty$, of a Hubbard-like Hamiltonian. It is a priori unclear why a method based on weak-coupling expansion, such as fRG, is applicable.

This issue can in part be investigated by systematically raising the loop order in a multiloop pffRG scheme, thus introducing more diagrams into the truncated flow equations. Ground states with classical order induce a divergence of some susceptibility component, though the presence of this divergence may depend on loop order [P1]. For disordered states, reaching $\Lambda = 0$ is likewise not possible, since the susceptibility tends to slowly diverge with decreasing Λ [P1, 69]. In this case, different loop orders tend to differ more as Λ is decreased [65, 69], and it should be noted that statements about loop convergence in pffRG can only be made with respect to a minimum value of Λ, beyond which higher loop orders are necessary for convergence. Obtaining results that are converged in loop order for disordered phases at small values of Λ is therefore numerically expensive. It should be emphasized that the weak-coupling nature of the fRG is not overcome by multiloop terms, which only improve the fRG flow within the parquet approximation.

As was already pointed out in publication [P1], evaluating the flow equations close to a phase transition or at small values of Λ in an error-controlled way requires careful implemented of all numerical algorithms. For instance, an ODE solver without error control may cross a phase transition and emit qualitatively inaccurate data, which has to be rejected for physical reasons. In this situation, an ODE solver with step size control decreases the step size, such that the phase transition is approached in a controlled manner. Both at small values of Λ and close to a divergence, the vertex develops very sharp features in frequency space. These features can be parameterized with an adaptive frequency grid, as was done in our publications [P1, 65, 67, 68]. Since some numerical parameters of the grid have to be carefully adjusted for each model, considerable effort is required to set up pffRG calculations for each new model. This motivated a search for a more flexible and robust parameterization of vertices and other correlators. In the next two chapters, we show that quantics tensor trains (QTT) are able to fill this role.

Even with these technical improvements, pffRG is still inherently limited by its ill-justified truncation of a weak-coupling expansion, and the only approximate fulfillment of the pseudofermion constraint (2.21). It is therefore doubtful if the pffRG can be improved to a quantitatively accurate method. Within the current state of the art, pffRG is more suitable for qualitative exploration of phase diagrams of quantum magnetic systems, supplementing mean-field approaches, as demonstrated by e.g., Ref. [70].

Chapter Three

Compressed tensor train representations of functions

3.1 Introduction to tensor networks

Tensor networks are a family of numerical methods in quantum many-body physics and the study of strongly correlated electron systems. These methods rely on a formalism where quantum mechanical states and operators are represented by tensors, i.e. multilinear operators respecting certain transformation properties [71]. Multilinear algebra is then used to obtain approximate eigenstates of a Hamiltonian, which lives in an exponentially large Hilbert space.

This section introduces matrix product states and the density matrix renormalization group (DMRG), two concepts that are necessary to understand publications [P2] and [P3]. It also explains the graphical notation which is extensively used in both publications, as well as in the tensor networks literature.

3.1.1 Matrix product states and the curse of dimensionality

The state space of a quantum system grows exponentially with the number of particles. This well-known property forms the basis for the theoretical advantage that quantum computing would offer. Conversely, it becomes a computational and practical problem when trying to simulate a quantum system on a classical computer. In condensed matter physics, the systems to be simulated are typically given by second-quantized lattice Hamiltonians on N sites, where N is on the order of 10^{23}. Now, if each site admits d states, the full many-body state space is d^N-dimensional. How can we encode a state $|\Psi\rangle$ in such a high-dimensional space? The naive approach is to expand the state in an orthonormal single-particle basis $\{|1\rangle, \ldots, |d\rangle\}$ as

$$|\Psi\rangle = \sum_{\sigma_1=1}^{d} \cdots \sum_{\sigma_N=1}^{d} A^{\sigma_1,\ldots,\sigma_N} |\sigma_1\rangle \otimes \cdots \otimes |\sigma_N\rangle \tag{3.1}$$

which then requires keeping d^N coefficients $A^{\sigma_1,\dots,\sigma_N}$ in memory. In the same spirit, linear operators such as the Hamiltonian are matrices with $d^N \times d^N$ entries. Even with sophisticated numerical techniques, exact diagonalization of the Hamiltonian is feasible only up to $N \approx 40$ in the current state of the art [4]. This exponential growth of computational complexity with dimensionality is known as the *curse of dimensionality* [20].

Given the enormous gap between exactly solvable $N = 40$, and desired system size $N = 10^{23}$, we have to accept approximate solutions in exchange for larger systems. One way to do so is to truncate the Hilbert space to some subspace, and approximate $|\Psi\rangle$ using only that subspace. The simplest such subspace is the space of product states[1]

$$|\Psi\rangle \approx |\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_N\rangle \,. \tag{3.2}$$

The single-particle states $|\psi_j\rangle$ can be expanded as

$$|\psi_j\rangle = \textstyle\sum_{\sigma=1}^{d} a_j^\sigma \,|\sigma\rangle\,, \quad \text{where } a_j^\sigma = \langle\sigma|\psi_j\rangle\,, \tag{3.3}$$

and therefore

$$\begin{aligned} |\Psi\rangle &\approx \left[\textstyle\sum_{\sigma_1} a_1^{\sigma_1} |\sigma_1\rangle\right] \otimes \left[\textstyle\sum_{\sigma_2} a_2^{\sigma_2} |\sigma_2\rangle\right] \otimes \cdots \otimes \left[\textstyle\sum_{\sigma_N} a_N^{\sigma_N} |\sigma_N\rangle\right] \\ &= \textstyle\sum_{\sigma_1} \cdots \sum_{\sigma_N} a_1^{\sigma_1} \cdots a_N^{\sigma_N} \,|\sigma_1\rangle \otimes \cdots \otimes |\sigma_N\rangle\,. \end{aligned} \tag{3.4a}$$

Observe that instead of the d^N coefficients that would be necessary to encode an arbitrary state within the N-particle Hilbert space, there are only $N \times d$ coefficients: we have now limited ourselves to a $N \times d$-dimensional subspace. Computations such as diagonalization of the Hamiltonian are now feasible within that subspace, which is the basis of the Hartree–Fock approach.

The price for truncating the state space is a loss of expressiveness:[2] we have neglected all entanglement between particles, and all associated correlations. In many systems, such as the quantum magnetic systems considered in the previous chapter, it is precisely those states that are their most interesting feature. Restricting the Hilbert space to product states is equivalent to approximating the exponentially large tensor $A^{\sigma_1,\dots,\sigma_N}$ using a product of scalars,

$$A^{\sigma_1,\dots,\sigma_N} \approx a_1^{\sigma_1} \cdots a_N^{\sigma_N} \,. \tag{3.5}$$

A small step towards more complexity, without losing the ability to easily manipulate an explicit representation of states, is to promote the scalar coefficients a_j^σ to $\chi \times \chi$ matrices M_j^σ:

$$A^{\sigma_1,\dots,\sigma_N} \approx M_1^{\sigma_1} \cdots M_N^{\sigma_N}; \tag{3.6a}$$

[1] Some complications arise from the (anti-)symmetry of wave functions of indistinguishable particles here, which will be ignored in this brief overview. A discussion of how to implement these symmetries in tensor networks can be found in Ref. [72].

[2] The term *expressiveness* is lifted from the machine learning literature. In intuitive terms, a more expressive model is able to model a larger class of functions compared to a less expressive one.

$$|\Psi\rangle \approx \sum_{\sigma_1} \cdots \sum_{\sigma_N} M_1^{\sigma_1} \cdots M_N^{\sigma_N} |\sigma_1\rangle \otimes \cdots \otimes |\sigma_N\rangle \,. \tag{3.6b}$$

This is known as a *matrix product state* (MPS) [8], and χ is called the *bond dimension*.

Matrix product states thus constitute a middle ground between the simplicity of product states and complexity of the full state space. The benefit of this additional complexity is mainly the ability of MPS to efficiently encode states with *area-law entanglement* [7, 8]. When a system in a state with area-law entanglement is partitioned into two subsystems, the entanglement entropy between the subsystems scales with the area of the dividing surface [6, 8]. Although these states form only a small part of the full Hilbert space, ground states of Hamiltonians with local interactions and gapped excitations fall into this category. Finding the ground state of such a Hamiltonian is a very common problem in condensed matter physics. This is the basis for the tremendous success of MPS-based methods, most famously the DMRG introduced below.

3.1.2 Graphical notation

To discuss more advanced tensor network methods, it is worthwhile to first introduce a graphical notation for tensors and their contractions. This graphical notation serves to overcome some weaknesses of the usual mathematical notation of tensors: When considering a contraction of multiple tensors, it is necessary to identify those indices attached to different tensors that have the same name if one tries to understand the structure of the contraction pattern. Index names are arbitrary and carry little meaning; it is the property of being shared between different tensors and being summed over that is important. Therefore, it is much more convenient to draw a tensor contraction pattern as a graph, where the vertices represent tensors and edges are indices. Two tensors being contracted then correspond to two vertices connected by an edge, and dangling edges only connected to one vertex correspond to open indices. This way, the graph's topology immediately corresponds to the underlying contraction pattern in an intuitive way.

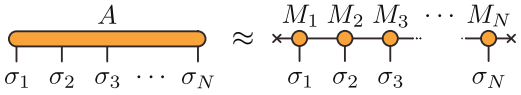
It is customary to draw tensors as filled circles or polygons, and to attach edges at different points on the polygon according to the meaning of the index. For example, a tensor M_{ab}^{σ} might be drawn as follows:

$M_{ab}^{\sigma} =:$ 

(3.7)

Thus, an MPS corresponds to the following graph:

$$A^{\sigma_1 \cdots \sigma_N} \approx \sum_{a_0=1}^{1} \sum_{a_1=1}^{\chi} \cdots \sum_{a_{N-1}=1}^{\chi} \sum_{a_N=1}^{1} \left[M_1^{\sigma_1}\right]_{a_0 a_1} \left[M_2^{\sigma_2}\right]_{a_1 a_2} \cdots \left[M_N^{\sigma_N}\right]_{a_{N-1} a_N} \tag{3.8a}$$



(3.8b)

The indices $a_0 \dots a_N$ are unnamed graphical notation, and their summation range is implicitly given by the dimension of the tensors being contracted. Here and in the following, indices named $\sigma, \tau, \dots$ are always external indices, and indices named α, β, or $a, b, \dots$ are always internal (contracted) indices.

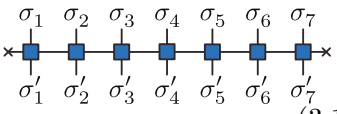
3.1.3 Density matrix renormalization group

Tensor networks are the basis for a plethora of algorithms for classical simulation of many-body quantum physics, the most famous of which is the density matrix renormalization group (DMRG) mentioned earlier [8, 15, 16]. DMRG can be described as a variational optimization of an MPS *ansatz* for the wave function [7, 8]. In a similar way, operators such as the Hamiltonian can be decomposed into a matrix product *operator* (MPO), given by

$$H = \sum_{\sigma_1} \sum_{\sigma_1'} \cdots \sum_{\sigma_N} \sum_{\sigma_N'} H_1^{\sigma_1' \sigma_1} \cdots H_N^{\sigma_N' \sigma_N} \left|\sigma_1'\right\rangle \left\langle\sigma_1\right| \otimes \cdots \otimes \left|\sigma_N'\right\rangle \left\langle\sigma_N\right|, \tag{3.9}$$

where each $H_\ell^{\sigma_\ell \sigma_\ell'}$ is a matrix [8]. For a system with $N = 7$ sites, this is

$$H_1^{\sigma_1' \sigma_1} H_2^{\sigma_2' \sigma_2} H_3^{\sigma_3' \sigma_3} H_4^{\sigma_4' \sigma_4} H_5^{\sigma_5' \sigma_5} H_6^{\sigma_6' \sigma_6} H_7^{\sigma_7' \sigma_7} =$$

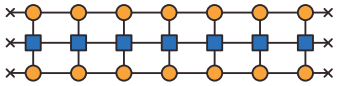


(3.10)

in graphical notation. Then, the energy expectation value to be optimized is

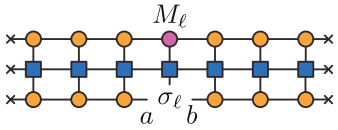
$$\langle \Psi | H | \Psi \rangle = \sum_{\sigma_1} \sum_{\sigma_1'} \cdots \sum_{\sigma_N} \sum_{\sigma_N'} \left[M_1^{\sigma_1'} \cdots M_N^{\sigma_N'} \right]^* H_1^{\sigma_1' \sigma_1} \cdots H_N^{\sigma_N' \sigma_N} \left[M_1^{\sigma_1} \cdots M_N^{\sigma_N} \right] \tag{3.11a}$$

$=$



. (3.11b)

Complex conjugation is denoted graphically by mirroring the MPS on the horizontal axis. Now, the energy expectation value can be optimized site-by-site, solving the eigenvalue problem



(3.12)

on each site [8]. This is known as a 1-site update. The update equation (3.12) can be evaluated very efficiently by exploiting isometry structures in the MPS tensors M_ℓ [7, 8]. Successive 1-site updates to each tensor M_ℓ, alternating between forward sweeps $\ell \leftarrow 1, 2, 3, \dots, N$ and backward sweeps

$\ell \leftarrow N, \ldots, 3, 2, 1$, then lead to an iterative global optimization in the ideal case. In practice, this simplistic optimization with 1-site update is prone to get stuck in local minima, and more advanced update schemes are necessary, such as 2-site updates or the controlled bond expansion scheme [73].

The DMRG algorithm is an example of a general strategy commonly used when working with tensor networks [7, 71]: A tensor network that represents some wave function is set up formally, but not initially contracted, since that would be exponentially expensive. Instead, single tensors within the tensor network are optimized with respect to some cost function, in this case, the energy expectation value $\langle\Psi|H|\Psi\rangle$. The computational effort necessary for this optimization step is minimized by exploiting structures in the tensors as much as possible, most importantly the order in which the necessary tensor contractions are evaluated. Some of these techniques are also shown in this chapter, mainly in publication [P3].

3.2 Function representation with tensor trains

From a more general point of view, the tensor network methods introduced in the previous section are useful because one is only interested in the low-energy states of the system. For a state to have low energy, it has to satisfy some properties imposed by the Hamiltonian, which imply certain structures dependent on that Hamiltonian. States with this structure typically form a manifold that has a much smaller dimensionality than the full Hilbert space, which permits a compressed, yet highly accurate representation. In the case of MPS, the structure being exploited for compression is the aforementioned area-law entanglement induced by local Hamiltonians [6–8].

High-dimensional function spaces that contain a small subset of interesting functions are commonly encountered in physics, other natural sciences, engineering, and applied mathematics. Finding a parameterization of this subset is therefore a problem for which many solutions have been developed, amongst them the aforementioned MPS and tensor network techniques. Though our choice of representation is based on the physics of the system, the linear algebra operations that decompose a tensor into an MPS on the technical level do not require the full tensor to be a quantum mechanical wave function. Can an MPS represent functions other than wave functions, and if yes, which functions are representable efficiently? A structurally equivalent scheme known as a *tensor train* (TT) has indeed been used in applied mathematics to compress functions [20, 21]. The two publications in this chapter, [P2] and [P3], are a partial answer to this question in the physics context. Publication [P2] focuses on one specific variant, where a function is regarded as a tensor with indices corresponding to binary digits of its function argument. A TT decomposition of this tensor is generated using only very few samples of the function with a machine learning algorithm, the so-called *tensor cross interpolation* (TCI).

Publication [P3] describes TCI more generally, including new variants and improvements. It also presents a variety of different applications, as well as an open-source library that can be used to perform such decompositions.

3.3 Publication 2: Quantics tensor cross interpolation for high-resolution, parsimonious representations of multivariate functions

In this section, the following publication is reprinted:

P2 *Quantics tensor cross interpolation for high-resolution, parsimonious representations of multivariate functions*,
Marc K. Ritter, Yuriel Núñez Fernández, Markus Wallerberger, Jan von Delft, Hiroshi Shinaoka, and Xavier Waintal,
Physical Review Letters **132**, 056501 (2024),
doi:10.1103/PhysRevLett.132.056501.
Reprinted on pages 36–43.

Quantics Tensor Cross Interpolation for High-Resolution Parsimonious Representations of Multivariate Functions

Marc K. Ritter,[1,*] Yuriel Núñez Fernández,[2] Markus Wallerberger,[3] Jan von Delft,[1] Hiroshi Shinaoka,[4] and Xavier Waintal[2]

[1]Arnold Sommerfeld Center for Theoretical Physics, Center for NanoScience, and Munich Center for Quantum Science and Technology, Ludwig-Maximilians-Universität München, 80333 Munich, Germany
[2]Université Grenoble Alpes, CEA, Grenoble INP, IRIG, Pheliqs, F-38000 Grenoble, France
[3]Institute of Solid State Physics, TU Wien, 1040 Vienna, Austria
[4]Department of Physics, Saitama University, Saitama 338-8570, Japan

(Received 12 April 2023; accepted 25 October 2023; published 29 January 2024)

Multivariate functions of continuous variables arise in countless branches of science. Numerical computations with such functions typically involve a compromise between two contrary desiderata: accurate resolution of the functional dependence, versus parsimonious memory usage. Recently, two promising strategies have emerged for satisfying both requirements: (i) The *quantics* representation, which expresses functions as multi-index tensors, with each index representing one bit of a binary encoding of one of the variables; and (ii) *tensor cross interpolation* (TCI), which, if applicable, yields parsimonious interpolations for multi-index tensors. Here, we present a strategy, *quantics TCI*, which combines the advantages of both schemes. We illustrate its potential with an application from condensed matter physics: the computation of Brillouin zone integrals.

DOI: 10.1103/PhysRevLett.132.056501

Introduction.—Let f be a multivariate function of n continuous real variables u_i $(i = 1, \ldots, n)$:

$$f\colon U \subset \mathbb{R}^n \to \mathbb{C}, \qquad \mathbf{u} = (u_1, \ldots, u_n) \mapsto f(\mathbf{u}). \quad (1)$$

Such functions arise in essentially all branches of science. In physics, e.g., they could stand for the fields used in classical or quantum field theories, with $\mathbf{u} = (\mathbf{x}, t)$ or $\mathbf{u} = (\mathbf{k}, \omega)$ representing space-time or momentum-frequency variables in $n = \mathcal{D} + 1$ dimensions, respectively; or for m-point correlation functions of such fields, with $\mathbf{u} = (\mathbf{x}_1, t_1, \ldots, \mathbf{x}_m, t_m)$ and $n = m(\mathcal{D} + 1)$, etc.

Often such functions have structure (peaks, wiggles, divergences, even discontinuities) on length scales, time-scales, or momentum or frequency scales differing by orders of magnitude. Then, their numerical treatment is challenging due to two contrary requirements: On the one hand, accurate resolution of small-scale structures requires a fine-grained discretization grid, while large-scale structures require a large domain of definition U; and, on the other hand, memory usage should be parsimonious, hence a fine-grained grid cannot be used throughout U. In practice, compromises are needed, sacrificing resolution and/or restricting U to limit memory costs, or using nonuniform grid spacings to resolve some parts of U more finely than others.

Very recently, in different branches of physics, it was pointed out that if the structures in f exhibit *scale separation*, in a sense made precise below, they can be encoded both accurately and parsimoniously, on both small and large scales [1–5]. This is done using a representation first discussed in the context of quantum information [6–9], independently introduced in the mathematics literature by Oseledets [10], and dubbed the *quantics* representation by Khoromskij [11]: it encodes each variable u_i through R binary digits, or bits, and expresses $f(\mathbf{u})$ as a multi-index tensor $f_{\sigma_1 \ldots \sigma_\mathcal{L}}$, with $\mathcal{L} = nR$, where each index represents a bit. If f exhibits scale separation, this tensor is highly compressible, i.e., it can be well approximated by a tensor train (TT) of fairly low rank. These previous works found the TT via singular value decomposition (SVD) of the full tensor, demonstrating that low-rank quantics TT (QTT) representations *exist*. It remains to design more practical algorithms to find them, since the computational costs of the SVD approach grow exponentially with $\mathcal{L}$.

In an unrelated very recent development [12], TT representations were used for multivariate correlation functions arising in diagrammatic Monte Carlo methods (albeit without using the quantics encoding). It was found that these TTs are not only highly compressible, but that the compression can be achieved very efficiently using the tensor cross interpolation (TCI) algorithm. This technique, pioneered by Oseledets and coworkers [13–15] and improved by Dolgov and Savostyanov [16,17], is computationally exponentially cheaper than SVDs (albeit theoretically less optimal, though with controlled errors [16]).

The purpose of this Letter is to point out that quantics TTs and TCI can be combined. This leads to a strategy that

we call quantics tensor cross interpolation (QTCI). It has several highly desirable properties: (i) Arbitrary resolution via an exponentially large grid with 2^R points for each variable, obtained at a cost linear in R. (ii) Efficient construction of the QTCI at cost linear in $\mathcal{L} = nR$. (iii) Access to many ultrafast algorithms once the QTCI has been obtained [1,5,18–20]; for instance, integrals, convolutions, and Fourier transforms can be computed at $\mathcal{O}(\mathcal{L})$ costs (i.e., exponentially cheaper than standard fast Fourier transforms) [21]. (iv) More generally, TT representations yield access to a whole range of matrix product states or matrix product operators (MPS/MPO) algorithms which were devised in the context of many-body physics [18] and have spawned the mathematical field of TTs.

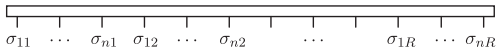
We illustrate the power of QTCI by using it to resolve the momentum dependence of functions defined on the Brillouin zone of the celebrated Haldane model [29]. We construct QTCIs for its noninteracting Green's function and Berry curvature, and compute the Chern number of a band with topological properties.

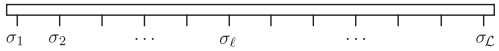
Quantics tensor trains.—For the quantics representation of $f(\mathbf{u})$, each variable u_i is rescaled to lie within the unit interval $I = [0, 1)$, then discretized on a grid of 2^R points and expressed as a tuple of R bits [10,11],

$$u_i = \sum_{b=1}^{R} \frac{\sigma_{ib}}{2^b} \mapsto (\sigma_{i1} \ldots \sigma_{iR}), \qquad \sigma_{ib} \in \{0, 1\}. \tag{2}$$

Here, σ_{ib} resolves the variable u_i at the scale 2^{-b}. Arbitrarily high resolution can be achieved by choosing R sufficiently large. Thus, $\mathbf{u}$ is represented by a tuple of $\mathcal{L} = nR$ bits. To facilitate scale separation, the bits are relabled [10] as $\sigma_\ell = \sigma_{ib}$, using a single index $\ell = i + (b-1)n \in 1, \ldots, \mathcal{L}$. This interleaves them such that all bits σ_{ib} describing the same scale 2^{-b} have contiguous σ_ℓ labels. Then, f can be viewed and graphically depicted as tensor of degree $\mathcal{L}$:

$$f : \{0, 1\}^{\mathcal{L}} \to \mathbb{C}, \quad \boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_{\mathcal{L}}) \mapsto f_{\boldsymbol{\sigma}} = f(\mathbf{u}).$$

$f_{\boldsymbol{\sigma}} =$ 

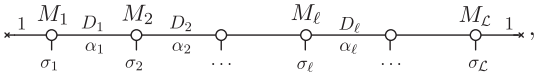
$=$ 

(3)

Alternatively, all same-scale bits can be *fused* together as $\tilde{\sigma}_b = \sum_{i=1}^{n} 2^{i-1} \sigma_{ib} \in \{0, \ldots, 2^n - 1\}$, yielding the *fused* representation $f_{\tilde{\boldsymbol{\sigma}}} = f(\mathbf{u})$, $\tilde{\boldsymbol{\sigma}} = (\tilde{\sigma}_1, \ldots, \tilde{\sigma}_R)$. It employs only $\tilde{\mathcal{L}} = R$ indices, each of dimension $d = 2^n$ [30].

Any tensor can be *unfolded* as a TT [11,13,15,31], graphically depicted as a chain of ℓ sites connected by bonds representing sums over repeated indices:

$$f_{\boldsymbol{\sigma}} = \prod_{\ell=1}^{\mathcal{L}} M_\ell^{\sigma_\ell} = [M_1]^{\sigma_1}_{1\alpha_1} [M_2]^{\sigma_2}_{\alpha_1\alpha_2} \cdots [M_{\mathcal{L}}]^{\sigma_{\mathcal{L}}}_{\alpha_{\mathcal{L}-1}1}$$

$=$ 

(4)

Each site ℓ hosts a three-leg tensor M_ℓ with elements $[M_\ell]^{\sigma_\ell}_{\alpha_{\ell-1}\alpha_\ell}$. Its "local" and "virtual" bond indices, σ_ℓ and $\alpha_{\ell-1}, \alpha_\ell$, have dimensions $d = 2$ and $D_{\ell-1}, D_\ell$, respectively, with $D_0 = D_{\mathcal{L}} = 1$ for the outermost (dummy) bonds.

If $f_{\boldsymbol{\sigma}}$ is full rank, exact TT unfoldings have exponential bond growth towards the chain center, $D_\ell = 2^{\min\{\ell, \mathcal{L}-\ell\}}$, implying exponential memory costs, $\mathcal{O}(2^{\mathcal{L}/2})$. However, tensors $f_{\boldsymbol{\sigma}}$ with lower information content admit accurate TT unfoldings with lower virtual bond dimensions. Such unfoldings are obtained via iterative factorization and truncation of bonds with low information content. Usually, this is done using a sequence of SVDs, discarding all singular values smaller than a specified *truncation threshold* ϵ. The largest D_ℓ value so obtained, $D_{\max}$, is the rank of the ϵ-truncated TT. SVD truncation is provably optimal [15], yielding the smallest possible $D_{\max}$ for specified ϵ. If $D_{\max} \ll 2^{\mathcal{L}/2}$, $f_{\boldsymbol{\sigma}}$ is *strongly compressible*, implying that it has internal structure. Building on the pioneering studies of Oseledets [13,31] and Khoromskij [11], Refs. [2–5] argued that for quantics tensors $f_{\boldsymbol{\sigma}} = f(\mathbf{u})$, strong compressibility reflects scale separation: structures in $f(\mathbf{u})$ occurring on different scales are only "weakly entangled," in that the virtual bonds connecting the corresponding sites in the TT do not require large dimensions.

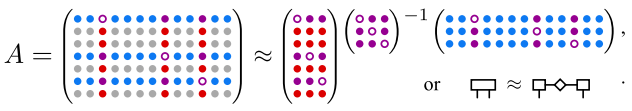
This perspective is informed by the study of one-dimensional quantum lattice models using matrix product states (MPSs)—many-body wave functions of the form (4) [18]. In that context, σ_ℓ labels physical degrees of freedom at site ℓ, and the entanglement of sites ℓ and $\ell + 1$ is characterized by an entanglement entropy bounded by 2^{D_ℓ}. By analogy, if a quantics TT is strongly compressible, requiring only small $D_{\max}$, the sites representing different scales are not strongly entangled—indeed, $D_{\max}$ quantifies the degree of scale separation inherent in $f(\mathbf{u})$.

The SVD unfolding strategy requires knowledge of the full tensor $f_{\boldsymbol{\sigma}}$: it uses $2^{\mathcal{L}}$ function calls, implying exponentially long runtimes, even if $f_{\boldsymbol{\sigma}}$ is strongly compressible. Thus, this strategy is optimally accurate but exponentially inefficient: it uncovers structure in $f_{\boldsymbol{\sigma}}$, but does not exploit it already while constructing the unfolded TT.

Tensor cross interpolation.—The TCI algorithm [12,13,15,17] solves this problem. It serves as a *black box* that samples $f_{\boldsymbol{\sigma}}$ at some clever choices of $\boldsymbol{\sigma}$ and iteratively constructs the TT from the sampled values. TCI is slightly less accurate than SVD unfoldings, requiring a slightly larger $D_{\max}$ for a specified error tolerance ϵ. But it

is exponentially more efficient, needing at most $\mathcal{O}(D_{\max}^2\mathcal{L})$ function evaluations and a run-time of at most $\mathcal{O}(D_{\max}^3\mathcal{L})$.

We refer to [12] for details about TCI in general and its actual implementation. Here, we just sketch the main idea. TCI achieves the factorization needed for unfolding by employing matrix cross interpolation (MCI) rather than SVD. Given a matrix A, the MCI formula approximates it as $A \approx CP^{-1}R = \tilde{A}$, graphically depicted as follows:



Here, the column, row and pivot matrices C, R, and P, are all constructed from elements of A: C contains D columns (red), R contains D rows (blue), and P their intersections, the *pivots* (purple). The resulting $\tilde{A}$ exactly reproduces all elements of A contained in C and R; the remaining elements are in effect interpolated from the "crosses" formed by these (hence "cross interpolation"). The accuracy of the interpolation depends on the number and choice of pivots; it can be improved systematically by adding more pivots. If $D = \mathrm{rank}(A)$, one can obtain an exact representation of the full matrix, $A = \tilde{A}$ [15].

Tensors can be unfolded by iteratively using MCI while treating multiple indices (e.g., $\sigma_2 \ldots \sigma_\mathcal{L}$) as a single composite index, e.g., $f_{\sigma_1\sigma_2\ldots\sigma_\mathcal{L}} \approx [C_1]^{\sigma_1}_{1\beta_1}[P_1^{-1}]_{\beta_1\alpha_1}[R_1]^{\sigma_2\ldots\sigma_\mathcal{L}}_{\alpha_1}$. Iterative application of MCI to each new tensor on the right ultimately yields a fully unfolded TT, $f_{\boldsymbol{\sigma}} \approx f^{\mathrm{QTCI}}_{\boldsymbol{\sigma}}$:

[A TT of the form (4) is obtained by defining $M_\ell = C_\ell P_\ell^{-1}$, -◇- = -□◇-.] This naive algorithm is inefficient, but illustrates how the interpolation properties of MCI carry over to TCI. In practice, it is more efficient to use a sweeping algorithm, successively sampling more function values $f_{\boldsymbol{\sigma}}$ and adding pivots to each tensor until the relative error ε, which decreases during sweeping, drops below a specified tolerance ϵ [21]. We define ε as $\max_{\boldsymbol{\sigma}\in S}|f^{\mathrm{QTCI}}_{\boldsymbol{\sigma}} - f_{\boldsymbol{\sigma}}|/\max_{\boldsymbol{\sigma}\in S}|f_{\boldsymbol{\sigma}}|$, where S is the set of all $\boldsymbol{\sigma}$ index values sampled while constructing $f^{\mathrm{QTCI}}_{\boldsymbol{\sigma}}$.

Integration.—The integral over a function f in QTT form is easily accessible in $\mathcal{O}(D_{\max}^2\mathcal{L})$ steps [12,15]. It can be approximated by a Riemann sum since the quantics grid is exponentially fine, and all σ_ℓ sums can be performed independently due to the TT's factorized form:

$$\int_{I^n} d^n\mathbf{u} f(\mathbf{u}) \approx \frac{1}{2^\mathcal{L}} \sum_{\boldsymbol{\sigma}} f_{\boldsymbol{\sigma}} \approx \frac{1}{2^\mathcal{L}} \prod_\ell \sum_{\sigma_\ell} [M_\ell]^{\sigma_\ell}. \quad (5)$$

1D example.—We first demonstrate QTCI for computing the integral $I[f] = \int_0^{\ln(20)} dx\, f(x)$ of the function

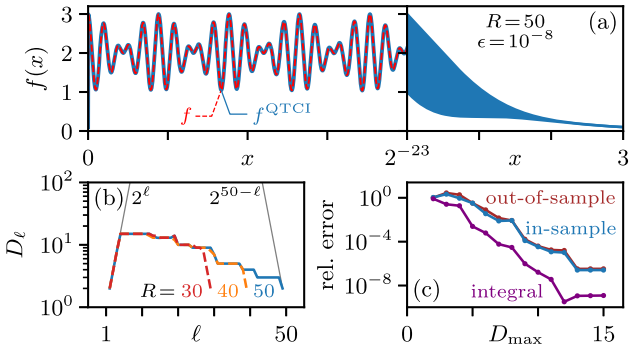


FIG. 1. QTCI representation of a rapidly oscillating function, for $\mathcal{L} = R = 50$ with tolerance $\epsilon = 10^{-8}$. (a) Plot of $f(x)$ (see text). Left: the interval $x \in [0; 2^{-23}]$; red dashed: the actual function, blue: its QTCI representation. Right: the envelope structure up to $x = \log(20) \approx 3$; the rapid oscillations are not resolvable on this scale. (b) Virtual bond dimensions D_ℓ of the QTT, for $R = 30$, 40, 50. Gray lines indicate how $D_\ell^{R=50}$ would grow without any truncation. (c) Relative error estimates as a function of $D_{\max} = \max(D_\ell)$, for $R = 50$.

$f(x) = \cos(x/B)\cos(x/4\sqrt{5}B)e^{-x^2} + 2e^{-x}$ with $B = 2^{-30} \approx 10^{-9}$. This function, shown in Fig. 1(a), involves structure on widely different scales: rapid, incommensurate oscillations and a slowly decaying envelope. A standard representation thereof on an equidistant mesh would require much more than $\mathcal{O}(1/B)$ sampling points, as would the computation of the integral $I[f] = (19/10) + \mathcal{O}(e^{-1/(4B^2)})$. By contrast, for a quantics representation, it suffices to choose R somewhat larger than 30 (ensuring $2^{-R} \ll B$); and since the information content of $f(x)$ is not very high, $f_{\boldsymbol{\sigma}}$ is strongly compressible. We unfolded it using QTCI with $\epsilon = 10^{-8}$ and $R = 50$ (quite a bit larger than 30, just to demonstrate the capabilities of TCI). Figure 1(b) shows the resulting profile of D_ℓ vs ℓ, revealing the scale separation inherent in $f(x)$: the initial growth of the bond dimension, $D_\ell \sim e^\ell$, quickly stops at a fairly small maximum, $D_{\max} = 15$, confirming strong compressibility; thereafter, D_ℓ decreases steadily with ℓ, becoming $\mathcal{O}(1)$ for ℓ larger than 30, since $f(x)$ has very little structure at scales below 2^{-30}. Remarkably, although $f_{\boldsymbol{\sigma}}$ has $2^{50} \approx 10^{15}$ elements, the TCI algorithm finds the relevant structure using only 8706 samples, i.e., roughly 1 sample per 59000 oscillations. Nevertheless, it yields an accurate representation of $f(x)$: the in-sample error, the out-of-sample error (defined as maximum error over 2000 random samples) [32], and the error for the integral $I[f]$, computed via Eq. (5), all decrease exponentially with $D_{\max}$ [Fig. 1(c)]. The runtimes for computing $I[f]$ using QTCI or adaptive Gauss-Kronrod quadrature are 44 ms vs 6 h on an Intel Xeon W-2245 processor, illustrating the efficiency of QTCI vs conventional approaches.

Haldane model.—As an example with relevance in physics, we apply QTCI to the Green's function and Berry curvature of the well-known Haldane model [29].

It is one of the simplest models with topological properties, yet produces nontrivial structure with multiple peaks and sign changes in reciprocal space. Its Bloch Hamiltonian is

$$H(\mathbf{k}) = \sum_{i=1}^{3}\left[\sigma^1 \cos(\mathbf{k}\cdot\mathbf{a}_i) + \sigma^2 \sin(\mathbf{k}\cdot\mathbf{a}_i)\right] + \sigma^3\left[m - 2t_2\sum_{i=1}^{3}\sin(\mathbf{k}\cdot\mathbf{b}_i)\right], \quad (6)$$

where σ^μ are Pauli matrices, $\mathbf{k} = (k_x, k_y)$, while $\mathbf{a}_{1,2,3}$ connect nearest neighbors and $\mathbf{b}_{1,2,3}$ next-nearest neighbors of a honeycomb lattice. Compared to Haldane's more general version of the model, we fix his parameters $\phi = (\pi/2)$, $t_1 = 1$ and set $t_2 = 0.1$. The parameter m tunes the model through two phase transitions: $|m| < m_c = t_2 3\sqrt{3}$ yields a Chern insulator with Chern number $\mathcal{C} = -1$, and $|m| > m_c$ a trivial phase with $\mathcal{C} = 0$ [29]. At $m = \pm m_c$, a single Dirac point appears at $\mathbf{k} = (\mp\frac{4}{3}\pi, 0)$ and symmetry-related $\mathbf{k}$; there, the Chern number is $\mathcal{C} = -\frac{1}{2}$ [34].

Green's function in reciprocal space.—To illustrate QTCI for the Haldane model, we study the momentum dependence of the Green's function, $G(\mathbf{k}, i\omega_0) = \mathrm{Tr}[(i\omega_0 - H(\mathbf{k}) + \mu)^{-1}]$, with $\omega_0 = \pi/\beta$ the lowest fermionic Matsubara frequency and Tr traces over the 2×2 space of H.

Figure 2(a) shows an intensity plot of the QTCI representation of G in reciprocal space; Fig. 2(b) shows that the relative error with respect to the exact value is below 10^{-5} throughout, hence the momentum dependence is captured accurately. There are small Fermi surfaces around $\mathbf{k} = (-\frac{4}{3}\pi, 0)$ and symmetry-related $\mathbf{k}$. To construct QTTs, we define $f_{\boldsymbol{\sigma}} = G(\mathbf{k}, i\pi/\beta)$, where $\boldsymbol{\sigma}$ encodes $\mathbf{k}$ and β is fixed. Figure 2(c) shows the relative in-sample error as a function of $D_{\max}$ for TTs constructed with $R = 10$ for $\beta = 16, 64, 512$, using either SVD or TCI. For both, the error decreases exponentially as $D_{\max}$ increases. Moreover, TCI is nearly optimal, achieving the same error as SVD for a $D_{\max}$ that is only a few percent larger.

Figure 2(d) shows how SVD and TCI runtimes depend on the number of bits, R, for a fixed $D_{\max}$ at large $\beta = 512$, where the features in G are sharp. The times, including function evaluations, were measured on a single CPU core of AMD EPYC 7702P. The SVD runtimes become prohibitively large for $R > 10$ due to exponential scaling; by contrast, the TCI runtimes depend only mildly on R.

Figure 2(e) shows how TCI profiles of D_ℓ vs ℓ depend on R, for $\beta = 512$ and a specified error tolerance $\epsilon = 10^{-5}$. The bond dimension initially grows as $D_\ell \sim 2^\ell$, reaches a maximum near $\ell \approx 20$, then decreases back to 1. The curves for $R = 20$ and 30 almost coincide, indicating that a good resolution of the sharp features at $\beta = 512$ requires $R > 20$—well beyond the reach of SVD unfoldings.

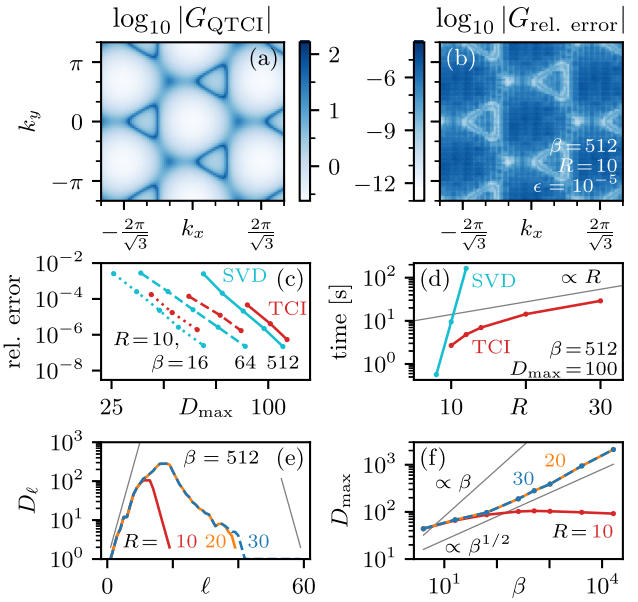


FIG. 2. Green's function $G(\mathbf{k})$ of the Haldane model, computed with error tolerance $\epsilon = 10^{-5}$ throughout. (a),(b) QTCI of the $|G|$ and its relative error, $|G_{\mathrm{QTCI}} - G|/|G|$, for $\beta = 512$, $R = 10$. (c),(d) Comparison of QTT unfoldings via SVD and TCI, showing (c) the relative error vs the maximum bond dimension for $R = 10$ and $\beta = 16$, 64, 512; and (d) runtimes vs R for $\beta = 512$. (e),(f) QTCI bond dimensions for $R = 10$, 20, 30, showing (e) D_ℓ vs ℓ for $\beta = 512$; and (f) $D_{\max}$ vs β.

The low computational cost of TCI allows us to investigate the β dependence of $D_{\max}$, easily reaching $\beta = 2^{14} = 16384$. Figure 2(f) suggests $D_{\max} \propto \beta^\alpha$ with $\alpha \approx 1/2$ for large β. Remarkably, this growth is slower than that, $D_{\max} \propto \beta$, conjectured for a scheme based on SVD and patching [5]. A detailed analysis for general models and higher spatial dimensions is an interesting topic for future research.

Chern number.—Finally, we consider the Chern number, $\mathcal{C}$, for the Haldane model at $\mu = 0$ and $\beta = \infty$. To avoid cumbersome gauge-fixing procedures, we use the gauge-invariant method described in Ref. [35]. First, we discretize the Brillouin zone (BZ) into $2^R \times 2^R$ plaquettes. Then, the Chern number can be obtained from a sum over plaquettes, $\mathcal{C} \approx (1/2\pi i)\sum_{\mathbf{k}\in\mathrm{BZ}} F(\mathbf{k})$, where $F(\mathbf{k}) \approx -i\arg(\langle\psi_{\mathbf{k}_1}|\psi_{\mathbf{k}_2}\rangle\langle\psi_{\mathbf{k}_2}|\psi_{\mathbf{k}_3}\rangle\langle\psi_{\mathbf{k}_3}|\psi_{\mathbf{k}_4}\rangle\langle\psi_{\mathbf{k}_4}|\psi_{\mathbf{k}_1}\rangle)$ is the Berry flux through the plaquette with corners $\mathbf{k}_1 \ldots \mathbf{k}_4$, and $|\psi_{\mathbf{k}}\rangle$ are valence band wave functions.

Close to the transition, for small $\delta_m = m - m_c$, the band gap is $2\delta_m$. This induces peaks of width $\sim\delta_m$ in the Berry flux $F(\mathbf{k})$, shown in Figs. 3(a) and 3(b) for $\delta_m = 10^{-5}$. There, we used a fused quantics representation with $R = 20$, ensuring a mesh spacing 2^{-R} well smaller than δ_m. Whereas a calculation of $\mathcal{C}$ via direct summation or SVD unfolding would require $2^{2R} \approx 10^{12}$ function evaluations, QTCI is much more efficient: for a relative tolerance of $\epsilon = 10^{-10}$, it needed only 4×10^5 samples (and 20 s

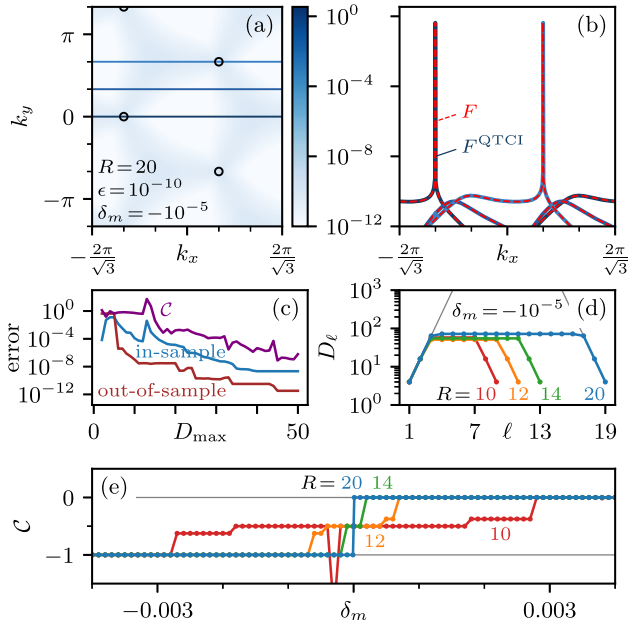


FIG. 3. Evaluation of the Chern number in the Haldane model using QTCI for the Berry flux $F(\mathbf{k})$, with error tolerance $\epsilon = 10^{-10}$ throughout. (a) QTCI of $F(\mathbf{k})$ on the integration domain (circles mark peak positions); (b) cuts through $F(\mathbf{k})$ along the colored lines shown in (a); and (c) errors for $F(\mathbf{k})$ and $\mathcal{C}$ as functions of $D_{\max}$, all computed for $\delta_m = -10^{-5}$, $R = 20$. (d) Bond dimension D_ℓ for QTTs of length $R = 10$, 12, 14, 20. (e) Chern number $\mathcal{C}$ as a function of δ_m, for four choices of R.

runtime on a single core of an Apple M1 processor). It yielded a QTT with maximum bond dimension $D_{\max} = 50$, and a Chern number within 10^{-6} of the expected value $\mathcal{C} = -1$ (see Figs. 3(c) and 3(d)). When plotted as a function of δ_m, $\mathcal{C}$ shows a sharp step from -1 to 0 at $\delta_m = 0$ if computed using $R = 20$ (Fig. 3(e)), beautifully demonstrating that the $\mathbf{k}$ mesh is fine enough. For smaller R the mesh becomes too coarse, incorrectly yielding a plateau at $-1/2$ instead of a sharp step.

For benchmarking purposes, we deliberately chose a model that is analytically solvable. However, our prior knowledge of the peak positions of the Berry curvature was not made available to TCI. This demonstrates its reliability in finding sharp structures, provided enough quantics bits are provided to resolve them. Random sampling misses these sharp structures, which is why in Fig. 3(c) the out-of-sample error, obtained from 2000 random samples, lies well below the in-sample error.

Outlook.—We have shown that the combination of the quantics representation [1–11] with TCI [12,13,15,17] is a powerful tool for uncovering low-rank structures in exponentially large, yet very common objects: functions of few variables resolved with high resolution. Numerical work with such objects always involves truncations—the radically new perspective opened up by QTCI is that they can be performed at *polynomial costs* by discarding weak entanglement between different scales. Once a low-rank QTT has been found, it may be further used within one of the many existing MPO/MPS algorithms [1,5,18–20].

We anticipate that the class of problems for which QTCI can be instrumental is actually very large, reaching well beyond the scope of physics. Intuitively speaking, the only requirement is that the functions should entail some degree of scale separation and not be too irregular (since random structures are not compressible). Thus, a large new research arena, potentially connecting numerous different branches of science, awaits exploration. Fruitful challenges: establish criteria for which types of multivariate functions admit low-rank QTT representations; develop improved algorithms for constructing low-rank approximations to tensors; and above all, explore the use of QTCI for any of the innumerable problems in science requiring high-resolution numerics. The initial diagnosis is easy: simply use SVDs or QTCI [28] to check whether the functions of interest are compressible or not.

We thank Takashi Koretsune, Ivan Oseledets, and Björn Sbierski for inspiring discussions, and Jeongmin Shim for important help at the beginning of this work. We carried out part of the calculations using computer code based on ITensors.jl [36] written in JULIA [37]. H. S. was supported by JSPS KAKENHI Grants No. 21H01041, and No. 21H01003, and JST PRESTO Grant No. JPMJPR2012, Japan. X. W. acknowledges funding from the Plan France 2030 ANR-22-PETQ-0007 "EPIQ"; and J. v. D. from the Deutsche Forschungsgemeinschaft under Germany's Excellence Strategy EXC-2111 (Project No. 390814868), and the Munich Quantum Valley, supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus.

*Corresponding author: ritter.marc@physik.uni-muenchen.de

[1] J. J. García-Ripoll, Quantum-inspired algorithms for multivariate analysis: From interpolation to partial differential equations, Quantum **5**, 431 (2021).
[2] E. Ye and N. F. G. Loureiro, Quantum-inspired method for solving the Vlasov-Poisson equations, Phys. Rev. E **106**, 035208 (2022).
[3] N. Gourianov, M. Lubasch, S. Dolgov, Q. Y. van den Berg, H. Babaee, P. Givi, M. Kiffner, and D. Jaksch, A quantum inspired approach to exploit turbulence structures, Nat. Comput. Sci. **2**, 30 (2022).
[4] N. Gourianov, Exploiting the structure of turbulence with tensor networks, Ph.D. thesis, University of Oxford, 2022.
[5] H. Shinaoka, M. Wallerberger, Y. Murakami, K. Nogaki, R. Sakurai, P. Werner, and A. Kauch, Multiscale space-time ansatz for correlation functions of quantum systems based on quantics tensor trains, Phys. Rev. X **13**, 021015.
[6] S. Wiesner, Simulations of many-body quantum systems by a quantum computer, arXiv:quant-ph/9603028.

[7] C. Zalka, Simulating quantum systems on a quantum computer, Proc. R. Soc. A **454**, 313 (1998).

[8] L. Grover and T. Rudolph, Creating superpositions that correspond to efficiently integrable probability distributions, arXiv:quant-ph/0208112.

[9] J. I. Latorre, Image compression and entanglement, arXiv:quant-ph/0510031.

[10] I. V. Oseledets, Approximation of matrices with logarithmic number of parameters, Dokl. Math. **80**, 653 (2009).

[11] B. N. Khoromskij, $\mathcal{O}(d \log n)$-quantics approximation of $n - d$ tensors in high-dimensional numerical modeling, Constr. Approx. **34**, 257 (2011).

[12] Y. Núñez Fernández, M. Jeannin, P. T. Dumitrescu, T. Kloss, J. Kaye, O. Parcollet, and X. Waintal, Learning Feynman diagrams with tensor trains, Phys. Rev. X **12**, 041018 (2022).

[13] I. V. Oseledets, Tensor-train decomposition, SIAM J. Sci. Comput. **33**, 2295 (2011).

[14] D. Savostyanov and I. Oseledets, Fast adaptive interpolation of multi-dimensional arrays in tensor train format, in The 2011 International Workshop on Multidimensional (nD) Systems (IEEE, Poitiers, France, 2011), pp. 1–8, https://doi.org/10.1109/nDS.2011.6076873.

[15] I. Oseledets and E. Tyrtyshnikov, TT-cross approximation for multidimensional arrays, Linear Algebra Appl. **432**, 70 (2010).

[16] D. V. Savostyanov, Quasioptimality of maximum-volume cross interpolation of tensors, Linear Algebra Appl. **458**, 217 (2014).

[17] S. Dolgov and D. Savostyanov, Parallel cross interpolation for high-precision calculation of high-dimensional integrals, Comput. Phys. Commun. **246**, 106869 (2020).

[18] U. Schollwöck, The density-matrix renormalization group in the age of matrix product states, Ann. Phys. (Amsterdam) **326**, 96 (2011).

[19] M. Lubasch, P. Moinier, and D. Jaksch, Multigrid renormalization, J. Comput. Phys. **372**, 587 (2018).

[20] P. García-Molina, L. Tagliacozzo, and J. J. García-Ripoll, Global optimization of MPS in quantum-inspired numerical analysis, arXiv:2303.09430.

[21] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevLett.132.056501, Sec. S-1 for a brief explanation of the QTT Fourier transform and Sec. S-2 for an overview of the sweeping algorithm. The Supplemental Material contains Refs. [5,22–28].

[22] S. Dolgov, B. Khoromskij, and D. Savostyanov, Superfast Fourier transform using QTT approximation, J. Fourier Anal. Appl. **18**, 915 (2012).

[23] M. Holzäpfel, T. Baumgratz, M. Cramer, and M. B. Plenio, Scalable reconstruction of unitary processes and Hamiltonians, Phys. Rev. A **91**, 042129 (2015).

[24] J. Chen, E. M. Stoudenmire, and S. R. White, Quantum Fourier transform has small entanglement, PRX Quantum **4**, 040318 (2023).

[25] A. Cortinovis, D. Kressner, and S. Massei, On maximum volume submatrices and cross approximation for symmetric semidefinite and diagonally dominant matrices, Linear Algebra Appl. **593**, 251 (2020).

[26] S. A. Goreinov and E. E. Tyrtyshnikov, Quasioptimality of skeleton approximation of a matrix in the Chebyshev norm, Dokl. Math. **83**, 374 (2011).

[27] D. V. Savostyanov, Quasioptimality of maximum-volume cross interpolation of tensors, Linear Algebra Appl. **458**, 217 (2014).

[28] A ready-for-use QTCI toolbox will be published as open source library in the near future.

[29] F. D. M. Haldane, Model for a quantum Hall effect without Landau levels: Condensed-matter realization of the "parity anomaly," Phys. Rev. Lett. **61**, 2015 (1988).

[30] For example, for $n = 2$, $R = 3$, the point $(u_1, u_2) = (\frac{5}{8}, \frac{4}{8})$ has the binary representation (101,100). In the *interleaved* form, the bits are reordered such that $f(\frac{5}{8}, \frac{4}{8})$ is represented by $f_{\boldsymbol{\sigma}} = f_{110010}$; in *fused* form, by $f_{\tilde{\boldsymbol{\sigma}}} = f_{301}$.

[31] I. V. Oseledets, Approximation of $2^d \times 2^d$ matrices using tensor decomposition, SIAM J. Matrix Anal. Appl. **31**, 2130 (2010).

[32] The in-sample error is $\max_{\boldsymbol{\sigma} \in S} |f_{\boldsymbol{\sigma}}^{\mathrm{QTCI}} - f_{\boldsymbol{\sigma}}|$, where S is the set of all $\boldsymbol{\sigma}$ evaluated during construction of the QTCI. For the out-of-sample error, S is instead a set of 2000 pseudorandom $\boldsymbol{\sigma}$ generated by Xoshiro256++ [33]. *Relative* errors are defined as $\max_{\boldsymbol{\sigma} \in S} |f_{\boldsymbol{\sigma}}^{\mathrm{QTCI}} - f_{\boldsymbol{\sigma}}| / |f_{\boldsymbol{\sigma}}|$.

[33] D. Blackman and S. Vigna, Scrambled linear pseudorandom number generators, ACM Trans. Math. Softw. **47**, 36:1(2021).

[34] H. Watanabe, Y. Hatsugai, and H. Aoki, Manipulation of the Dirac cones and the anomaly in the graphene related quantum Hall effect, J. Phys. Conf. Ser. **334**, 012044 (2011).

[35] T. Fukui, Y. Hatsugai, and H. Suzuki, Chern numbers in discretized Brillouin zone: Efficient method of computing (spin) Hall conductances, J. Phys. Soc. Jpn. **74**, 1674 (2005).

[36] M. Fishman, S. White, and E. Stoudenmire, The ITensor software library for tensor network calculations, SciPost Phys. Codebases 4 (2022).

[37] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, JULIA: A fresh approach to numerical computing, SIAM Rev. **59**, 65 (2017).

3.4 Publication 3: Learning tensor networks with tensor cross interpolation: new algorithms and libraries

In this section, the following publication is reprinted:

P3 *Learning tensor networks with tensor cross interpolation: new algorithms and libraries*,
Yuriel Núñez Fernández, Marc K. Ritter, Matthieu Jeannin, Jheng-Wei Li, Thomas Kloss, Thibaud Louvet, Satoshi Terasaki, Olivier Parcollet, Jan von Delft, Hiroshi Shinaoka, and Xavier Waintal,
SciPost Physics **18**, 104 (2025),
doi:10.21468/SciPostPhys.18.3.104.
Reprinted on pages 43–118.

Learning tensor networks with tensor cross interpolation: New algorithms and libraries

Yuriel Núñez Fernández[1,2⋆], Marc K. Ritter[3,4], Matthieu Jeannin[2], Jheng-Wei Li[2], Thomas Kloss[1], Thibaud Louvet[2], Satoshi Terasaki[5], Olivier Parcollet[4,6], Jan von Delft[3], Hiroshi Shinaoka[7] and Xavier Waintal[2†]

1 Université Grenoble Alpes, Neel Institute CNRS, F-38000 Grenoble, France
2 Université Grenoble Alpes, CEA, Grenoble INP, IRIG, Pheliqs, F-38000 Grenoble, France
3 Arnold Sommerfeld Center for Theoretical Physics, Center for NanoScience, and Munich Center for Quantum Science and Technology, Ludwig-Maximilians-Universität München, 80333 Munich, Germany
4 Center for Computational Quantum Physics, Flatiron Institute, 162 5th Avenue, New York, NY 10010, USA
5 AtelierArith, 980-0004, Miyagi, Japan
6 Université Paris-Saclay, CNRS, CEA, Institut de physique théorique, 91191, Gif-sur-Yvette, France
7 Department of Physics, Saitama University, Saitama 338-8570, Japan

⋆ yurielnf@gmail.com , † xavier.waintal@cea.fr

Abstract

The tensor cross interpolation (TCI) algorithm is a rank-revealing algorithm for decomposing low-rank, high-dimensional tensors into tensor trains/matrix product states (MPS). TCI learns a compact MPS representation of the entire object from a tiny training data set. Once obtained, the large existing MPS toolbox provides exponentially fast algorithms for performing a large set of operations. We discuss several improvements and variants of TCI. In particular, we show that replacing the cross interpolation by the partially rank-revealing LU decomposition yields a more stable and more flexible algorithm than the original algorithm. We also present two open source libraries, `xfac` in Python/C++ and `TensorCrossInterpolation.jl` in Julia, that implement these improved algorithms, and illustrate them on several applications. These include sign-problem-free integration in large dimension, the "superhigh-resolution" quantics representation of functions, the solution of partial differential equations, the superfast Fourier transform, the computation of partition functions, and the construction of matrix product operators.

Received 2024-07-23
Accepted 2025-01-20
Published 2025-03-20
doi:10.21468/SciPostPhys.18.3.104

Check for updates

Contents

1 Introduction

Tensor networks, widely used in quantum physics, are increasingly being used also in other areas of science. They offer compressed representations of functions of one or more variables. *A priori*, a tensor of degree $\mathcal{L}$, $F_{\sigma_1 \cdots \sigma_\mathcal{L}}$, with indices $\sigma_\ell = 1, \ldots, d$, requires exponential resources in memory and computation time to be stored and manipulated, since it contains $d^\mathcal{L}$ elements—a manifestation of the well-known *curse of dimensionality*. However, just as a matrix (a tensor of degree 2) can be compressed if it has low rank, a tensor of higher degree can be strongly compressed if it has a low-rank structure. Then, exponential reductions in computational costs for performing standard linear algebra operations are possible, allowing the curse of dimensionality to be evaded.

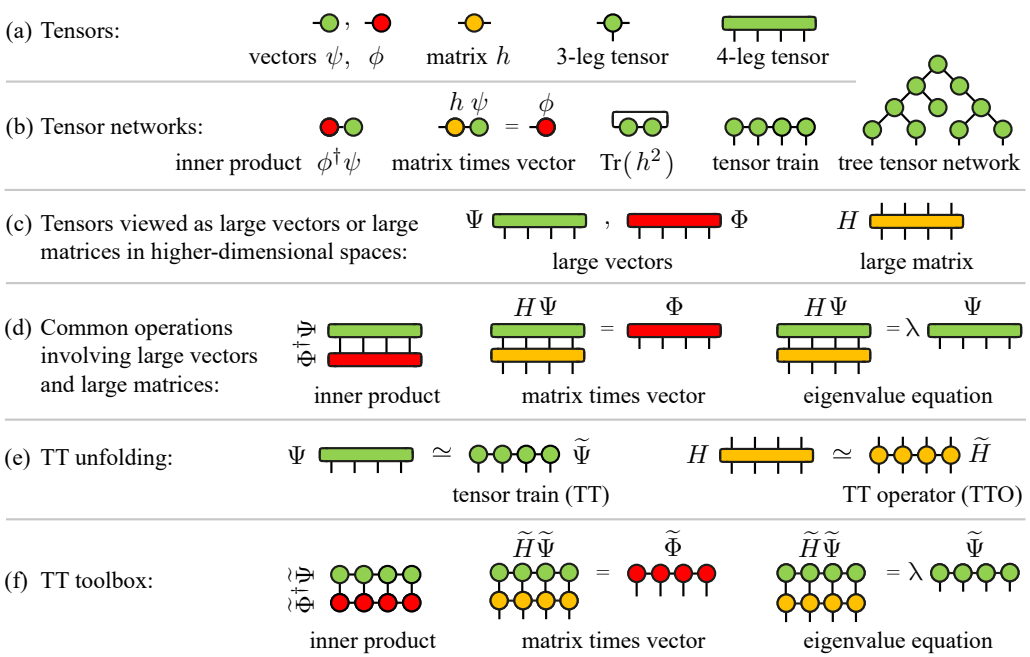


Figure 1: Schematic depiction of key ingredients of the standard MPS toolbox. (a) Colored shapes with legs represent tensors with indices. (b) Tensors connected by bonds, representing sums over shared indices, form tensor networks. (c) Tensors and linear operators acting on them represent large vectors (green, red) and large matrices (yellow) in higher-dimensional vector spaces. (d) Common calculations in these spaces include computing inner products $\Phi^\dagger\Psi$, solving linear problems $H\Psi = \Phi$, computing a few eigenvalues $H\Psi = \lambda\Psi$, and more. (e) Tensors representing large vectors or linear operators can be *unfolded* into MPS or tensor train operators (MPO), respectively. (f) The standard MPS toolbox includes algorithms for performing calculations with MPS and MPO. If these have have low rank, such calculations can be performed in polynomial time, even for exponentially large vector spaces. The `xfac` and `TCI.jl` libraries expand the MPS toolbox by providing tools for unfolding tensors into MPS using exponentially fast tensor cross interpolation (TCI) algorithms, for expressing functions as MPO, and for manipulating the latter.

In physics, functions describing physical quantities and the tensors representing them indeed often do have a hidden structure. A prominent example is the density matrix renormalization group (DMRG), the method of choice for treating one-dimensional quantum lattice models [1]. There, quantum wavefunctions and operators are expressed as tensor networks that in the physics community are called *matrix product states* (MPSs) and *matrix product operators* (MPOs), respectively, or *tensor trains* in the applied mathematics community. (In this work, "MPS" and "tensor train" will be used interchangeably.) Many algorithms for manipulating such objects have been developed in the quantum information and many-body communities [2–5]. We collectively refer to them as the "standard MPS toolbox" [6,7]; Figure 1 depicts some of its ingredients using tensor network diagrams. These algorithms achieve exponential speedup for linear algebra operations (computing scalar products, solving linear systems, diagonalization, ...) with large but compressible vectors and matrices. Although initially developed for many-body physics, the MPS toolbox is increasingly being used in other, seemingly unrelated, domains of application. It appears, indeed, that many common mathematical objects are in fact of low rank.

A crucial recent development is the emergence of a new category of algorithms that allow one to detect low-rank properties and automatically construct the associated low rank tensor representations. They are collectively called tensor cross interpolation (TCI) algorithms [8–12], the subject of this article. Based on the cross interpolation (CI) decomposition of matrices instead of the singular value decomposition (SVD) widely used in standard tensor network techniques, TCI algorithms construct low-rank decompositions of a given tensor. Their main characteristic is that they do not take the entire tensor as input (in contrast to SVD-based decompositions) but request only a small number of tensor elements (the "pivots"). Their costs thus scale linearly with $\mathcal{L}$, even though the tensor has exponentially many elements. In this sense, TCI algorithms are akin to machine learning: they seek compact representations of a large dataset (the tensor) based on a small subset (the pivots). Moreover, they are *rank-revealing*: for low-rank tensors they rapidly find accurate low-rank decompositions (in most cases, see discussions below); for high-rank tensors they exhibit slow convergence rather than giving bad decompositions. TCI has been used recently, e.g., as an efficient (sign-problem-free) alternative to Monte Carlo sampling for calculating high-dimensional integrals arising in Feynman diagrams for the quantum many-body problem [13]; to find minima of functions [14]; to calculate topological invariants [15]; to calculate overlaps between atomic orbitals [16]; to solve the Schrödinger equation of the H_2^+ ion [16]; and, in mathematical finance, to speed up Fourier-transform-based option pricing [17].

Among the many applications of tensor networks, the so-called *quantics* [18–20] representation of functions of one or more variables has recently gained interest in various fields, including many-body field theory [21–25], turbulence [26–28], plasma physics [29], quantum chemistry [16], and denoising in quantum simulation [30]. Quantics tensor representations yield exponentially high resolution, and often have low-rank, even for functions exhibiting scale separation between large- and small-scale features. Such representation can be efficiently revealed using TCI [21]. Moreover, it can be exploited to perform many standard operation on functions (e.g. integration, multiplication, convolution, Fourier transform, ...) exponentially faster than when using naive brute-force discretizations. For example, quantics yields a compact basis for solving partial differential equations, similar to a basis of orthogonal (e.g. Chebyshev) polynomials.

This article has three main goals:

- We present new variants of TCI algorithms that are more robust and/or faster than previous ones. They are based on rank-revealing partial LU (prrLU) decomposition, which is equivalent to but more flexible and stable than traditional CI. The new variants offer useful new functionality beyond proposing new pivots, such as the ability to remove bad pivots, to add global pivots, to compress an existing MPS.
- We showcase various TCI applications (both with and without quantics), such as integrating multivariate functions, computing partition functions, integrating partial differential equations, constructing complex MPOs for many-body physics.
- We present the API of two open source libraries that implement TCI and quantics algorithms as well as related tools: `xfac`, written in C++ with python bindings; and `TensorCrossInterpolation.jl` (or `TCI.jl` for short), written in Julia.

Below, Sec. 2 very briefly describes and illustrates the capabilities of TCI, serving as a minimal primer for starting to use the libraries. Readers interested mainly in trying out TCI (or learning what it can do) may subsequently proceed directly to Secs. 5–7, which present several illustrative applications. Sec. 3 describes the formal relation between CI and prrLU at the matrix level, Sec. 4 presents our prrLU-based algorithms for tensors of higher degree. Finally Sec. 8 discusses the API of the `xfac` and `TCI.jl` libraries. Several appendices are devoted to technical details.

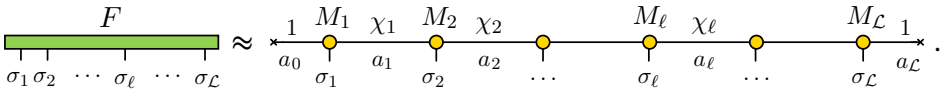
2 An introduction to tensor cross interpolation (TCI)

In this section, we present a quick primer on TCI algorithms without details, to set the scene for exploring our libraries and studying the examples in Section 5 and beyond.

2.1 The input and output of TCI

Consider a tensor F of degree $\mathcal{L}$, with elements $F_{\boldsymbol{\sigma}}$ labeled by indices $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_\mathcal{L})$, with $1 \leq \sigma_\ell \leq d_\ell$. For simplicity, we will denote the dimension $d = d_\ell$ if all the dimensions d_ℓ are equal. Our goal is to obtain an approximate factorization of F as a matrix product state (MPS), that we denote $\widetilde{F}_{\boldsymbol{\sigma}}$. An MPS has the following form and graphical representation:

$$F_{\boldsymbol{\sigma}} \approx \widetilde{F}_{\boldsymbol{\sigma}} = \prod_{\ell=1}^{\mathcal{L}} M_\ell^{\sigma_\ell} = [M_1]^{\sigma_1}_{1 a_1} [M_2]^{\sigma_2}_{a_1 a_2} \cdots [M_\mathcal{L}]^{\sigma_\mathcal{L}}_{a_{\mathcal{L}-1} 1}\,, \tag{1}$$



Implicit summation over repeated indices (Einstein convention) is understood and depicted graphically by connecting tensors by bonds. Each three-leg tensor M_ℓ has elements $[M_\ell]^{\sigma_\ell}_{a_{\ell-1} a_\ell}$, and can also be viewed as a matrix $M_\ell^{\sigma_\ell}$ with indices $a_{\ell-1}, a_\ell$. The *external indices* σ_ℓ have dimensions d_ℓ. The *internal (or bond) indices* a_ℓ have dimensions χ_ℓ, called the bond dimensions of the tensor. By convention, we use $\chi_0 = \chi_\mathcal{L} = 1$ to preserve a matrix product structure. We define $\chi \equiv \max_\ell \chi_\ell$ as the *rank* of the tensor.

The approximation (1) can be made arbitrarily accurate by increasing χ_ℓ, potentially exponentially with $\mathcal{L}$ like $\chi_\ell \sim \min\{d^\ell, d^{\mathcal{L}-\ell}\}$. A tensor is said to be *compressible* or *low-rank* if it can be approximated by a MPS form with a small rank χ.

TCI algorithms aim to construct low-rank MPS approximations (actually interpolations) for a given tensor F using a minimal number of its elements. They are high-dimensional generalizations of matrix decomposition methods, like the cross interpolation (CI) decomposition or the partially rank-revealing LU decomposition (prrLU) [31]. Indeed, they progressively refine the $\widetilde{F}$ approximation, increasing the ranks, by searching for *pivots* (high-dimensional generalizations of Gaussian elimination pivots), using CI or prrLU on two-dimensional slices of the tensor. TCI algorithms come with an error estimate $\epsilon(\chi_\ell)$, which can be reduced below a specified tolerance τ by suitably increasing χ_ℓ. Moreover, they are *rank-revealing*: if a given tensor F admits a low-rank MPS approximation, the algorithms will almost always find it; if the tensor is not of low rank (e.g. a tensor with random entries), the algorithms fail to converge and the computed error remains large.

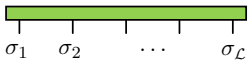
Concretely, TCI algorithms take as input a tensor F in the form of a function returning the value $F_{\boldsymbol{\sigma}}$ for any $\boldsymbol{\sigma}$; they explore its structure by sampling (in a deterministic way) some of its elements; and they return as output a list of tensors $M_1, \ldots, M_\mathcal{L}$ for the MPS approximation $\widetilde{F}$. Importantly, TCI algorithms do not require *all* $d^\mathcal{L}$ tensor elements of F but can construct $\widetilde{F}$ by calling $F_{\boldsymbol{\sigma}}$ only $\mathcal{O}(\mathcal{L} d \chi^2)$ times. The TCI algorithms have a time complexity $\mathcal{O}(\mathcal{L} d \chi^3)$ [12], that is exponentially smaller than the total number of elements. The TCI form is fully specified by $\mathcal{O}(\mathcal{L}\chi^2)$ pivot indices, which are sufficient to reconstruct the whole tensor at the specified tolerance. Furthermore, the TCI form allows an efficient evaluation of any tensor element.

Since TCI algorithms sample a given tensor F in a deterministic manner to construct a compressed representation $\widetilde{F}$, they can be viewed as machine learning algorithms. We will discuss the analogy with neural networks learning techniques in Section 4.8.

2.2 An illustrative application: Integration in large dimension

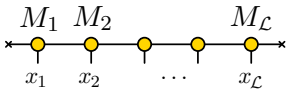
TCI algorithms allow new usages of the MPS tensor representation not contained in other tensor toolkits, for example integration or summation in large dimensions [8,12]. Consider a function $f(\mathbf{x})$, with $\mathbf{x} = (x_1, \ldots, x_\mathcal{L})$. We wish to calculate the $\mathcal{L}$-dimensional integral $\int d^\mathcal{L}\mathbf{x}\, f(\mathbf{x})$. We map f onto a tensor F by discretizing each variable x_ℓ onto a grid of d distinct points $\{p_1, p_2, \ldots, p_d\}$, e.g. the points of a Gauss quadrature or the Chebyshev points. Then, the *natural tensor representation* F of f on this grid is defined as

$$F_{\boldsymbol{\sigma}} = f(p_{\sigma_1}, p_{\sigma_2}, \ldots, p_{\sigma_\mathcal{L}}) = \quad , \tag{2}$$



with $\sigma_\ell = 1, \ldots, d$. This can be given as input to TCI. The resulting $\widetilde{F}$ yields a factorized approximation for f when all its arguments lie on the grid,

$$f(x_1, \ldots, x_\mathcal{L}) \approx M_1(x_1) M_2(x_2) \cdots M_\mathcal{L}(x_\mathcal{L}) = \quad , \tag{3}$$



for $x_\ell \in \{p_1, p_2, \ldots, p_d\}$, with $M_\ell(p_{\sigma_\ell}) \equiv M_\ell^{\sigma_\ell}$. The notation $M_\ell(x)$ reflects the fact that the approximation can be extended to the continuum, i.e. for all x (see the discussion in App. A.4, as well as Eqs. (7–9) of Ref. [13]). When $\widetilde{F}$ is low rank, f is *almost separable* (it would be separable if the rank $\chi = 1$). The integral of the factorized f is straightforward to compute as [8, 12, 13]

$$\int d^\mathcal{L}\mathbf{x} f(\mathbf{x}) \approx \int dx_1\, M_1(x_1) \int dx_2\, M_2(x_2) \cdots \int dx_\mathcal{L}\, M_\mathcal{L}(x_\mathcal{L}), \tag{4}$$

i.e. *one*-dimensional integrals followed by a sequence of matrix-vector multiplications. Since TCI algorithms can compute the compressed MPS form with a "small" number of evaluations of f (one for each requested tensor element), the integral computation is performed in $\mathcal{O}(\mathcal{L} d \chi^2) \ll \mathcal{O}(d^\mathcal{L})$ calls to the function $f(\mathbf{x})$. In practice, this method has been shown to be very successful, even when the function f is highly oscillatory. For example, it was recently shown to outperform traditional approaches for computing high-order perturbative expansions in the quantum many-body problem [13, 32]. Quite generally, TCI can be considered as a possible alternative to Monte Carlo sampling, particularly attractive if a sign problem (rapid oscillations of the integrand) makes Monte Carlo fail.

As an illustration, we compute a 10-dimensional integral with an oscillatory argument,

$$I = 10^3 \int_{[-1,+1]^{10}} \mathrm{d}^{10}\mathbf{x} \cos\Big(10 \textstyle\sum_{\ell=1}^{10} x_\ell^2\Big) \exp\Big[-10^{-3} \Big(\textstyle\sum_{\ell=1}^{10} x_\ell\Big)^4\Big], \tag{5}$$

using TCI with Gauss–Kronrod quadrature rules. As shown in Fig. 2, TCI converges approximately as $1/N_{\text{eval}}^4$, where N_{eval} is the number of evaluations of the integrand. For comparison, Monte Carlo integration would converge as $\mathcal{O}(1/\sqrt{N_{\text{eval}}})$ and encounter a sign problem due to the cosine term in the integrand.

In practice, our `xfac/TCI.jl` libraries take a user-defined, real- or complex-valued function $f(\mathbf{x})$ as input and construct a tensor train representation $\widetilde{F}_{\boldsymbol{\sigma}}$ with a user-specified tolerance τ or rank χ. Our TCI toolbox contains algorithms to decompose a tensor F or to recompress a given MPS decomposition. After a MPS form of F has been obtained, it can be used directly or transformed into one of several canonical forms (cf. Sec. 4.5) and used with other standard tensor toolkits such as ITensor [33]. In Sections 5 and beyond, we present various examples of applications. Readers interested mainly in these may prefer to the upcoming two Sections 3 and 4, which are devoted to the details of the algorithms.

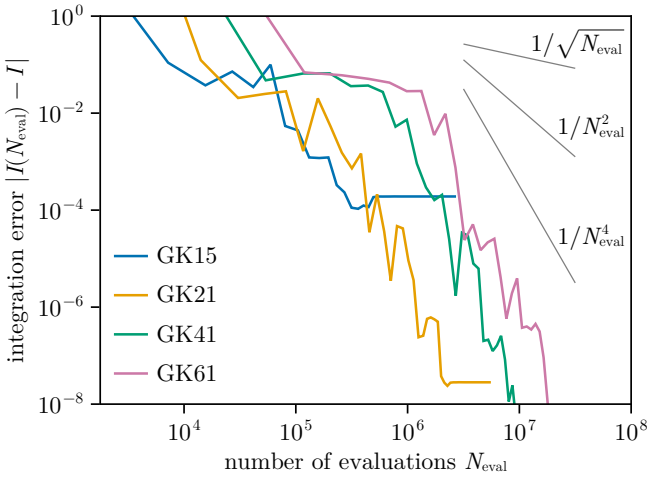


Figure 2: Convergence of the 10-dimensional integral I of Eq. (5). $I(N_{\text{eval}})$ is computed using TCI with 15, 21, 41 and 61-point Gauss–Kronrod quadrature in each dimension, and N_{eval} is the number of evaluations of the integrand. With 41- and 61-point quadrature, the value converges to $I = -5.4960415218049$. Convergence of the lower-order quadrature rules is limited by the number of discretization points.

3 Mathematical preliminaries: Low-rank decomposition of matrices from a few rows and columns

The original TCI algorithm [8–10] is based on the matrix cross interpolation (CI) formula, which constructs low rank approximations of matrices from crosses formed by subsets of their rows and columns. In this paper, we focus on a different but mathematically equivalent strategy for constructing cross interpolations, based on partial rank-revealing LU (prrLU) decompositions. This offers several advantages, in particular in term of stability.

A low-rank matrix is strongly *compressible*. Indeed, if $A = (\mathbf{a}_1, \ldots, \mathbf{a}_n)$ is an $m \times n$ matrix with column vectors $\mathbf{a}_j$ and (low) rank χ, each column can be expressed as a linear combination of a subset of χ of them ($\mathbf{a}_j = \sum_{i=1}^{\chi} \mathbf{b}_i C_{ij}$). Denoting the $m \times \chi$ submatrix $B = (\mathbf{b}_1, \ldots, \mathbf{b}_\chi)$, we have $A = BC$. It is sufficient to store B and C, i.e. $\chi(m+n)$ elements instead of mn, which is a large reduction when the rank is small ($\chi \ll \min(m,n)$).

The compressibility extends to matrices which are *approximately* of low rank. Using the SVD decomposition, a matrix A is rewritten as $A = UDV^\dagger$ with D a diagonal matrix of singular values, which can be truncated at some tolerance to yield a low-rank approximation $\widetilde{A}$ of A. While SVD is optimal (it minimizes the error $\|A - \widetilde{A}\|_F$ in the Frobenius norm), this comes at a cost: the *entire* matrix A is required for the decomposition. Here, we are interested in CI and prrLU, two low-rank approximations techniques which require only a subset of rows and columns of the matrix. Both are well-known and in fact intimately related [34].

This section is organized as follows: after recalling CI in Section 3.1, we review some standard material on Schur complements, prrLU and its relationship with CI. This section focuses exclusively on matrices; we generalize to tensors in the next section.

3.1 Matrix cross interpolation (CI)

Let us first recall the matrix cross interpolation (CI) formula [11, 35–42], cf. section III of Ref. [13] for an introduction.

Let A be a $m\times n$ matrix of rank χ. We write $\mathbb{I}=\{1,\dots,m\}$ and $\mathbb{J}=\{1,\dots,n\}$ for the ordered sets of all row or column indices, respectively, and $\mathcal{I}=\{i_1,\dots,i_{\widetilde{\chi}}\}\subset\mathbb{I}$ and $\mathcal{J}=\{j_1,\dots,j_{\widetilde{\chi}}\}\subset\mathbb{J}$ for subsets of $\widetilde{\chi}$ row and column indices. Following a standard MATLAB convention, we write $A(\mathcal{I},\mathcal{J})$ for the submatrix or *slice* containing all intersections of $\mathcal{I}$-rows and $\mathcal{J}$-columns (i.e. rows and columns labeled by indices in $\mathcal{I}$ and $\mathcal{J}$, respectively), with elements

$$[A(\mathcal{I},\mathcal{J})]_{\alpha\beta}\equiv A_{i_\alpha,j_\beta}\,, \tag{6}$$

$\forall\alpha,\beta\in\{1,\dots,\widetilde{\chi}\}$. In particular, $A(\mathbb{I},\mathbb{J})=A$. In the following, we assume $\widetilde{\chi}\leq\chi$, with $\mathcal{I}$ and $\mathcal{J}$ chosen such that the matrix $A(\mathcal{I},\mathcal{J})$ is non-singular. We define the following slices of A:

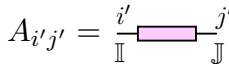
$$P=A(\mathcal{I},\mathcal{J})\,,\qquad C=A(\mathbb{I},\mathcal{J})\,,\qquad R=A(\mathcal{I},\mathbb{J})\,. \tag{7}$$

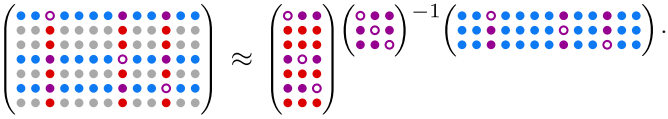
$P=A(\mathcal{I},\mathcal{J})$ is the *pivot matrix*. Its elements are called *pivots*, labeled by index pairs $(i,j)\in\mathcal{I}\times\mathcal{J}$. These index pairs are called pivots, too (a common abuse of terminology), and the index sets $\mathcal{I}$, $\mathcal{J}$ specifying them are called *pivot lists*. In other words, the slice $C=A(\mathbb{I},\mathcal{J})$ gathers all columns containing pivots, the slice $R=A(\mathcal{I},\mathbb{J})$ gathers all rows containing pivots, and P contains their intersections (thus it is a subslice of both).

The *CI formula* gives a rank-$\widetilde{\chi}$ approximation $\widetilde{A}$ of A [38] that can be expressed in the following equivalent forms:

$$A\approx CP^{-1}R=\widetilde{A}, \tag{8}$$

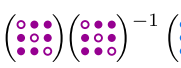
$$A(\mathbb{I},\mathbb{J})\approx A(\mathbb{I},\mathcal{J})\,P^{-1}A(\mathcal{I},\mathbb{J})=\widetilde{A}(\mathbb{I},\mathbb{J})\,, \tag{9}$$

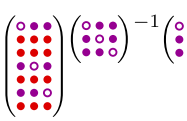
$A_{i'j'}=$  ,



The third line depicts this factorization diagrammatically through the *insertion of two pivot bonds*. There, the external indices $i'\in\mathbb{I}$ and $j'\in\mathbb{J}$ are fixed, ◆ represents P^{-1}, and the two internal bonds represent sums $\sum_{j\in\mathcal{J}}\sum_{i\in\mathcal{I}}$ over the pivot lists $\mathcal{I}$, $\mathcal{J}$. The fourth line visualizes this for $\widetilde{\chi}=3$, with $\mathcal{J}$-columns colored red, $\mathcal{I}$-rows blue, and pivots purple.

The CI formula (9) has two important properties: (i) For $\widetilde{\chi}=\chi$, Eq. (9) exactly reproduces the entire matrix, $\widetilde{A}=A$ (as explained below). (ii) For any $\widetilde{\chi}\leq\chi$ it yields an interpolation, i.e. it exactly reproduces all $\mathcal{I}$-rows and $\mathcal{J}$-columns of A. Indeed, when considering only the $\mathcal{I}$-rows or $\mathcal{J}$-columns of $\widetilde{A}(\mathbb{I},\mathbb{J})$ in Eq. (9), we obtain

 : $\widetilde{A}(\mathcal{I},\mathbb{J})=A(\mathcal{I},\mathbb{J})\,,\quad$ since $\quad A(\mathcal{I},\mathcal{J})P^{-1}=\mathbb{1}\,,$ (10a)

 : $\widetilde{A}(\mathbb{I},\mathcal{J})=A(\mathbb{I},\mathcal{J})\,,\quad$ since $\quad P^{-1}A(\mathcal{I},\mathcal{J})=\mathbb{1}\,,$ (10b)

where $\mathbb{1}$ denotes a $\widetilde{\chi}\times\widetilde{\chi}$ unit matrix.

The accuracy of a CI interpolation depends on the choice of pivots. Efficient heuristic strategies for finding good pivots are thus of key importance. They will be discussed in Sec. 3.3.2.

3.2 A few properties of Schur complements

This section discusses an important object of linear algebra, the Schur complement. Of primary importance to us are two facts that allow us to make the connection between CI and prrLU.

First, the Schur complement is essentially the *error* of the CI approximation. Second, the Schur complement can be obtained *iteratively* by eliminating (in the sense of Gaussian elimination) rows and columns of the initial matrix one after the other and in any order. With these two properties, we will be able to prove that the prrLU algorithm discussed in the next section actually yields a CI approximation.

3.2.1 Definitions and basic properties

Let us consider a matrix A made of 4 blocks

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \tag{11}$$

with A_{11} assumed square and invertible. The Schur complement $[A/A_{11}]$ is defined by

$$[A/A_{11}] \equiv A_{22} - A_{21}(A_{11})^{-1}A_{12}\,. \tag{12}$$

The matrix A can be factorized as

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} \mathbb{1}_{11} & 0 \\ A_{21}A_{11}^{-1} & \mathbb{1}_{22} \end{pmatrix} \begin{pmatrix} A_{11} & 0 \\ 0 & [A/A_{11}] \end{pmatrix} \begin{pmatrix} \mathbb{1}_{11} & A_{11}^{-1}A_{12} \\ 0 & \mathbb{1}_{22} \end{pmatrix}. \tag{13}$$

This leads to the Schur determinant identity

$$\det A = \det A_{11} \det[A/A_{11}], \tag{14}$$

and (by inverting (13), see also Appendix A.1) to the relation

$$\left(A^{-1}\right)_{22} = [A/A_{11}]^{-1}\,. \tag{15}$$

3.2.2 The quotient property

When used for successively eliminating blocks, the Schur complement does not depend on the order in which the different blocks are eliminated. This is expressed by *the quotient property of the Schur complement* [43]. We illustrate this property on a 3×3 block matrix,

$$A = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix}, \qquad B \equiv \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \tag{16}$$

where B is a submatrix of A. We assume that A_{11} and A_{22} are square and invertible. Then the quotient formula reads

$$\big[[A/A_{11}]/[B/A_{11}]\big] = \big[A/B\big] = \big[[A/A_{22}]/[B/A_{22}]\big]. \tag{17}$$

A simple explicit proof of this property is provided in Appendix A.1, see also [44].

As the order of block elimination does not matter, we will use a simpler notation

$$\big[[A/1]/2\big] = \big[[A/2]/1\big] = \big[A/(1,2)\big], \tag{18}$$

where $/1$ or $/2$ denotes the elimination of the 11- or 22 block, and $/(1,2)$ the elimination of the square matrix containing both. Let us also note that permutations of rows and columns in the 11- and 22-blocks can be taken before or after taking the Schur complement $[A/(1,2)]$ without affecting the result [44]. For matrices involving a larger number of blocks, iterative application of the Schur quotient rule to successively eliminate blocks 11 to xx reads

$$\Big[\big[\big[[A/1]/2\big]\ldots\big]/x\Big] = \big[A/(1,2,\ldots,x)\big]. \tag{19}$$

3.2.3 Relation with CI

The error in the matrix cross interpolation formula is directly given by the Schur complement to the pivot matrix.

To see this, let us permute the rows and columns of A such that all pivots lie in the first $\widetilde{\chi}$ rows and columns, labeled $\mathcal{I}_1 = \mathcal{J}_1 = \{1,\dots,\widetilde{\chi}\}$, with $\mathcal{I}_2 = \mathbb{I}\setminus\mathcal{I}_1$ and $\mathcal{J}_2 = \mathbb{J}\setminus\mathcal{J}_1$ labeling the remaining rows and columns, respectively. Then, the permuted matrix (again denoted A for simplicity) has the block form

$$A(\mathbb{I},\mathbb{J}) = \begin{pmatrix} A(\mathcal{I}_1,\mathcal{J}_1) & A(\mathcal{I}_1,\mathcal{J}_2) \\ A(\mathcal{I}_2,\mathcal{J}_1) & A(\mathcal{I}_2,\mathcal{J}_2) \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \tag{20}$$

and the pivot matrix is $P = A_{11} = A(\mathcal{I}_1,\mathcal{J}_1)$. The CI formula (9) now takes the form

$$\widetilde{A} = \begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} (A_{11})^{-1} \begin{pmatrix} A_{11} & A_{12} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{21}(A_{11})^{-1}A_{12} \end{pmatrix}, \tag{21}$$

$$A - \widetilde{A} = \begin{pmatrix} 0 & 0 \\ 0 & [A/A_{11}] \end{pmatrix}. \tag{22}$$

The interpolation is exact for the 11-, 21- and 12-blocks, but not for the 22-block where the error is the Schur complement $[A/A_{11}]$. Since the latter depends on the inverse of the pivot matrix, a strategy for reducing the error is to choose the pivots such that $|\det A_{11}|$ is maximal —a criterion known as the *maximum volume principle* [35, 41]. Finding the pivots that satisfy the maximum volume principle is in general exponentially difficult but, as we shall see, there exist good heuristics that get close to this optimum in practice.

3.2.4 Relation with self-energy

In physics context, the Schur complement is closely related to the notion of self-energy, which appears in a non-interacting model by integrating out some degrees of freedom. Consider a Hamiltonian matrix

$$H = H_0 + V = \begin{pmatrix} H_{11} & 0 \\ 0 & H_{22} \end{pmatrix} + \begin{pmatrix} 0 & H_{12} \\ H_{21} & 0 \end{pmatrix}. \tag{23}$$

The Green's function at energy E is defined as $G(E) = (E-H)^{-1}$. Its restriction to the 22-block is given by the Dyson equation,

$$[G(E)]_{22} = (E - H_{22} - \Sigma)^{-1}, \tag{24}$$

where $\Sigma = H_{21}(E-H_{11})^{-1}H_{12}$ is the so-called self-energy. The Dyson equation can be proven by applying Eq. (15) to $[G(E)]_{22} = [(E-H)^{-1}]_{22}$ and inserting the definition of the Schur complement, Eq. (12):

$$\begin{aligned} [G(E)]_{22} &= [(E-H)^{-1}]_{22} = [(E-H)/(E-H)_{11}]^{-1} \\ &= \big[(E-H)_{22} - \underbrace{H_{21}[(E-H)_{11}]^{-1}H_{12}}_{\Sigma}\big]^{-1}. \end{aligned} \tag{25}$$

3.2.5 Restriction of the Schur complement

A trivial, yet important, property of the Schur complement is that the restriction of the Schur complement to a limited numbers of rows and columns is equal to the Schur complement of the full matrix restricted to those rows and columns (plus the pivots). More precisely, if $\mathcal{I}_1$

and $\mathcal{J}_1$ are the lists of pivots specifying the Schur complement and $\mathcal{I}_2$ and $\mathcal{J}_2$ are lists of rows and columns of interest, one has

$$[A(\mathcal{I},\mathcal{J})/A(\mathcal{I}_1,\mathcal{J}_1)](\mathcal{I}_2,\mathcal{J}_2) = [A(\mathcal{I}_1\cup\mathcal{I}_2,\mathcal{J}_1\cup\mathcal{J}_2)/A(\mathcal{I}_1,\mathcal{J}_1)], \tag{26}$$

where $\mathcal{I}_1,\mathcal{I}_2\subseteq\mathcal{I}$ and $\mathcal{J}_1,\mathcal{J}_2\subseteq\mathcal{J}$. This property follows directly from the definition of the Schur complement.

3.3 Partial rank-revealing LU decomposition

In this section, we discuss partial rank-revealing LU (prrLU) decomposition. While mathematically equivalent to the CI decomposition, it is numerically more stable as the pivot matrices are never constructed nor inverted explicitly.

A matrix decomposition is *rank-revealing* when it allows the determination of the rank of the matrix: the decomposition $A = XDY$ is rank-revealing if both X and Y are well-conditioned and D is diagonal. The rank is given by the number of non-zero entries on the diagonal of D. A well-known rank-revealing decomposition is SVD.

3.3.1 Default full search prrLU algorithm

The standard LU decomposition factorizes a matrix as $A = LDU$, where L is lower-triangular, D diagonal and U upper-triangular [31]. It implements the Gaussian elimination algorithm for inverting matrices or solving linear systems of equations. The prrLU decomposition is an LU variant with two particular features: (i) It is *rank-revealing*: the largest remaining element, found by pivoting on both rows and columns, is used for the next pivot. (ii) It is *partial*: Gaussian elimination is stopped after constructing the first $\widetilde{\chi}$ columns of L and rows of U, such that LDU is a rank-$\widetilde{\chi}$ factorization of A.

The prrLU decomposition is computed using a fully-pivoted Gaussian elimination scheme, based on Eq. (13), which we reproduce here for convenience.

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} \mathbb{1}_{11} & 0 \\ A_{21}A_{11}^{-1} & \mathbb{1}_{22} \end{pmatrix} \begin{pmatrix} A_{11} & 0 \\ 0 & [A/A_{11}] \end{pmatrix} \begin{pmatrix} \mathbb{1}_{11} & A_{11}^{-1}A_{12} \\ 0 & \mathbb{1}_{22} \end{pmatrix}. \tag{27}$$

Note that the right side has a block LDU structure. The algorithm utilizes this as follows. First, we permute the rows and columns of A such that its largest element (in modulus) is positioned into the top left 11-position, then apply the above identity with a 11-block of size 1×1. Next, we repeat this procedure on the lower-right block of the second matrix on the right of Eq. (27) (hereafter, the "central" matrix), i.e. on $[A/1]$. We continue iteratively, yielding $[A/(1,2)]$, $[A/(1,2,3)]$, etc., thereby progressively diagonalizing the central matrix while maintaining the lower- and upper-triangular form of L and U. Before each application of Eq. (27) we choose the largest element of the previous Schur complement as new pivot and permute it to the top left position of that submatrix. This strategy of maximizing the pivot improves the algorithm's stability, since it minimizes the inverse of the new pivot, which enters the left and right matrices [35,41] and corresponds to the maximum volume strategy over the new pivot, see Appendix B2 of [13]. After $\widetilde{\chi}$ steps we obtain a prrLU decomposition of the form

$$A = \begin{pmatrix} L_{11} & 0 \\ L_{21} & \mathbb{1}_{22} \end{pmatrix} \begin{pmatrix} D & 0 \\ 0 & [A/(1,\ldots,\widetilde{\chi})] \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ 0 & \mathbb{1}_{22} \end{pmatrix}. \tag{28}$$

Here, L_{11} and U_{11} have diagonal entries equal to 1 and are lower- or upper-triangular, respectively, and D (shorthand for D_{11}) is diagonal [31,42]. The block subscripts 11, 12, 21, 22 label blocks with row and column indices given by $\mathcal{I}_1 = \mathcal{J}_1 = \{1,\ldots,\widetilde{\chi}\}$, $\mathcal{I}_2 = \mathbb{I}\setminus\mathcal{I}_1$, and $\mathcal{J}_2 = \mathbb{J}\setminus\mathcal{J}_1$, where these indices refer to the *pivoted* version of the original A. When the Schur

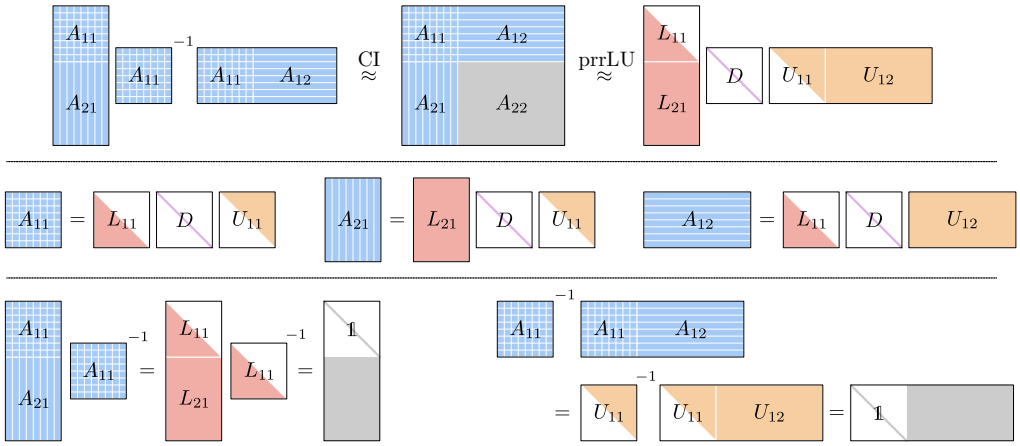


Figure 3: Equivalence between CI and prrLU. The prrLU decomposition provides all the matrices of the CI. Top: Eqs. (22) and (30); middle: Eqs. (32a-32c); bottom: Eqs. (32d-32e). White portions of matrices are equal to 0.

complement becomes zero, after χ steps, the scheme terminates, identifying χ as the rank of A.

Now, note that (for any $\widetilde{\chi} \leq \chi$) Eq. (28) can be recast into the form

$$A = LDU + \begin{pmatrix} 0 & 0 \\ 0 & [A/(1,\dots,\widetilde{\chi})] \end{pmatrix}, \qquad L = \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix}, \qquad U = \begin{pmatrix} U_{11} & U_{12} \end{pmatrix}. \tag{29}$$

This precisely matches the CI formula (22). Again the Schur complement $[A/(1,\dots,\widetilde{\chi})]$ is the error in the factorization. Thus, prrLU actually yields an CI [34, 42], given by

$$\widetilde{A} = LDU = \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} D \begin{pmatrix} U_{11} & U_{12} \end{pmatrix}. \tag{30}$$

Explicit relations between the CI and prrLU representations are obtained from Eq. (30):

$$\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} (A_{11})^{-1} \begin{pmatrix} A_{11} & A_{12} \end{pmatrix} = \begin{pmatrix} L_{11}DU_{11} \\ L_{21}DU_{11} \end{pmatrix} (L_{11}DU_{11})^{-1} \begin{pmatrix} L_{11}DU_{11} & L_{11}DU_{12} \end{pmatrix}, \tag{31}$$

where, abusing notation, $A_{xy} = A(\mathcal{I}_x, \mathcal{J}_y)$ now denote blocks of the pivoted version of the original A. This yields the following identifications, depicted schematically in Fig. 3:

$$A_{11} = P = L_{11}DU_{11}, \tag{32a}$$
$$A_{21} = L_{21}DU_{11}, \tag{32b}$$
$$A_{12} = L_{11}DU_{12}, \tag{32c}$$
$$\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} (A_{11})^{-1} = \begin{pmatrix} \mathbb{1}_{11} \\ L_{21}L_{11}^{-1} \end{pmatrix}, \tag{32d}$$
$$(A_{11})^{-1} \begin{pmatrix} A_{11} & A_{12} \end{pmatrix} = \begin{pmatrix} \mathbb{1}_{11} & U_{11}^{-1}U_{12} \end{pmatrix}. \tag{32e}$$

The main advantage of prrLU over a direct CI is numerical stability, as we avoid the construction and inversion of ill-conditioned pivot matrices [31]. In our experience, prrLU is also more stable than the QR-stabilization approach to CI used in [13]. Furthermore, prrLU is updatable: new rows and columns can be added easily.

Let us note that the maximal pivot strategy of prrLU eliminates the largest contribution to the next Schur complement, hence reducing the CI error. Hence, it is a simple, greedy algorithm for constructing a near-maximum volume submatrix [42, 45].

3.3.2 Alternative pivot search methods: Full, rook or block rook

The above algorithm uses a *full search* for the pivots, i.e. it uses the information of the entire matrix A and scales as $O(mn)$. It provides a quasi-optimal CI approximation but is expensive computationally as each new pivot is searched on the entire Schur complement $[A/(1,\dots,\widetilde{\chi})]$.

Rook search is a cheaper alternative, first proposed in [46, 47]. (See Algorithm 2 of [12] and Ref. [13, Sec. III.B.3], where it was called *alternating* search). It explores the Schur complement $[A/(1,\dots,\widetilde{\chi})]$ by moving in alternating fashion along its rows and columns, similar to a chess rook. It searches along a randomly chosen initial column for the row yielding the maximum error, along that row for the column yielding the maximal error, and so on. The process terminates when a "rook condition is established", i.e. when an element is found that maximizes the error along both its row and column; that element is selected as new pivot. Compared to full pivoting, rook pivoting has the following useful properties: (i) computational cost reduced to $O[\max(m,n)]$ from $O(mn)$; (ii) comparable robustness [48]; (iii) almost as good convergence of the CI in practice.

Algorithm 1: Block rook pivoting search. Given pivot lists $\mathcal{I}$, $\mathcal{J}$, the algorithm updates the lists $\mathcal{I}$, $\mathcal{J}$ in place by alternating between searching for better pivots along the rows and columns in even or odd iterations, respectively. In each iteration, the pivot lists $\mathcal{I}$, $\mathcal{J}$ are updated with new, improved pivots (the 'rook move') from a prrLU decomposition with tolerance ϵ (line 8). The algorithm terminates when either the rook condition is met, i.e. when there are no better pivots along the available rows and columns, or when a maximum depth of n_{rook} iterations has been reached (typically $n_{\text{rook}} \leq 5$). Upon exiting the algorithm, the updated lists $\mathcal{I}$ and $\mathcal{J}$ are of equal size.

Input: A matrix function A with row indices $\mathbb{I}$ and column indices $\mathbb{J}$, initial pivot lists $\mathcal{I} \subseteq \mathbb{I}, \mathcal{J} \subseteq \mathbb{J}$ with χ elements each, and tolerance τ.

Output: Updated pivot lists $\mathcal{I}, \mathcal{J}$ for the prrLU of A with up to 2χ elements each.

1 $\mathcal{J}' \leftarrow \mathcal{J} \cup \{\chi$ new random column indices $\in \mathbb{J} \setminus \mathcal{J}\}$
2 **for** $t \leftarrow 1$ **to** n_{rook} **do**
3 **if** t *is odd* **then**
4 search among the columns: set $B \leftarrow A(\mathbb{I}, \mathcal{J}')$
5 **else**
6 search among the rows: set $B \leftarrow A(\mathcal{I}', \mathbb{J})$
7 **end**
8 find new pivots: $(\mathcal{I}', \mathcal{J}') \leftarrow$ pivots of $\text{prrLU}_\epsilon(B)$
9 **if** $\mathcal{I}' = \mathcal{I}$ *and* $\mathcal{J}' = \mathcal{J}$ **then** rook condition has been established.
10 **return** $\mathcal{I}, \mathcal{J}$
11 **else**
12 update the pivots: $(\mathcal{I}, \mathcal{J}) \leftarrow (\mathcal{I}', \mathcal{J}')$
13 **end**
14 **end**

We now introduce *block rook search*. It is a variant of rook search which searches for all pivots simultaneously. It is useful in the common situation that a CI of a matrix $A(\mathbb{I}, \mathbb{J})$ has been obtained and then this matrix is extended to a larger matrix $A(\mathbb{I}', \mathbb{J}')$ by adding some new rows and columns. One needs to construct a new set of pivots $\mathcal{I}'$ and $\mathcal{J}'$. The previous set of pivots $\mathcal{I}$ and $\mathcal{J}$ is a very good starting point that one wishes to leverage on to construct this new set. Block rook search is described in Algorithm 1.

To find pivots, the algorithm uses a series of prrLU, applied to a subset of rows and columns in alternating fashion. It starts with a set of columns made of previously found pivots and some random ones. It then LU factorizes the corresponding sub-matrix to yield new pivot rows and columns. The algorithm is repeated, alternatingly on rows and columns, until convergence (or up to n_{Rook} times). In practice, we observe that $n_{\text{Rook}} = 3$ is often sufficient to reach convergence. At convergence, the pivots satisfy rook conditions as if they had been sequentially found by rook search (see App. A.2 for a proof). The algorithm requires $\mathcal{O}(n_{\text{Rook}}\chi^2 \max(m,n))$ to factorize the matrix A.

4 Tensor cross interpolation

We now turn to the tensor case. After introducing the TCI form of an MPS, we present the TCI algorithm and its variants. Although this section is self-contained, it is somewhat compact and we recommend users new to TCI to read a more pedagogical introduction first, such as section III of [13]. Important proofs can also be found in the appendices of [13] and/or in the mathematical literature [8–12, 18, 19, 49, 50].

The algorithm used by some of us previously (e.g. in [13, 15, 16]) will be referred to as the 2-site TCI algorithm in *accumulative mode*. Below, we introduce a number of new algorithms that evolved from this original one. Our default TCI (discussed first, in section 4.3.1) is the 2-site TCI algorithm in *reset* mode. We also introduce a 1-site TCI, a 0-site TCI and a CI-canonical algorithm and explain their specific use cases.

4.1 TCI form of tensor trains

Tensor trains obtained from TCI decompositions of an input tensor $F_{\boldsymbol{\sigma}}$ have a very particular, characteristic form, called *TCI form*. It is obtained, e.g., through repeated use of the CI approximation, as discussed informally in Sec. III.B.1 of [13]. Its defining characteristic is that it is built *only* from one-dimensional slices of $F_{\boldsymbol{\sigma}}$ (on which all tensor indices σ_ℓ but one are fixed). Furthermore, TCI algorithms construct the TCI form using only *local* updates of these slices, as discussed in later sections.

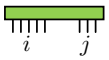
The most difficult part of implementing TCI algorithms lies in the book-keeping of various lists of indices. This is facilitated by the introduction of the following notations.

- An external index σ_ℓ ($\ell \in \{1, 2, \ldots, \mathcal{L}\}$) takes d_ℓ different values from a set $\mathbb{S}_\ell$.
- $\mathbb{I}_\ell = \mathbb{S}_1 \times \cdots \times \mathbb{S}_\ell$ denotes the set of *row multi-indices* up to site ℓ. An element $i \in \mathbb{I}_\ell$ is a row multi-index taking the form $i = (\sigma_1, \ldots, \sigma_\ell)$.
- $\mathbb{J}_\ell = \mathbb{S}_\ell \times \cdots \times \mathbb{S}_\mathcal{L}$ denotes the set of *column multi-indices* from site ℓ upwards. An element $j \in \mathbb{J}_\ell$ is a column multi-index taking the form $j = (\sigma_\ell, \ldots, \sigma_\mathcal{L})$.
- $\mathbb{I}_\mathcal{L} = \mathbb{J}_1$ is the full configuration space. A full configuration $\boldsymbol{\sigma} \in \mathbb{I}_\mathcal{L}$ takes the form $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_\mathcal{L})$.
- $i_\ell \oplus j_{\ell+1} \equiv (\sigma_1, \ldots, \sigma_\mathcal{L})$ denotes the concatenation of complementary multi-indices.

For each ℓ, we define a list of "pivot rows" $\mathcal{I}_\ell \subseteq \mathbb{I}_\ell$ and a list of "pivot columns" $\mathcal{J}_\ell \subseteq \mathbb{J}_\ell$. We also define $\mathcal{I}_0 = \mathcal{J}_{\mathcal{L}+1} = \{()\}$, where $()$ is an empty tuple. Note that $\mathcal{I}_\ell$ and $\mathcal{J}_\ell$ are lists of lists of external σ indices. Through the pivot rows and pivot columns, we define zero-, one-, and two-dimensional slices of the tensor F, where a k-dimensional slice has k free indices, as follows.

- A pivot matrix P_ℓ is a zero-dimensional slice of the input tensor F:

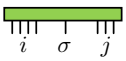
$$[P_\ell]_{ij} = F_{i\oplus j} = \tag{33a}$$



  for $i \in \mathcal{I}_\ell$ and $j \in \mathcal{J}_{\ell+1}$, or in Matlab notation, $P_\ell = F(\mathcal{I}_\ell, \mathcal{J}_{\ell+1})$. The two pivot lists have the same number of elements; P_ℓ is a square matrix of dimension $\chi_\ell = |\mathcal{I}_\ell| = |\mathcal{J}_{\ell+1}|$ and we will choose the pivots such that $\det P_\ell \neq 0$.

- A 3-leg T-tensor T_ℓ is a one-dimensional slice of F:

$$[T_\ell]_{i\sigma j} \equiv F_{i\oplus(\sigma)\oplus j} = \tag{33b}$$



  for $i \in \mathcal{I}_{\ell-1}$, $\sigma \in \mathbb{S}_\ell$ and $j \in \mathcal{J}_{\ell+1}$, or $T_\ell \equiv F(\mathcal{I}_{\ell-1}, \mathbb{S}_\ell, \mathcal{J}_{\ell+1})$. For specified σ, the matrix T_ℓ^σ is defined as $[T_\ell^\sigma]_{ij} \equiv [T_\ell]_{i\sigma j}$.

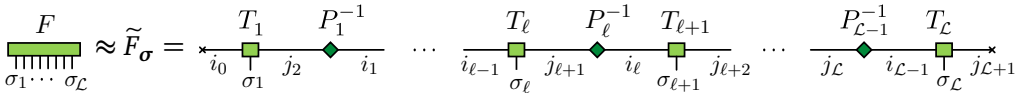
- A 4-leg Π-tensor Π_ℓ is a two-dimensional slice of F:

$$[\Pi_\ell]_{i\sigma\sigma' j} \equiv F_{i\oplus(\sigma,\sigma')\oplus j} = \tag{33c}$$

  for $i \in \mathcal{I}_{\ell-1}$, $\sigma \in \mathbb{S}_\ell$, $\sigma' \in \mathbb{S}_{\ell+1}$ and $j \in \mathcal{J}_{\ell+2}$, or $\Pi_\ell \equiv F(\mathcal{I}_{\ell-1}, \mathbb{S}_\ell, \mathbb{S}_{\ell+1}, \mathcal{J}_{\ell+2})$.

With these definitions, the TCI approximation $\widetilde{F}$ of F is defined as

$$F_{\boldsymbol{\sigma}} \approx \widetilde{F}_{\boldsymbol{\sigma}} = T_1^{\sigma_1} P_1^{-1} \cdots T_\ell^{\sigma_\ell} P_\ell^{-1} T_{\ell+1}^{\sigma_{\ell+1}} \cdots P_{\mathcal{L}-1}^{-1} T_{\mathcal{L}}^{\sigma_{\mathcal{L}}}, \tag{34}$$



with independent summations over all $i_\ell \in \mathcal{I}_\ell$ and all $j_{\ell+1} \in \mathcal{J}_{\ell+1}$, for $\ell = 1, \ldots, \mathcal{L}-1$. Here, ◆ represents P_ℓ^{-1}, the inverse of a pivot matrix, and ■ represents a T-tensor T_ℓ. Such a tensor cross interpolation is entirely defined by the T and P tensors, i.e. by slices of F. In other words, if one (i) knows the pivot lists $\{\mathcal{I}_\ell, \mathcal{J}_{\ell+1} | \ell = 1, \ldots, \mathcal{L}-1\}$ and (ii) can compute $F_{\boldsymbol{\sigma}}$ for any given $\boldsymbol{\sigma}$, then one can construct $\widetilde{F}$. Equation (34) defines a genuine tensor train with rank $\chi = \max \chi_\ell$. Its form matches Eq. (1) with the identification $T_\ell P_\ell^{-1} = M_\ell$.

Equation (34) defines the *TCI form*, which is fully specified by two ingredients: (i) the sets of rows $\mathcal{I}_\ell$ and columns $\mathcal{J}_\ell$, and (ii) the corresponding values (slices) T_ℓ and P_ℓ of the input tensor $F_{\boldsymbol{\sigma}}$. Any tensor train can be converted exactly to a TCI form (see Sec. 4.5.1).

4.2 Nesting conditions

TCI algorithm relies on an important property of the pivot lists $\mathcal{I}_\ell$ and $\mathcal{J}_\ell$ that we now discuss, the *nesting conditions*. By definition, for any ℓ:

- $\mathcal{I}_\ell$ is nested with respect to $\mathcal{I}_{\ell-1}$, denoted by $\mathcal{I}_{\ell-1} < \mathcal{I}_\ell$, if $\mathcal{I}_\ell \subseteq \mathcal{I}_{\ell-1} \times \mathbb{S}_\ell$, or equivalently, if removing the last index of any element of $\mathcal{I}_\ell$ yields an element of $\mathcal{I}_{\ell-1}$. $\mathcal{I}_{\ell-1} < \mathcal{I}_\ell$ implies that the pivot matrix P_ℓ is a slice of T_ℓ.

- $\mathcal{J}_\ell$ is nested with respect to $\mathcal{J}_{\ell+1}$, denoted by $\mathcal{J}_\ell > \mathcal{J}_{\ell+1}$, if $\mathcal{J}_\ell \subseteq \mathbb{S}_\ell \times \mathcal{J}_{\ell+1}$, or equivalently, if removing the first index of any element of $\mathcal{J}_\ell$ yields an element of $\mathcal{J}_{\ell+1}$. $\mathcal{J}_\ell > \mathcal{J}_{\ell+1}$ implies that the pivot matrix $P_{\ell-1}$ is a slice of T_ℓ.

Table 1: Example for a fully nested configuration of the pivot lists $\mathcal{I}_\ell$ and $\mathcal{J}_\ell$ for a TCI with 5 local indices $\sigma_1, \ldots, \sigma_5 \in \{0, 1\}$. Pivot lists that belong to the same bond are shown in the same row.

ℓ	$\mathcal{I}_\ell$	$\mathcal{J}_{\ell+1}$
1	$\mathcal{I}_1 = ((1))$	$\mathcal{J}_2 = ((1,0,0,1))$
2	$\mathcal{I}_2 = ((1,0),(1,1))$	$\mathcal{J}_3 = ((0,0,1),(1,0,1))$
3	$\mathcal{I}_3 = ((1,1,0),(1,0,1))$	$\mathcal{J}_4 = ((0,1),(1,1))$
4	$\mathcal{I}_4 = ((1,1,0,0))$	$\mathcal{J}_5 = ((1))$

We say that the pivots are:

- *left-nested* up to ℓ if
$$\mathcal{I}_0 < \mathcal{I}_1 < \cdots < \mathcal{I}_\ell\,, \tag{35}$$
- *right-nested* up to ℓ if
$$\mathcal{J}_\ell > \mathcal{J}_{\ell+1} > \cdots > \mathcal{J}_{\mathcal{L}+1}\,, \tag{36}$$
- *fully left-nested* if they are left-nested up to $\mathcal{L}-1$, *fully right-nested* if they are right-nested up to 2. When the pivots are both fully left- *and* right-nested they are said to be *fully nested*, i.e. one has
$$\mathcal{I}_0 < \mathcal{I}_1 < \cdots < \mathcal{I}_{\mathcal{L}-1}\,, \qquad \mathcal{J}_2 > \mathcal{J}_{\ell+2} > \cdots > \mathcal{J}_{\mathcal{L}+1}\,. \tag{37}$$

The importance of nesting conditions stems from the fact that they provides some interpolation properties. We refer to Ref. [13] or Appendix A.3 for the associated proofs. In particular, if the pivots are left-nested up to $\ell - 1$ and right-nested up to $\ell + 1$ (we say *nested w.r.t.* T_ℓ) then the TCI form is exact on the one-dimensional slice T_ℓ:

$$\widetilde{F}_{i\oplus(\sigma)\oplus j} = [T_\ell]_{i\sigma j} = F_{i\oplus(\sigma)\oplus j}\,, \quad \forall i \in \mathcal{I}_{\ell-1}\,,\ \sigma \in \mathbb{S}_\ell\,,\ j \in \mathcal{J}_{\ell+1}\,. \tag{38}$$

It follows that if the pivots are fully nested, then the TCI form is exact on every T_ℓ and P_ℓ, i.e. on all slices used to construct it. Hence, it is an interpolation.

An example for a fully nested configuration of the pivot lists $\mathcal{I}_\ell$ and $\mathcal{J}_\ell$ for a TCI with 5 local indices $\sigma_1, \ldots, \sigma_5 \in \{0, 1\}$ is shown in Table 1. Full nesting could be broken for example by adding $(0, 0)$ to $\mathcal{I}_2$, or by adding $(1, 1, 0)$ to $\mathcal{J}_3$.

4.3 2-site TCI algorithms

The goal of TCI algorithms is to obtain a TCI approximation of a given tensor F at a specified tolerance $\|F - \widetilde{F}\|_\infty < \tau$ (over the maximum norm), by finding a minimal set of suitable pivots. In this section, we present various 2-site TCI algorithms and discuss their variants and options. They are all based on the fact that the TCI form (34) (with fully nested pivots) is exact on all one-dimensional slices T_ℓ but not on the two-dimensional slices Π_ℓ. All 2-site TCI algorithms thus aim to iteratively improve the representation of the Π_ℓ slices.

4.3.1 Basic algorithm

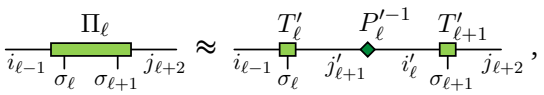
We start by presenting a TCI algorithm in a version based on LU factorization. In Sec. 4.3.2 we will describe its connection to the algorithm based on CI factorizations presented in prior work [12, 13]. The algorithm proceeds as follows:

(1) Start with an index $\hat{\boldsymbol{\sigma}}$ for which $F_{\hat{\boldsymbol{\sigma}}} \neq 0$, and construct initial pivots from it:
$\mathcal{I}_\ell = \{(\hat{\sigma}_1, \ldots, \hat{\sigma}_\ell)\}$ and $\mathcal{J}_{\ell+1} = \{(\hat{\sigma}_{\ell+1}, \ldots, \hat{\sigma}_{\mathcal{L}})\}$ for all ℓ.

(2) Sweeping back and forth over $\ell = 1, \ldots, \mathcal{L}-1$, perform the following update at each ℓ:

- Construct the Π_ℓ tensor (33c).
- View the tensor Π_ℓ as a matrix $F(\mathcal{I}_{\ell-1} \times \mathbb{S}_\ell, \mathbb{S}_{\ell+1} \times \mathcal{J}_{\ell+2})$ and perform its prrLU decomposition which approximates it as $\Pi_\ell \approx \widetilde{\Pi}_\ell$ with

$$[\Pi_\ell]_{i_{\ell-1}\sigma_\ell\sigma_{\ell+1}j_{\ell+2}} \approx [T_\ell'^{\sigma_\ell}]_{i_{\ell-1}j'_{\ell+1}} (P'_\ell)^{-1}_{j'_{\ell+1}i'_\ell} [T_\ell'^{\sigma_{\ell+1}}]_{i'_\ell j_{\ell+2}}, \tag{39}$$



  where $i'_\ell \in \mathcal{I}'_\ell \subset \mathcal{I}_{\ell-1} \times \mathbb{S}_\ell$ and $j'_{\ell+1} \in \mathcal{J}'_{\ell+1} \subset \mathbb{S}_{\ell+1} \times \mathcal{J}_{\ell+2}$ are the new pivots.
- Replace the old pivot lists $\mathcal{I}_\ell, \mathcal{J}_{\ell+1}$ by the new ones $\mathcal{I}'_\ell, \mathcal{J}'_{\ell+1}$. By construction, the nesting conditions $\mathcal{I}_{\ell-1} < \mathcal{I}'_\ell$, $\mathcal{J}'_{\ell+1} > \mathcal{J}_{\ell+2}$ are satisfied. The matrices P_ℓ, T_ℓ and $T_{\ell+1}$ are also updated along with the pivots, according to their definitions (33a, 33b). Note that this step may break the full nesting condition: one may have $\mathcal{I}_\ell < \mathcal{I}_{\ell+1}$ but not $\mathcal{I}'_\ell < \mathcal{I}_{\ell+1}$; similarly, one may have $\mathcal{J}_\ell > \mathcal{J}_{\ell+1}$ but not $\mathcal{J}_\ell > \mathcal{J}'_{\ell+1}$.

(3) Iterate step (2) until the specified tolerance is reached, or a specified number of times.

When pivots are left-nested up to $\ell - 1$ and right-nested up to $\ell + 2$ — a property that our algorithm actually preserves — (we say that the tensor train is *nested w.r.t.* Π_ℓ), then the following crucial relation holds (for a proof, see [13, App. C.2], or our App. A.3):

$$\left[\Pi_\ell - \widetilde{\Pi}_\ell\right]_{i_{\ell-1}\sigma_\ell\sigma_{\ell+1}j_{\ell+2}} = \left[F - \widetilde{F}\right]_{i_{\ell-1}\sigma_\ell\sigma_{\ell+1}j_{\ell+2}}, \tag{40}$$

for all $\sigma_\ell, \sigma_{\ell+1}$. Thus, the error made by approximating the local tensor Π_ℓ by its prrLU decomposition $\widetilde{\Pi}_\ell$ is also the error, on this two-dimensional slice, of approximating $F_{\boldsymbol{\sigma}}$ by the TCI decomposition $\widetilde{F}_{\boldsymbol{\sigma}}$. By construction, the TCI form (34) (with fully nested pivots) is exact on *one-dimensional slices*, $\mathcal{I}_{\ell-1} \times \mathbb{S}_\ell \times \mathcal{J}_{\ell+1}$, but not on the *two-dimensional slices* $\mathcal{I}_{\ell-1} \times \mathbb{S}_\ell \times \mathbb{S}_{\ell+1} \times \mathcal{J}_{\ell+2}$. Hence, the algorithm chooses the pivots in order to minimize the error on the latter.

The algorithm presented in this section deviates significantly from the one used by some of us in Ref. [12, 13]: there, new pivots could be added but they were never removed in order to maintain the full nesting condition. However, a close examination of [13, App. C.2] shows that partial nesting is sufficient to ensure Eq. (40). We use this fact to use an update strategy where the pivots $\mathcal{I}'_\ell, \mathcal{J}'_{\ell+1}$ are reset at each step (2) of the algorithm. The ability to discard "bad" pivots (e.g. ones found in early iterations that later turn out to be suboptimal) significantly improves the numerical stability of the present TCI algorithm compared to the original one [12]. This point will be discussed further in Sec. 4.3.3. If desired, full nesting can be restored at the end using 1-site TCI, discussed in Sec. 4.4.

4.3.2 CI vs prrLU

The TCI algorithm as described in this paper is also different from the standard TCI algorithm [12, 13] in that it uses prrLU instead of the CI decomposition for the Π_ℓ tensor. While CI and prrLU are equivalent, as shown in Sec. 3.3, the prrLU yields a more stable implementation, as it avoids inverting the pivot matrices P, which may become ill-conditioned. We emphasize again that we have found prrLU to be more efficient and stable than the alternative QR approach used in Appendix B of [13] to address the conditioning issue of the pivot matrices.

For convenience, we explicitly rewrite the correspondence between CI and LU factorization shown in Eqs. (32) as appropriate for the update of $\widetilde{\Pi}_\ell$:

$$\widetilde{\Pi}_\ell = T_\ell (P_\ell)^{-1} T_{\ell+1} = LDU = \begin{pmatrix} P_\ell & L_{11}DU_{12} \\ L_{21}DU_{11} & L_{21}DU_{12} \end{pmatrix}, \tag{41a}$$

$$P_\ell = L_{11}DU_{11}\,, \tag{41b}$$

$$T_\ell = \begin{pmatrix} L_{11}DU_{11} \\ L_{21}DU_{11} \end{pmatrix}, \qquad T_\ell P_\ell^{-1} = \begin{pmatrix} \mathbb{1} \\ L_{21}L_{11}^{-1} \end{pmatrix}, \tag{41c}$$

$$T_{\ell+1} = \begin{pmatrix} L_{11}DU_{11} & L_{11}DU_{12} \end{pmatrix}, \qquad P_\ell^{-1} T_{\ell+1} = \begin{pmatrix} \mathbb{1} & U_{11}^{-1}U_{12} \end{pmatrix}. \tag{41d}$$

Since U_{11} and L_{11} are triangular matrices, the two terms involving a matrix inversion can be computed in a stable manner using forward/backward substitution.

4.3.3 Pivot update method: Reset vs accumulative

In order to update the pivots in the TCI algorithm, we can use two different methods, which we call *reset* and *accumulative*.

- In reset mode, we recompute the full prrLU decomposition of Π_ℓ at each ℓ, hence reconstructing new pivots $\mathcal{I}_\ell$, $\mathcal{J}_{\ell+1}$. This version was presented in Sec. 4.3.1.
- In accumulative mode, we update the pivot lists $\mathcal{I}_\ell$, $\mathcal{J}_{\ell+1}$ by only *adding* pivots. Typically, pivots are added one at a time, thereby increasing χ_ℓ to $\chi_\ell+1$. Once a pivot has been added, it is never removed. This strategy preserves full nesting, thus ensuring the interpolation property of the TCI approximation. This is the method presented in Ref. [12, algorithm #5].

The main advantage of reset mode is that it eliminates bad pivots which are almost linearly dependent, thereby leading to poorly conditioned P matrices. These occur when the algorithm first explores configurations where $F_{\boldsymbol{\sigma}}$ is small and only later discovers other configurations with larger values of $F_{\boldsymbol{\sigma}'}$. In such cases, the late pivots correspond to a much larger absolute value of F than the first, leading to ill-conditioned P_ℓ. Therefore, in accumulative mode, it is crucial to choose as an initial pivot a point where F is of the same order of magnitude as its maximum. In reset mode, the bad pivots are automatically eliminated, which yields a better TCI approximation and very stable convergence. On the other hand, accumulative mode requires a (slightly) smaller number of values of F, as the exploration of configurations for finding pivots is kept to a minimum.

The runtime of both approaches scales as $O(\chi^3)$. Accumulative mode requires $O(\chi^2)$ per update and χ updates to reach a rank of χ. Reset mode requires $O(\chi^3)$ for each update, but typically converges within a small number of updates independently of χ.

We note that the pioneering work of Ref. [10] used a method similar to reset mode, recalculating the pivots at each step. MPS recompression was performed very differently, however, using a combination of SVD and the maximum volume principle, which led to slower scaling. Here, pivot optimization is done entirely within the LU decomposition.

4.3.4 Pivot search method: Full, rook or block rook

A crucial component of 2-site TCI algorithms is the search for pivots, as the largest elements of the error tensor $|\Pi_\ell - \widetilde{\Pi}_\ell|$. As discussed in Sec. 3.3.2, three different search modes are available: *Full search* is the simplest and most stable mode, but also most expensive, scaling as $\mathcal{O}(d^2)$. *Rook search* is a cheaper alternative, scaling as $\mathcal{O}(d)$ (since rows and columns are

explored alternatingly), and is almost as good in practice. Rook search is well adapted to accumulative mode [12] and is advantageous when the dimension d is large.

Block rook search is especially useful when used with reset pivot update mode. Indeed, it allows reusing previously found pivots and therefore reusing previously computed values of F. This is particularly useful when $F_{\boldsymbol{\sigma}}$ is an expensive function to evaluate on $\boldsymbol{\sigma}$. The algorithm requires $\mathcal{O}(n_{Rook}\chi^2 d)$ function evaluations to factorize a Π tensor.

4.3.5 Proposing pivots from outside of TCI

In its normal mode, TCI constructs new pivots by making local updates of existing pivots. In several situations, it is desirable to enrich the pivot search by proposing a list of values of the indices $\boldsymbol{\sigma}$ which the TCI algorithm is required to try as pivots. It is a way to incorporate prior knowledge about F into TCI. We call such values of $\boldsymbol{\sigma}$ global pivots. This section discusses our strategy to perform this operation in a stable way.

Given a list of global pivots, we split each index $\boldsymbol{\sigma}$ as $\boldsymbol{\sigma} = i_\ell \oplus j_{\ell+1}$ for all $\ell = 1, \dots \mathcal{L}-1$, and i_ℓ and $j_{\ell+1}$ are added to the corresponding pivot lists $\mathcal{I}_\ell$ and $\mathcal{J}_{\ell+1}$. This operation preserves nesting conditions. Next, we perform a prrLU decomposition of the pivot matrices P_ℓ to remove possible spurious pivots. Last, we perform a few sweeps using 2-sites TCI in reset mode to stabilize the pivots lists. We provide a simple example of global pivot addition in Appendix B.3.5.

Global pivot proposals can be useful in several situations. First, the TCI algorithm can experience some ergodicity issues as discussed in Sec. 4.3.6, which can be solved by adding some pivots explicitly. The construction of the Matrix Product Operators discussed in Section 7 belongs to this category. Second, the TCI decomposition of a tensor F_2 close to another F_1 for which the TCI is already known, e.g. due to an adiabatic change of some parameter, can benefit from initialization with the pivots of $\widetilde{F}_1$. Third, global pivot proposal can be used to separate the exploration of the configuration space (the way these global pivots are constructed) from the algorithm used to update the tensor train. For instance, one could use a separate algorithm to globally look for pivots where the TCI error is large using a separate global optimizer; then propose these pivots to TCI; and iteratively repeat the process until convergence.

The above algorithm, which we call *StrictlyNested*, works well but suffers from one (albeit relatively rare) problem: it occasionally discards perfectly valid proposed global pivots. This may happen when χ_ℓ depends on ℓ in such a manner that the MPS has a "constriction", i.e. a bond with a smaller dimension χ_ℓ than all others. Upon sweeping through this bond, some pivots will be deleted (which is fine), but that deletion will propagate upon continuing to sweep (which is a weakness of the algorithm).

A simple fix is to construct an *enlarged* tensor $\bar{\Pi}_\ell$ that extends Π_ℓ with additional rows and columns containing deleted pivots, thus retaining these for consideration as potential pivots. Concretely, denoting pivots obtained in a previous sweep by $\bar{\mathcal{I}}_\ell$ and $\bar{\mathcal{J}}_\ell$, we define

$$\bar{\Pi}_\ell = F([\mathcal{I}_{\ell-1} \times \mathbb{S}_\ell] \cup \bar{\mathcal{I}}_\ell \ , \ [\mathbb{S}_{\ell+1} \times \mathcal{J}_{\ell+2}] \cup \bar{\mathcal{J}}_{\ell+1}), \tag{42}$$

and use $\bar{\Pi}_\ell$ instead of Π_ℓ for the prrLU decomposition. We note that such enlargements can break nesting conditions, i.e. this is an *UnStrictlyNested* mode. However, we have not observed this to cause any problems in our numerical experiments.

4.3.6 Ergodicity

The construction of tensor trains using TCI is based on the exploration of configuration space. In analogy with what can happen with Monte Carlo techniques, this exploration may encounter ergodicity problems, remaining stuck in a subpart of the configuration space and not visiting other relevant parts. Examples where this may occur include: very sparse tensors $F_{\boldsymbol{\sigma}}$, where

TCI might miss some nonzero entries (see the Matrix Product Operator construction section 7 for an example); tensors with discrete symmetries, where the exploration may remain in one symmetry sector (relevant for the partition function of the Ising model, see Sec. 5.3); or multivariate functions with very narrow peaks.

All ergodicity problems that we have encountered so far could be fixed by proposing global pivots, as described in Sec. 4.3.5. For sparse tensors, one feeds the algorithm with a list of nonzero entries. For discrete symmetries, one initializes the algorithm with one configuration per symmetry sector. One could also consider more elaborate strategies that use a dedicated algorithm to explore new configurations, in analogy to the construction of complex moves when building a Monte Carlo algorithm. In fact, existing Monte Carlo algorithms could be used directly as way to propose global pivots. Such an algorithm would separate entirely the pivot exploration strategy from the way the tensor train is updated.

Let us illustrate the above ideas with a toy example. Consider a fermionic operator c ($c^\dagger$) that destroys (creates) an electron on a unique site ($\{c,c\}=\{c^\dagger,c^\dagger\}=0; \{c,c^\dagger\}=1$). We want to factorize

$$F_{\boldsymbol{\sigma}} = \langle a_{\sigma_1} \cdots a_{\sigma_\mathcal{L}} \rangle , \tag{43}$$

into a tensor train, where $a_0 = c$ and $a_1 = c^\dagger$ and the average is taken with respect to the state $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}c^\dagger|0\rangle$. For even $\mathcal{L}$, this tensor has only two non-zero elements, namely $F_{\boldsymbol{\sigma}} = 1/2$ for $\boldsymbol{\sigma}_1 = (1,0,1,0,\ldots,1,0)$ and $\boldsymbol{\sigma}_2 = (0,1,0,1,\ldots,0,1)$. This is due to the fermionic algebra, which implies $a_0 a_0 = cc = 0$ and $a_1 a_1 = c^\dagger c^\dagger = 0$. Using TCI in a standard way with one of the two elements as the starting pivot, TCI fails to find the second one. The reason is that the TCI updates are local, thus TCI quickly (wrongly) concludes that it correctly describes all configurations, whereas it correctly describes only the configurations that it has seen. A simple cure is to propose both $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ as global pivots. This works and is the easiest solution when the important configurations are known. An alternative cure is to *enlarge the configuration space* to obtain a larger but less sparse tensor. This idea is analogous to the concept of worms in Monte Carlo, where the configuration space is enlarged to remove constrains and allow for non-local updates. Here, we enlarge the local dimension from $d = 2$ to $d = 3$ by adding identity as a third operator, $a_2 = 1$. The new tensor is much less sparse and is correctly reconstructed using TCI with $(2,2,\ldots,2)$ as initial pivot. Restricting the resulting tensor train to $\sigma_i \in \{0,1\}$ yields the correct factorization.

4.3.7 Error estimation: Bare vs. environment

In the prrLU decomposition of the Π_ℓ tensor described in Sec. 4.3.1 above, each new pivot is chosen in order to minimize the *bare error* $|\Pi_\ell - \widetilde{\Pi}_\ell|_{i_{\ell-1}\sigma_\ell\sigma_{\ell+1}j_{\ell+2}}$. An alternative choice is to define an *environment error* whose minimization aims to find the best approximation of the "integrated" tensor $\sum_{\boldsymbol{\sigma}} F_{\boldsymbol{\sigma}}$, i.e. summed over all external indices (see Sec. III.B.4 of Ref. [13]). The environment error has the form $|L_{i_{\ell-1}} R_{j_{\ell+2}}||\Pi_\ell - \widetilde{\Pi}_\ell|_{i_{\ell-1}\sigma_\ell\sigma_{\ell+1}j_{\ell+2}}$, with left and right environment tensors defined as

$$L_{i_{\ell-1}} = \sum_{\sigma_1,\ldots,\sigma_{\ell-1}} [T_1^{\sigma_1} P_1^{-1} \cdots T_{\ell-1}^{\sigma_{\ell-1}} P_{\ell-1}^{-1}]_{1 i_{\ell-1}} , \qquad R_{j_{\ell+2}} = \sum_{\sigma_{\ell+2},\ldots,\sigma_\mathcal{L}} [P_{\ell+1}^{-1} T_{\ell+2}^{\sigma_{\ell+2}} \cdots P_{\mathcal{L}-1}^{-1} T_\mathcal{L}^{\sigma_\mathcal{L}}]_{j_{\ell+2} 1} . \tag{44}$$

Minimization of the environment error can be very efficient for the computation of integrals involving integrands with long tails. An example of improved accuracy using this *environment mode* is given in Fig. 7 of Ref. [13].

4.4 The 1-site and 0-site TCI algorithms

In this section, we propose two more algorithms complementing 2-site TCI: the 1-site and 0-site TCI algorithms. The names reflect the number σ-indices of the objects decomposed with

LU: Π, T or P tensors with 2, 1 or 0 σ-indices, respectively. The 2-site algorithms described above are more versatile, and only they can increase the bond dimension χ_ℓ, so they are almost always needed during the initial learning stage (unless global pivots are used to start with a large enough rank). However, the 1-site and 0-site TCI algorithms are faster than 2-site TCI, and the former can also be used to achieve full nesting.

4.4.1 The 1-site TCI algorithm

The 1-site TCI algorithm sweeps through the tensor train and compresses its T tensors using prrLU. In a forward sweep we view T_ℓ as a matrix with indices $(\mathcal{I}_{\ell-1} \times \mathbb{S}_\ell, \mathcal{J}_{\ell+1})$, regrouping the σ_ℓ index with the left index $i_{\ell-1}$. Using prrLU, we obtain new pivots $\mathcal{I}'_\ell$, $\mathcal{J}'_{\ell+1}$ to replace $\mathcal{I}_\ell$, $\mathcal{J}_{\ell+1}$, satisfying $\mathcal{I}'_\ell > \mathcal{I}_{\ell-1}$ and $\mathcal{J}'_{\ell+1} \subseteq \mathcal{J}_{\ell+1}$, and update T_ℓ, P_ℓ and $T_{\ell+1}$ accordingly. After the forward sweep, the pivots are fully left-nested, i.e. $\mathcal{I}_0 < \cdots < \mathcal{I}_{\mathcal{L}-1}$.

In a backward sweep, T_ℓ is viewed as a matrix with indices $(\mathcal{I}_{\ell-1}, \mathbb{S}_\ell \times \mathcal{J}_{\ell+1})$, so prrLU yields new pivots $\mathcal{I}'_{\ell-1} \subseteq \mathcal{I}_{\ell-1}$, $\mathcal{J}'_\ell > \mathcal{J}_{\ell+1}$, and corresponding updates of T_ℓ, $P_{\ell-1}$ and $T_{\ell-1}$. After the backward sweep, the pivots are fully right-nested, i.e. $\mathcal{J}_2 > \cdots > \mathcal{J}_{\mathcal{L}+1}$, and all bond dimensions meet the tolerance (i.e. are suitable for achieving the specified tolerance). However, the backward sweep preserves left-nesting only if taking the subset $\mathcal{I}'_{\ell-1} \subseteq \mathcal{I}_{\ell-1}$ does not remove any pivots, i.e. if actually $\mathcal{I}'_{\ell-1} = \mathcal{I}_{\ell-1}$. To achieve full nesting, left nesting can be restored by performing one more forward sweep at the same tolerance. This preserves right-nesting, because all bond dimensions already meet the tolerance, thus the last forward sweep removes no pivots from $\mathcal{J}_{\ell+1}$ for $\ell = 1, \ldots, \mathcal{L}-1$. For a related discussion in a different context, see Sec. 4.5.

1-site TCI can be used to (i) compress a TCI to a smaller rank; (ii) restore full nesting; (iii) improve the pivots at lower computational cost than its 2-site counterpart.

4.4.2 The 0-site TCI algorithm

The 0-site TCI algorithm sweeps through the pivot matrices P_ℓ, prrLU decomposing each to yield updated pivot lists $\mathcal{I}'_\ell$, $\mathcal{J}'_{\ell+1}$ that replace $\mathcal{I}_\ell$, $\mathcal{J}_{\ell+1}$. 0-site TCI breaks nesting conditions. Its main usage is to improving the conditioning of P_ℓ, by removing "spurious" pivots. For example, if a very large list of global pivots has been proposed, 0-site TCI can be used as a first filter to keep only the most relevant ones. It does not require new calls to F tensor elements and hence can be used even when F is no longer available.

4.5 CI- and LU-canonicalization

The MPS form $F_{\boldsymbol{\sigma}} = M_1^{\sigma_1} M_2^{\sigma_2} \cdots M_{\mathcal{L}}^{\sigma_{\mathcal{L}}}$ of a tensor is not unique. Indeed one can always replace $M_\ell \leftarrow M_\ell N_\ell$ and $M_{\ell+1} \leftarrow N_\ell^{-1} M_{\ell+1}$ for any ℓ and invertible matrix N_ℓ of appropriate dimension $(\chi_\ell \times \chi_\ell)$. This is known as the gauge freedom. One can exploit this freedom to write the MPS into *canonical forms*. A standard way is to express it as a product of left- and right-unitary matrices around an *orthogonality center*, using the SVD decomposition [2] (the SVD-canonical form). In this section, we show how an arbitrary MPS can be put in TCI form, described uniquely in terms of pivot lists and corresponding slices of F. We call the corresponding algorithm CI-canonicalization. LU-canonicalization is a variant thereof.

The different canonical forms offer different advantages for subsequent operations on the tensor train. The SVD-canonical form is widely used in the tensor network community to improve performance of certain contractions by exploiting the unitarity properties of the MPS matrices. It is also very useful for algorithms such as DMRG as it provides a degree of non-locality to an otherwise local optimization. The CI-canonical form, on the other hand, is made up entirely of slices of the original MPS, i.e. a selection of values of the function through the

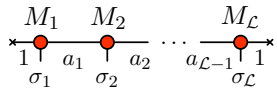
index sets $\mathcal{I}_\ell$ and $\mathcal{J}_\ell$. These set of points may have a value by themselves, e.g. as the starting point of a multi-variate optimization or to perform transformations (rotations, translations) in the case of quantics. Bringing a tensor into CI-canonical form is also a necessary step to enable the application of other TCI algorithms, such as TCI optimization (Sec. 4.3) or global pivot insertion (Sec. 4.3.5), which rely on the property that all core tensors of the MPS are defined through $\mathcal{I}_\ell$ and $\mathcal{J}_\ell$. LU canonicalization is a minor modification of CI canonicalization, and is mentioned here for completeness. The authors are not currently aware of any application unique to the LU-canonical form.

A simple way to put the MPS in a TCI form would be to apply the 2-site TCI to $F_{\boldsymbol{\sigma}}$, considered as a function of $\boldsymbol{\sigma}$. However, we present here a specific and *direct CI-canonicalization* algorithm to achieve this, based on the MPS structure. This algorithm has several advantages over the 2-site TCI: first, it is faster, taking only $O(\chi^3)$ operations (like the usual SVD-canonicalization) instead of $O(\chi^4)$;[1] second, it bypasses all the potential issues of the 2-site TCI algorithm discussed above, like ergodicity. Let us emphasize that while the CI-canonicalization algorithm can seem similar to the 1-site TCI algorithm, the two algorithms are actually different, as the former directly exploits the MPS structure of $F_{\boldsymbol{\sigma}}$.

4.5.1 CI-canonicalization.

Let us consider a MPS of the form

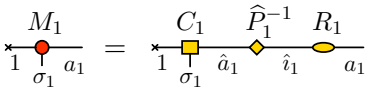
$$F_{\boldsymbol{\sigma}} = [M_1^{\sigma_1}]_{1a_1}[M_2^{\sigma_2}]_{a_1a_2}\cdots[M_{\mathcal{L}}^{\sigma_{\mathcal{L}}}]_{a_{\mathcal{L}-1}1} = \quad (45)$$



Here, the indices a_ℓ are ordinary MPS indices, *not* multi-indices i_ℓ or j_ℓ from pivot lists. CI-canonicalization is a sequence of exact transformations that convert the MPS to the TCI form of Eq. (34), built from T_ℓ and P_ℓ tensors that are slices of F carrying multi-indices i_ℓ, j_ℓ and that constitute full-rank matrices. We achieve this through three half-sweeps, involving exact (i.e. at machine precision) CI decompositions. A first forward sweep introduces left-nested lists $\widehat{\mathcal{I}}_\ell$ of row pivot multi-indices $\hat{\imath}_\ell$. Then, a backward sweep introduces right-nested lists $\mathcal{J}_\ell$ of column pivot multi-indices j_ℓ and matching subsets $\mathcal{I}_\ell \subset \widehat{\mathcal{I}}_\ell$ of row pivots i_ℓ (no longer left-nested). Finally, a second forward sweep restores left-nesting of row pivots. Important here is tracking the conversion from regular indices (a_ℓ) to row (i_ℓ, $\hat{\imath}_\ell$) and column (j_ℓ) multi-indices. We thus display these indices explicitly below.

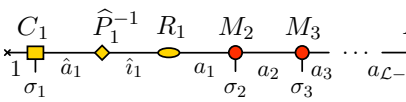
First forward sweep. We start with an exact CI decomposition (8) of M_1:

$$[M_1^{\sigma_1}]_{1a_1} = [C_1]_{\sigma_1\hat{a}_1}[\widehat{P}_1^{-1}]_{\hat{a}_1\hat{\imath}_1}[R_1]_{\hat{\imath}_1a_1}, \qquad (46)$$



Here, $\hat{\imath}_1 \in \widehat{\mathcal{I}}_1 \subseteq \{\sigma_1\}$ are new multi-indices labeling pivot rows. The hat on $\widehat{P}_1$ emphasizes that it is *not* a slice of F, since the $\hat{a}_1$ are not multi-indices. Defining matrices $C_1^{\sigma_1}$ with elements $[C_1^{\sigma_1}]_{1\hat{a}_1} \equiv [C_1]_{\sigma_1\hat{a}_1}$ we obtain

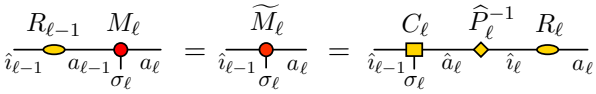
$$F_{\boldsymbol{\sigma}} = \left[C_1^{\sigma_1}\widehat{P}_1^{-1}R_1M_2^{\sigma_2}M_3^{\sigma_3}\cdots M_{\mathcal{L}}^{\sigma_{\mathcal{L}}}\right]_{11} = \quad (47)$$



[1] The complexity of using TCI for this purpose splits into $O(\chi^2)$ evaluations of the MPS which require $O(\chi^2)$ operations each. There is a possibility to cache the partial contractions of the MPS to bring the global cost down to $O(\chi^3)$ but the resulting algorithm is still inferior to the CI-canonicalization algorithm.

For $\ell \geq 2$ we iteratively define $\widetilde{M}_\ell^{\sigma_\ell} = R_{\ell-1} M_\ell^{\sigma_\ell}$ and group σ_ℓ with $\hat{\imath}_{\ell-1}$ to reshape $\widetilde{M}_\ell$ into a matrix which we factorize exactly with CI:

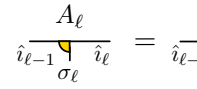
$$[R_{\ell-1} M_\ell^{\sigma_\ell}]_{\hat{\imath}_{\ell-1} a_\ell} = [\widetilde{M}_\ell]_{(\hat{\imath}_{\ell-1},\sigma_\ell) a_\ell} = [C_\ell^{\sigma_\ell}]_{\hat{\imath}_{\ell-1}\hat{a}_\ell} [\widehat{P}_\ell^{-1}]_{\hat{a}_\ell \hat{\imath}_\ell} [R_\ell]_{\hat{\imath}_\ell a_\ell}\,, \tag{48}$$



The tensor C_ℓ can be viewed as a matrix $C_\ell^{\sigma_\ell}$ with elements $[C_\ell^{\sigma_\ell}]_{\hat{\imath}_{\ell-1}\hat{a}_\ell} = [C_\ell]_{(\hat{\imath}_{\ell-1},\sigma_\ell)\hat{a}_\ell}$. The new row pivots are left-nested, $\hat{\imath}_\ell \in \widehat{\mathcal{I}}_\ell > \widehat{\mathcal{I}}_{\ell-1}$.

In practice, we do not calculate C_ℓ and $\widehat{P}_\ell$ separately. Instead, the prrLU decomposition directly yields the combination $A_\ell^{\sigma_\ell} = C_\ell^{\sigma_\ell} \widehat{P}_\ell^{-1}$:

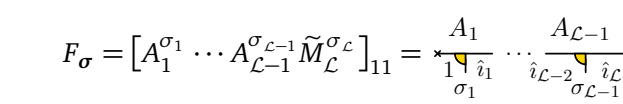
$$[A_\ell^{\sigma_\ell}]_{\hat{\imath}_{\ell-1}\hat{\imath}_\ell} = [C_\ell^{\sigma_\ell}]_{\hat{\imath}_{\ell-1}\hat{a}_\ell} [\widehat{P}_\ell^{-1}]_{\hat{a}_\ell \hat{\imath}_\ell}\,, \qquad \tag{49}$$



By construction, see Eq. (10a), this product collapses to $[A_\ell^{\sigma_\ell}]_{\hat{\imath}_{\ell-1}\hat{\imath}_\ell} = \delta_{\hat{\imath}_{\ell-1}\oplus(\sigma_\ell),\hat{\imath}_\ell}$ whenever $\hat{\imath}_{\ell-1}\oplus(\sigma_\ell) \in \widehat{\mathcal{I}}_\ell$ (see also App. A.3).

After a full forward sweep to the very right we arrive at a tensor train of the form

$$F_{\boldsymbol{\sigma}} = \big[A_1^{\sigma_1} \cdots A_{\mathcal{L}-1}^{\sigma_{\mathcal{L}-1}} \widetilde{M}_{\mathcal{L}}^{\sigma_{\mathcal{L}}}\big]_{11} = \tag{50}$$



Here, the row pivots are by construction all left-nested as $\widehat{\mathcal{I}}_0 < \cdots < \widehat{\mathcal{I}}_{\mathcal{L}-1}$. This ensures the following important property: for any $\ell \leq \mathcal{L}-1$, the product $A_1 \cdots A_\ell$ collapses telescopically (starting from $A_1 A_2$) if evaluated on any pivot $\bar{\imath}_\ell = (\bar{\sigma}_1, \ldots, \bar{\sigma}_\ell) \in \widehat{\mathcal{I}}_\ell$ (cf. Eq. (A.12)):

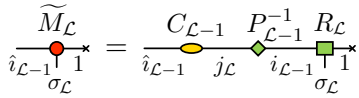
$$\begin{array}{ccccc} A_1 & & A_2 & & A_\ell \\ 1 & \bar{\imath}_1 & \bar{\imath}_1 & \bar{\imath}_2 & \cdots \bar{\imath}_{\ell-1} \bar{\imath}_\ell \\ \bar{\sigma}_1 & & \bar{\sigma}_2 & & \bar{\sigma}_\ell \end{array} = [A_1^{\bar{\sigma}_1} A_2^{\bar{\sigma}_2} \cdots A_\ell^{\bar{\sigma}_\ell}]_{1\bar{\imath}_\ell} = \delta_{\bar{\imath}_\ell \hat{\imath}_\ell} \quad \text{if} \quad \hat{\imath}_\ell \in \widehat{\mathcal{I}}_\ell\,. \tag{51}$$

If Eq. (50) is evaluated on pivot configurations of $\widetilde{M}_{\mathcal{L}}$, having $\bar{\imath}_{\mathcal{L}-1} \in \widehat{\mathcal{I}}_{\mathcal{L}-1}$, we find via Eq. (51) that $F_{\bar{\imath}_{\mathcal{L}-1}\oplus(\sigma_{\mathcal{L}})} = [\widetilde{M}_{\mathcal{L}}^{\sigma_{\mathcal{L}}}]_{\bar{\imath}_{\mathcal{L}-1},1}$. Thus, $\widetilde{M}_{\mathcal{L}}$ is a slice of F, namely $\widetilde{M}_{\mathcal{L}} = F(\widehat{\mathcal{I}}_{\mathcal{L}-1}, \mathbb{S}_{\mathcal{L}})$. All C_ℓ and $\widehat{P}_\ell$ have full rank when viewed as matrices $[C_\ell]_{(\hat{\imath}_{\ell-1},\sigma_\ell)\hat{a}_\ell}$ and $\widehat{P}_{\hat{\imath}_\ell \hat{a}_\ell}$. However, C_ℓ and $\widetilde{M}_{\mathcal{L}}$ may still be rank-deficient when viewed as matrices $[C_\ell]_{\hat{\imath}_{\ell-1}(\sigma_\ell,\hat{a}_\ell)}$ or $[\widetilde{M}_{\mathcal{L}}]_{\hat{\imath}_{\mathcal{L}-1}\sigma_{\mathcal{L}}}$.

Backward sweep. Starting from Eq. (50), we sweep backward to generate right-nested column multi-indices j_ℓ. The CI factorizations are analogous to those of the forward sweep, with two differences: they group σ_ℓ with column (not row) indices prior to factorization; the resulting P_ℓ and R_ℓ matrices are slices of F, thus revealing the bond dimensions of F.

We initialize the backward sweep by factorizing $\widetilde{M}_{\mathcal{L}}^{\sigma_{\mathcal{L}}}$ exactly as $C_{\mathcal{L}-1} P_{\mathcal{L}-1}^{-1} R_{\mathcal{L}}^{\sigma_{\mathcal{L}}}$:

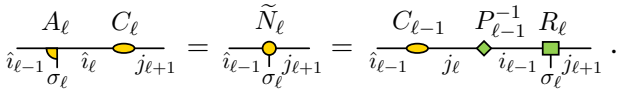
$$[\widetilde{M}_{\mathcal{L}}^{\sigma_{\mathcal{L}}}]_{\hat{\imath}_{\mathcal{L}-1}1} = [C_{\mathcal{L}-1}]_{\hat{\imath}_{\mathcal{L}-1} j_{\mathcal{L}}} [P_{\mathcal{L}-1}^{-1}]_{j_{\mathcal{L}} i_{\mathcal{L}-1}} [R_{\mathcal{L}}]_{i_{\mathcal{L}-1}\sigma_{\mathcal{L}}}\,, \qquad \tag{52}$$



Here, $j_{\mathcal{L}} \in \mathcal{J}_{\mathcal{L}} \subseteq \{\sigma_{\mathcal{L}}\}$ are multi-indices labeling pivot columns; $i_{\mathcal{L}-1} \in \mathcal{I}_{\mathcal{L}-1} \subseteq \widehat{\mathcal{I}}_{\mathcal{L}-1}$ are row pivots. Note that $R_{\mathcal{L}}$ and $P_{\mathcal{L}-1}$, being subslices of $\widetilde{M}_{\mathcal{L}}$, are slices of F, namely $R_{\mathcal{L}} = F(\mathcal{I}_{\mathcal{L}-1}, \mathbb{S}_{\mathcal{L}})$ and $P_{\mathcal{L}} = F(\mathcal{I}_{\mathcal{L}-1}, \mathcal{J}_{\mathcal{L}})$. We thus make the identification $T_{\mathcal{L}} = R_{\mathcal{L}}$.

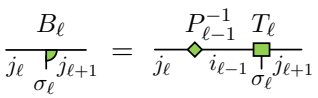
For $\ell \leq \mathcal{L}-1$ we iteratively define $\widetilde{N}_\ell^{\sigma_\ell} = A_\ell^{\sigma_\ell} C_\ell$ and factorize it as $C_{\ell-1} P_{\ell-1}^{-1} R_\ell^{\sigma_\ell}$:

$$[A_\ell^{\sigma_\ell}]_{\hat{\imath}_{\ell-1}\hat{\imath}_\ell} [C_\ell]_{\hat{\imath}_\ell j_{\ell+1}} = [\widetilde{N}_\ell]_{\hat{\imath}_{\ell-1}(\sigma_\ell, j_{\ell+1})} = [C_{\ell-1}]_{\hat{\imath}_{\ell-1} j_\ell} [P_{\ell-1}^{-1}]_{j_\ell i_{\ell-1}} [R_\ell^{\sigma_\ell}]_{i_{\ell-1} j_{\ell+1}}\,, \tag{53}$$



Here, the new column multi-indices are right-nested, $j_\ell \in \mathcal{J}_\ell > \mathcal{J}_{\ell+1}$, while the row multi-indices are a subset of the previous ones, $i_{\mathcal{L}-1} \in \mathcal{I}_{\mathcal{L}-1} \subseteq \widehat{\mathcal{I}}_{\mathcal{L}-1}$ (thus possibly breaking left-nesting, $\mathcal{I}_{\ell-1} \not< \mathcal{I}_\ell$). We show below that R_ℓ is a slice of F, thus we rename it $T_\ell = R_\ell$, and that $P_{\ell-1}$, too, is a slice of F. We also define $B_\ell^{\sigma_\ell} = P_{\ell-1}^{-1} T_\ell^{\sigma_\ell}$,

$$[B_\ell^{\sigma_\ell}]_{j_\ell j_{\ell+1}} = [P_{\ell-1}^{-1}]_{j_\ell i_{\ell-1}} [T_\ell^{\sigma_\ell}]_{i_{\ell-1} j_{\ell+1}}\,, \qquad$$

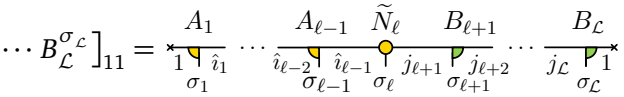


(54)

Via Eq. (10b) it collapses to $[B_\ell^{\sigma_\ell}]_{j_\ell, j_{\ell+1}} = \delta_{j_\ell,(\sigma_\ell)\oplus j_{\ell+1}}$ if $(\sigma_\ell)\oplus j_{\ell+1} \in \mathcal{J}_\ell$. Importantly, the inner summation for B_ℓ now involves multi-indices $i_{\ell-1}$ (for A_ℓ it still involved $\hat{a}_\ell$ indices).

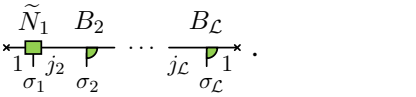
Sweeping backward up to site ℓ, and then all the way to the very left, we obtain

$$F_{\boldsymbol{\sigma}} = \left[A_1^{\sigma_1} \cdots A_{\ell-1}^{\sigma_{\ell-1}} \widetilde{N}_\ell^{\sigma_\ell} B_{\ell+1}^{\sigma_{\ell+1}} \cdots B_{\mathcal{L}}^{\sigma_{\mathcal{L}}}\right]_{11} =$$



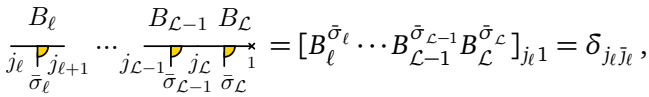
(55)

$$= [\widetilde{N}_1^{\sigma_1} B_2^{\sigma_2} \cdots B_{\mathcal{L}}^{\sigma_{\mathcal{L}}}]_{11} =$$



(56)

In Eq. (55), the column pivots are by construction all right-nested as $\mathcal{J}_{\ell+1} > \cdots > \mathcal{J}_{\mathcal{L}+1}$, and in Eq. (56) they are fully right-nested, $\mathcal{J}_2 > \cdots > \mathcal{J}_{\mathcal{L}+1}$. Importantly, this ensures that for any $\ell \geq 2$ the product $B_\ell \cdots B_{\mathcal{L}}$ collapses telescopically (starting from $B_{\mathcal{L}-1}B_{\mathcal{L}}$) if it is evaluated on any pivot $\bar{\jmath}_\ell = (\bar{\sigma}_\ell, \ldots, \bar{\sigma}_{\mathcal{L}}) \in \mathcal{J}_\ell$ (cf. Eq. (A.12b)):



$$= [B_\ell^{\bar{\sigma}_\ell} \cdots B_{\mathcal{L}-1}^{\bar{\sigma}_{\mathcal{L}-1}} B_{\mathcal{L}}^{\bar{\sigma}_{\mathcal{L}}}]_{j_\ell 1} = \delta_{j_\ell \bar{\jmath}_\ell}\,, \quad \forall\, \bar{\jmath}_\ell \in \mathcal{J}_\ell\,. \tag{57}$$

Consider Eq. (55) with $\ell > 1$. If evaluated on pivot configurations of $\widetilde{N}_\ell$, having $\bar{\imath}_{\ell-1} \in \widehat{\mathcal{I}}_{\ell-1}$ and $\bar{\jmath}_{\ell+1} \in \mathcal{J}_{\ell+1}$, it collapses telescopically via Eqs. (50) and (57) to $F_{\bar{\imath}_{\ell-1}\oplus\sigma_\ell\oplus\bar{\jmath}_{\ell+1}} = [\widetilde{N}_\ell^{\sigma_\ell}]_{\bar{\imath}_{\ell-1}\bar{\jmath}_{\ell+1}}$. Therefore, $\widetilde{N}_\ell$ is a slice of F, namely $\widetilde{N}_\ell = F(\widehat{\mathcal{I}}_{\ell-1}, \mathbb{S}_\ell, \mathcal{J}_{\ell+1})$. It follows that the same is true for its subslices, $T_\ell = R_\ell = F(\mathcal{I}_{\ell-1}, \mathbb{S}_\ell, \mathcal{J}_{\ell+1})$ and $P_{\ell-1}(\mathcal{I}_{\ell-1}, \mathcal{J}_\ell)$, as announced above. Therefore, the CI factorization of $\widetilde{N}_\ell$ reveals the bond dimension of F for bond $\ell-1$, namely $\chi_{\ell-1} = |\mathcal{I}_{\ell-1}| = |\mathcal{J}_\ell|$. The latter is an intrinsic property of F and will remain unchanged under arbitrary gauge transformations (e.g. exact SVDs or CIs) on its bonds. A telescope argument shows that $\widetilde{N}_1$ in (56) is a slice of F, too, thus we identify $T_1 = \widetilde{N}_1 = F(\mathbb{S}_1, \mathcal{J}_2)$.

Using $B_\ell^{\sigma_\ell} = P_{\ell-1}^{-1} T_\ell^{\sigma_\ell}$ in Eq. (56), we obtain a tensor train in the TCI form of Eq. (34), namely $F_{\boldsymbol{\sigma}} = [T_1^{\sigma_1} P_1^{-1} T_2^{\sigma_2} \cdots P_{\mathcal{L}-1}^{-1} T_{\mathcal{L}}^{\sigma_{\mathcal{L}}}]_{11}$. Here, all ingredients are slices of F, labeled by multi-indices, and each T_ℓ is full rank for both ways of viewing it as a matrix, $[T]_{(i_{\ell-1},\sigma_\ell) j_{\ell+1}}$ or $[T]_{i_{\ell-1}(\sigma_\ell, j_{\ell+1})}$. The column pivots are fully right-nested. However, the row pivots are *not* fully left-nested, since the backward sweep dropped some row pivots.

Second forward sweep. To obtain a tensor train in fully nested TCI form, we perform a second exact forward sweep, using the 1-site TCI algorithm of Sec. 4.4.1. This generates fully left-nested row pivots. Moreover, since all bond dimensions have already been revealed during the backward sweep, no column pivots are lost during the second forward sweep, thus the column pivots remain fully right-nested. More explicitly: during the second forward sweep, the rank of $[T_\ell]_{(i_{\ell-1},\sigma_\ell) i_\ell}$ is equal to the number of its columns, $\chi_\ell = |\mathcal{I}_i|$, hence this matrix has full rank. Therefore, its exact CI decomposition retains all columns, loosing none. The resulting tensor train is fully nested, as desired.

Table 2: Computational cost of the main TCI algorithms in `xfac` / `tci.jl`.

action	variant		calls to F_σ	algebra cost
iterate	rook piv.	2-site	$\mathcal{O}(\chi^2 d n_{\text{rook}} \mathcal{L})$	$\mathcal{O}(\chi^3 d n_{\text{rook}} \mathcal{L})$
	full piv.	2-site	$\mathcal{O}(\chi^2 d^2 \mathcal{L})$	$\mathcal{O}(\chi^3 d^2 \mathcal{L})$
	full piv.	1-site	$\mathcal{O}(\chi^2 d \mathcal{L})$	$\mathcal{O}(\chi^3 d \mathcal{L})$
	full piv.	0-site	0	$\mathcal{O}(\chi^3 \mathcal{L})$
achieve full nesting			$\mathcal{O}(\chi^2 d \mathcal{L})$	$\mathcal{O}(\chi^3 d \mathcal{L})$
add n_p global pivots			$\mathcal{O}\big((2\chi + n_p) n_p \mathcal{L}\big)$	$\mathcal{O}\big((\chi + n_p)^3 \mathcal{L}\big)$
compress tensor train	SVD		0	$\mathcal{O}(\chi^3 d \mathcal{L})$
	LU			
	CI			

CI-canonicalization with compression. CI-canonicalization can optionally be combined with compression at the cost of an extra half-sweep. Then, the sequence becomes: (i) An exact forward sweep builds row indices $\{\hat{\imath}_\ell\}$. (ii) A backward sweep with compression builds column indices $\{j_\ell\}$ according to a specified tolerance τ and/or rank χ, while possibly reducing row indices from $\{\hat{\imath}_\ell\}$ to $\{i_\ell\}$. (iii) A forward sweep with compression finalizes row indices according to the specifications while possibly further reducing column indices; this yields a proper TCI form with the specified τ and/or χ. (iv) A final optional backward sweep without compression restores full nesting.

4.5.2 LU-canonicalization

LU-canonicalization proceeds in a similar manner, but instead of the CI decomposition $CP^{-1}R$ it iteratively uses the corresponding LU decomposition LDU, where L is lower-triangular, U upper-triangular and D diagonal. Forward sweeps generate $LLL\cdots$ products while absorbing DU factors rightwards; backward sweeps generate $\cdots UUU$ products while absorbing LD factors leftwards. In this manner, one can express F in the form $L_1 \cdots L_{\ell-1} \widetilde{N}_\ell U_{\ell+1} \cdots U_{\mathcal{L}}$, for any $\ell = 1, \ldots, \mathcal{L}$, if desired.

4.6 High-level algorithms

We have now enlarged our toolbox with several flavors of TCI algorithms and canonical forms with various options and variants. These algorithms can be combined in numerous ways to provide more abstract, high-level algorithms for different tasks. The best combination will depend on the intended application, and we provide some rough practical guidelines below. The corresponding computational costs are listed in Table 2.

- 2-site TCI in *accumulative* plus *rook pivoting* mode is the fastest technique. It requires the least pivot exploration and very often provides very good results on its own. The accuracy can be improved, if desired, by following this with a few (cheap) 1-site TCI sweeps to reset the pivots.
- 2-site TCI in *reset* plus *rook pivoting* mode is marginally more costly than the above but more stable. It is a good default. For small d, one should use the *full* search, which is even more stable and involves almost no additional cost if $d \leq 2n_{\text{rook}}$.
- If good heuristics for proposing pivots are available or ergodicity issues arise, one should consider switching to *global pivot proposal* followed by 2-site TCI.
- To obtain the best final accuracy at fixed χ, one can build a TCI with a higher rank $\chi' > \chi$, then compress it using either SVD or CI recompression.

- For calculations of integrals or sums, we recommend the environment mode. In some calculations, we have observed it to increase the accuracy by two digits for the same computational cost.

4.7 Operations on tensor trains

The various TCI algorithms can be combined with other MPS algorithms [2,9] in various ways. Let us mention a few examples.

Function composition. Given a TCI $\widetilde{F}_{\boldsymbol{\sigma}}$ approximating a function f, its composition with another function $g(f(x))$, can be performed by constructing another TCI, $\widetilde{G}_{\boldsymbol{\sigma}} \approx g(\widetilde{F}_{\boldsymbol{\sigma}})$. The repeated evaluations of $\widetilde{F}_{\boldsymbol{\sigma}}$ required for this can be accelerated by caching partial contractions of the tensor train. This gives a runtime complexity of $\mathcal{O}(\chi_{\widetilde{F}}\chi_{\widetilde{G}}^3 d\mathcal{L})$, where $\chi_{\widetilde{F}}$ and $\chi_{\widetilde{G}}$ are the ranks of $\widetilde{F}$ and $\widetilde{G}$. Since the tensors T_ℓ are slices of $\widetilde{F}$, the new TCI $\widetilde{G}$ can be initialized by applying g to each element of T_ℓ. For simple, monotonically increasing functions g, the subsequent optimization typically converges very quickly.

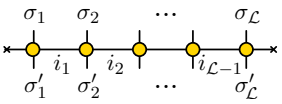
Element-wise tensor addition. Given two tensor trains, $\widetilde{F} = M_1 M_2 \cdots M_{\mathcal{L}}$ and $\widetilde{F}' = M_1' M_2' \cdots M_{\mathcal{L}}'$, their element-wise sum $\widetilde{F}''_{\boldsymbol{\sigma}} = \widetilde{F}_{\boldsymbol{\sigma}} + \widetilde{F}'_{\boldsymbol{\sigma}}$ can be computed by creating block matrices,

$$M_\ell''^{\sigma_\ell} = \begin{pmatrix} M_\ell^{\sigma_\ell} & 0 \\ 0 & M_\ell'^{\sigma_\ell} \end{pmatrix}, \tag{58}$$

and recompressing the resulting tensor train $\widetilde{F}''_{\boldsymbol{\sigma}} = \mathrm{Tr}(M_1''^{\sigma_1} M_2''^{\sigma_2} \cdots M_{\mathcal{L}}''^{\sigma_{\mathcal{L}}})$ using the CI-canonicalization algorithm. The total runtime complexity is dominated by that of the recompression, namely $\mathcal{O}\big((\chi+\chi')^3 d\mathcal{L}\big)$, where χ and χ' are the ranks of $\widetilde{F}$ and $\widetilde{F}'$. An advantage over the conventional SVD-based recompression is that the resulting MPS is truncated in terms of the maximum norm rather than the Frobenius norm, which can be more accurate for certain applications (see Sec. 7 for an example).

Matrix-vector contractions. Consider the contraction $G_{\boldsymbol{\sigma}'\boldsymbol{\sigma}} F_{\boldsymbol{\sigma}}$ in a $d^{\mathcal{L}}$-dimensional space. If G and F are compressible tensors, TCI can be used to approximate them by an MPO and MPS, respectively, where the former is of the form

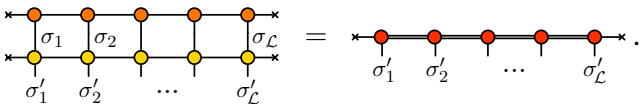
$$G_{\boldsymbol{\sigma}'\boldsymbol{\sigma}} \approx \widetilde{G}_{\boldsymbol{\sigma}'\boldsymbol{\sigma}} = [W_1]_{1 i_1}^{\sigma_1' \sigma_1} [W_2]_{i_1 i_2}^{\sigma_2' \sigma_2} \cdots [W_{\mathcal{L}}]_{i_{\mathcal{L}-1} 1}^{\sigma_{\mathcal{L}}' \sigma_{\mathcal{L}}} =$$



. (59)

Their contraction yields another MPS:

$$G_{\boldsymbol{\sigma}'\boldsymbol{\sigma}} F_{\boldsymbol{\sigma}} \approx \widetilde{G}_{\boldsymbol{\sigma}'\boldsymbol{\sigma}} \widetilde{F}_{\boldsymbol{\sigma}} =$$



. (60)

The MPO-MPS contraction can be computed exactly by performing the sum $\sum_{\boldsymbol{\sigma}}$, yielding an MPS with bond dimensions $\chi_{\ell,\widetilde{G}}\chi_{\ell,\widetilde{F}}$. The standard, SVD-based MPS toolbox offers two ways to obtain a compressed version of this result: (i) Fitting the exact result to an MPS with reduced bond dimensions; or (ii) zip-up compression, where the MPO-MPS contraction is performed one site at a time, followed by a local compression before proceeding to the next site [51–53]. TCI in principle offers further options, e.g. zip-up compression as in (ii), but performing all compressions using CI instead of SVD. The computational times of all these options are $\mathcal{O}(\chi^4\mathcal{L})$ for $\chi_{\widetilde{G}} = \chi_{\widetilde{F}} = \chi_{\widetilde{G}\widetilde{F}} = \chi$. The potential advantages of TCI- or CI-based contractions are twofold: the resulting MPS is truncated in terms of the maximum norm; and we can use the rook search, which can be efficient for large local dimensions d. To what extent TCI-based MPO-MPS contraction schemes have a chance of outperforming SVD-based ones will depend on context and is a question to be explored in future work.

4.8 Relation to machine learning

In this section we briefly compare and contrast TCI with other learning approaches such as deep neural network approaches.

TCI unfolding algorithms construct MPS representations $\widetilde{F}$ for F by systematically learning its structure. *Learning* the tensor F in the traditional machine learning sense would amount to the following sequence: (1) draw a training set of configurations/values $\{\boldsymbol{\sigma}, F_{\boldsymbol{\sigma}}\}$; (2) design a model $\widetilde{F}_{\boldsymbol{\sigma}}$ (typically a deep neural network); (3) fit the model to the training set by minimizing the error $\|F-\widetilde{F}\|$, measured w.r.t. to some norm (typically using a variant of stochastic gradient descent); and (4) use the model to evaluate $\widetilde{F}_{\boldsymbol{\sigma}}$ for new configurations. TCI implements this program with a few very important differences:

(1) TCI does not work with a given data set; instead, it actively requests the configurations that are likely to bring the most new information on the tensor (active learning).

(2) The model is not a neural network but a tensor train, i.e. a tensor network (a highly structured model). If F has a low-rank structure it can be accurately approximated by a low-rank tensor train $\widetilde{F}$, with an exponentially smaller memory footprint. For TCI to learn $\widetilde{F}$, the number of samples of $F_{\boldsymbol{\sigma}}$ requested by TCI will be $\ll d^{\mathcal{L}}$.

(3) The actual TCI algorithm used to minimize the error $\|F-\widetilde{F}\|$ is conceptually very different from gradient descent. It guarantees that the error is smaller than a specified tolerance τ for all known samples.

(4) Once $\widetilde{F}$ has been found, its elements $\widetilde{F}_{\boldsymbol{\sigma}}$ can be computed for *all* configurations $\boldsymbol{\sigma}$. This by itself may not seem like progress, since we had assumed that one could call any $F_{\boldsymbol{\sigma}}$ to begin with. Nevertheless, access to any $\widetilde{F}_{\boldsymbol{\sigma}}$ may be useful in cases where accessing $F_{\boldsymbol{\sigma}}$ is computationally expensive (e.g. the result of a complex simulation), or possible only in a limited time window (e.g. while collecting experimental data). Much more importantly, the tensor train structure of $\widetilde{F}$ permits subsequent operations (such as computing $\sum_{\boldsymbol{\sigma}} F_{\boldsymbol{\sigma}}$ over all configurations) to be performed exponentially faster.

5 Application: Computing integrals and sums

We now turn to practical illustrations of TCI in action. The following three sections give examples of various TCI applications, together with code listings illustrating how they can be coded using `xfac` or `TCI.jl` libraries.

The present section deals with the most obvious application of TCI: computing large integrals and sums. The basic idea has already been briefly introduced in Sec. 2.2. Here, we provide more details, a further example and the code listing used to compute it.

For historical reasons the `xfac` library implements two sets of algorithms corresponding to two classes `TensorCI1` and `TensorCI2`. The former is based on CI in accumulative mode and will eventually be deprecated while the latter is based on prrLU and supports many different modes. The Julia package `TCI.jl` follows closely the implementation of `TensorCI2`.

5.1 Quadratures for multivariate integrals

Consider a multi-dimensional integral, $\int_D d^{\mathcal{N}}\mathbf{x} f(\mathbf{x})$, with $\mathbf{x} = (x_1, \ldots, x_{\mathcal{N}})$, over a domain $D = D_1 \times \cdots \times D_{\mathcal{N}}$. (We here denote the number of variables by $\mathcal{N}$ (not $\mathcal{L}$), for notational consistency with Sec. 6 and Refs. [13, 15].) For each variable $x_\ell \in D_\ell$ we choose a grid of discretization points $\{x_\ell(\sigma_\ell)\}$, enumerated by an index $\sigma_\ell = 1, \ldots, d_\ell$, and an associated grid of quadrature weights $\{w_\ell(\sigma_\ell)\}$, such that its 1D integral is represented by the quadrature

rule $\int_{D_\ell} dx_\ell f(x_\ell) \approx \sum_{\sigma_\ell=1}^{d_\ell} w_\ell(\sigma_\ell) f\big(x_\ell(\sigma_\ell)\big)$. A typical choice would be the Gauss–Kronrod or Gauss–Legendre quadrature. Then, we use the natural tensor representation F (Eq. (2)) of f and its TCI unfolding $\widetilde{F}$ to obtain a factorized representation of the function,

$$f\big(\mathbf{x}(\boldsymbol{\sigma})\big) = F_{\boldsymbol{\sigma}} \simeq \widetilde{F}_{\boldsymbol{\sigma}} = \prod_{\ell=1}^{\mathcal{N}} M_\ell^{\sigma_\ell}\,. \tag{61}$$

Since $\widetilde{F}$ does not incorporate quadrature weights, this is called an *unweighted* unfolding. The $\mathcal{N}$-fold integral over f can thus be computed as [8, 12, 13]

$$\int_D d^{\mathcal{N}}\mathbf{x} f(\mathbf{x}) \approx \sum_{\boldsymbol{\sigma}} \Big(\prod_{\ell=1}^{\mathcal{N}} w_\ell(\sigma_\ell)\Big) f\big(\mathbf{x}(\boldsymbol{\sigma})\big) \approx \prod_{\ell=1}^{\mathcal{N}} \Big[\sum_{\sigma_\ell=1}^{d_\ell} w_\ell(\sigma_\ell) M_\ell^{\sigma_\ell}\Big]. \tag{62}$$

The first approximation $\approx$ refers to the error of the quadrature rule (controlled by the number of points d_ℓ in the discretization of each variable). The second $\approx$ is the factorization error (controlled by the rank χ) of the unfolding (61). Thus, the computation of one $\mathcal{N}$-dimensional integral has been replaced by $\mathcal{N}\chi^2$ exponentially easier problems, namely 1-dimensional integrals that each amount to performing a sum $\sum_{\sigma_\ell}$.

An alternative to unweighted unfolding is *weighted unfolding*, which unfolds the weighted tensor $\big(\prod_{\ell=1}^{\mathcal{N}} w_\ell(\sigma_\ell)\big) f\big(\mathbf{x}(\boldsymbol{\sigma})\big) = F_{\boldsymbol{\sigma}} \simeq \widetilde{F}_{\boldsymbol{\sigma}} = \prod_{\ell=1}^{\mathcal{N}} M_\ell^{\sigma_\ell}$. Then, the integral is given by

$$\int_D d^{\mathcal{N}}\mathbf{x} f(\mathbf{x}) \approx \sum_{\boldsymbol{\sigma}} \Big(\prod_{\ell=1}^{\mathcal{N}} w_\ell(\sigma_\ell)\Big) f\big(\mathbf{x}(\boldsymbol{\sigma})\big) \approx \prod_{\ell=1}^{\mathcal{N}} \Big[\sum_{\sigma_\ell=1}^{d_\ell} M_\ell^{\sigma_\ell}\Big]. \tag{63}$$

The weighted tensor has the same rank as the unweighted one since the weights form a rank-1 MPS. The weighted unfolding can sometimes be more efficient than unweighted unfolding—achieving higher accuracy for a given χ—since the error estimation during the TCI construction includes information about the weights. The weighted unfolding is typically combined with the use of the environment error that directly targets the best error for the calculation of integrals.

5.2 Example code for integrating multivariate functions

Next, we illustrate how TCI computations of multivariate integrals can be performed using the `xfac` toolbox. For definiteness, we consider a toy example from Ref. [54] for which the result is known analytically: the computation of the following integral over a hypercube:

$$I^{(\mathcal{N})} = \int_{[0,1]^{\mathcal{N}}} dx_1 \cdots dx_{\mathcal{N}} f(\mathbf{x}), \qquad f(\mathbf{x}) = \frac{2^{\mathcal{N}}}{1 + 2\sum_{\ell=1}^{\mathcal{N}} x_\ell}\,. \tag{64}$$

For $\mathcal{N} = 5$, the analytical solution of above integral is

$$I^{(5)} = [-65205\log(3) - 6250\log(5) + 24010\log(7) + 14641\log(11)]/24\,. \tag{65}$$

```
import xfacpy
from math import log

N = 5  # Number of dimensions


def f(x):  # Integrand function
  f.neval += 1
  return 2**N / (1 + 2 * sum(x))
```

```
10
11
12 f.neval = 0
13
14 # Exact integral value in 5 dimensions
15 i5 = (- 65205 * log(3) - 6250 * log(5) + 24010 * log(7) + 14641 * log(11)) / 24
16
17 # Gauss-Kronrod abscissas (xell) and weights (well)
18 xell, well = xfacpy.GK15(0, 1)
19
20 # TCI1 Tensor factorization, no environment
21 tci = xfacpy.CTensorCI1(f, [xell] * N)
22
23 # Estimate integral and error
24 for hsweep in range(14):
25   tci.iterate()
26   # calculate the integal over the hypercube
27   itci = tci.get_TensorTrain().sum([well] * N)
28   print("hsweep= {}, neval= {}, I_tci= {:e}, |I_tci - I_exact|= {:e}, in-sample err= {:e}"
29         .format(hsweep+1, f.neval, itci, abs(itci - i5), tci.pivotError[-1]))
```

Listing 1: Python code to numerically compute the integral $I^{(\mathcal{N}=5)}$ (Eq. (64)) using the `xfac` package with `TensorCI1`. The script performs 14 half-sweeps using continuous TCI on a 15 point Gauss–Kronrod grid. For each half-sweep (`hsweep`), the number of function evaluations (`neval`), the approximate integral value (`itci`), the absolute error with respect to the exact integral value (`i5`) from Eq. (65) and the in-sample error (`insample err`) is printed. These values are shown in Figs. 4(a-c).

The Python script to perform the integration numerically using the Python bindings of `xfac` (package `xfacpy`) is shown in code Listing 1; see Listing 11 for an equivalent Julia code using `TCI.jl`. Both codes can be trivially adapted to compute the integral of any function which is known explicitly by just modifying the definition of $f(\mathbf{x})$.

In the Python code, lines 1 and 2 import the packages `xfacpy` and the `log` function (needed for comparison with Eq. (65)). Lines 7–9 define the user-supplied function f; line 8 defines an (optional) attribute of f, `neval`, counting the number of times the integrand is called; line 9 defines the integrand. Here x is a list of floats or a numpy array. For each argument x_ℓ, the user specifies a grid $\{x_\ell(\sigma_\ell)\}$ of d_ℓ quadrature nodes, enumerated by an index σ_ℓ, and an associated grid of quadrature weights $\{w_\ell(\sigma_\ell)\}$ (cf. Sec. 5.1). Here, we use the nodes and weights of the Gauss–Kronrod quadrature, with $d_\ell = 15$ for all ℓ. For convenience, the Gauss–Kronrod quadrature is included in `xfac` so that the GK15 function in line 18 returns two lists, `xell` and `well`, containing the quadrature nodes $\{x_\ell(\sigma_\ell)\}$ and weights $\{w_\ell(\sigma_\ell)\}$, respectively (chosen the same for all ℓ).

The `CTensorCI()` object created in line 21 is the basic object used to perform TCI on a continuous function, discretized as $F_{\boldsymbol{\sigma}} = f\big(\mathbf{x}(\boldsymbol{\sigma})\big)$. This class performs the factorization in accumulative mode. Note that `CTensorCI()` is a thin wrapper over the corresponding discrete class `TensorCI()` that creates $F_{\boldsymbol{\sigma}}$ from $f(\mathbf{x})$ and the grids $\mathbf{x}_\ell(\boldsymbol{\sigma}_\ell)$. To instantiate the class, two arguments must be provided: the function `f`, and the grid on which the function will be called, `[xell] * N`. For $\mathcal{N} = 5$, the latter is equivalent to `[xell, xell, xell, xell, xell]`, i.e. five copies of the GK15 grid (a list of list of points).

The loop in lines 24–29 performs a series of half-sweeps, alternating left-to-right and right-to-left, 14 in total (i.e. 7 full sweeps), to iteratively improve the TCI approximation $\widetilde{F}_{\boldsymbol{\sigma}}$ of the tensor $F_{\boldsymbol{\sigma}}$. In line 25, `tci.iterate()` performs one half-sweep, and in line 27, `tci.get_TensorTrain().sum([well] * N)` calculates the integral according to Eq. (62). Finally, lines 28 and 29 print the results: the number of half-sweeps, `hsweep`; the number of calls to f, `neval`; the calculated value of the integral, `itci`; its error with respect to the exact calculation, $|I^{(\mathcal{N})} - \widetilde{I}^{(\mathcal{N})}|$; and the "in-sample error", `in-sample err`, defined as the maximum difference

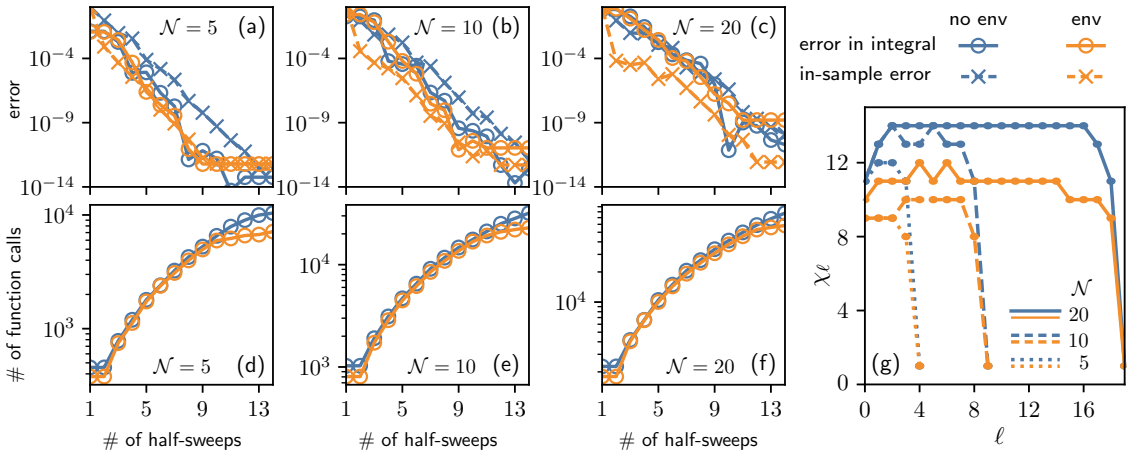


Figure 4: Performance metrics for the TCI computation of the $\mathcal{N}$-dimensional integral $I^{(\mathcal{N})} = \int d^{\mathcal{N}}\mathbf{x} f(\mathbf{x})$ of Eq. (64) using the natural tensor representation (2), for $\mathcal{N} = 5, 10, 20$. (a–c) The relative error for the integral $|1 - I^{(\mathcal{N})}/\widetilde{I}^{(\mathcal{N})}|$ (solid lines with circles), the maximum (over all sampled pivots) of the relative in-sample error $|1 - F_{\boldsymbol{\sigma}}/\widetilde{F}_{\boldsymbol{\sigma}}|_\infty$ (dashed lines with crosses), and (d–f) the number of function calls, all plotted versus the number of half-sweeps. The TCI computation of $\widetilde{I}^{(\mathcal{N})}$ has been performed on a 15-point Gauss–Kronrod grid (i.e. $d_\ell = 15$), either in the no environment mode ("no env", blue) or in the environment mode ("env", orange). (g) The final bond dimension χ_ℓ plotted vs. $\ell \in [1, \mathcal{N}]$, for $\mathcal{N} = 5, 10, 20$. The growth of χ_ℓ with increasing ℓ or $\mathcal{N} - \ell$ flattens off at rather small values of $\chi = \max\{\chi_\ell\}$, indicating that the function $f(\mathbf{x})$ is strongly compressible. [Code: Listing 1 (Python), 11 (Julia)]

$|F_{\boldsymbol{\sigma}} - \widetilde{F}_{\boldsymbol{\sigma}}|$ during the half-sweep (a "training set error", albeit a very conservative one because the algorithm is actively looking for points with large errors). The code above performs the bare variant (no environment) of the factorization of $F_{\boldsymbol{\sigma}}$. For comparison, we have also computed the factorisation in environment mode (see Sec. B.1.1 for the corresponding syntax).

Figure 4 shows the two errors (upper panel) and number of function calls (lower panel) as a function of the number of half-sweeps for $\mathcal{N} = 5, 10$, and 20. The convergence of the integral is very fast and depends only weakly on the number of dimensions. It turns out that, in this example, the environment mode (orange) does not bring much advantage over the bare mode (blue). To highlight the strength of TCI we note that for $\mathcal{N} = 5$ (or $\mathcal{N} = 20$) the 14 half-sweeps needed to reach an absolute error below 10^{-10} (or 10^{-8}) required roughly 10^4 (or 10^5) function calls, hence the ratio of the number of sampled points to all points of $F_{\boldsymbol{\sigma}}$ was only $10^4/15^5 \approx 10^{-2}$ (or $10^5/15^{20} \approx 10^{-19}$). In general, if the rank of the MPS unfolding of the integrand remains roughly constant as the number of dimensions increases, then the gain in favor of TCI increases exponentially.

Finally, let us state that the method presented above only works if the chosen quadrature model (e.g. the Gauss–Kronrod quadrature) is suitable for the integrand in question. A variant of this method using the quantics representation is presented in section 6.3.2.

5.3 Example of computation of partition functions

Our second example is very similar to the previous one except that we now consider an object that is already a (discrete) tensor, without any need to perform a discretization. This example was implemented in C++, and the code used to generate all data can be found in Listing 10 in App. B.2.1.

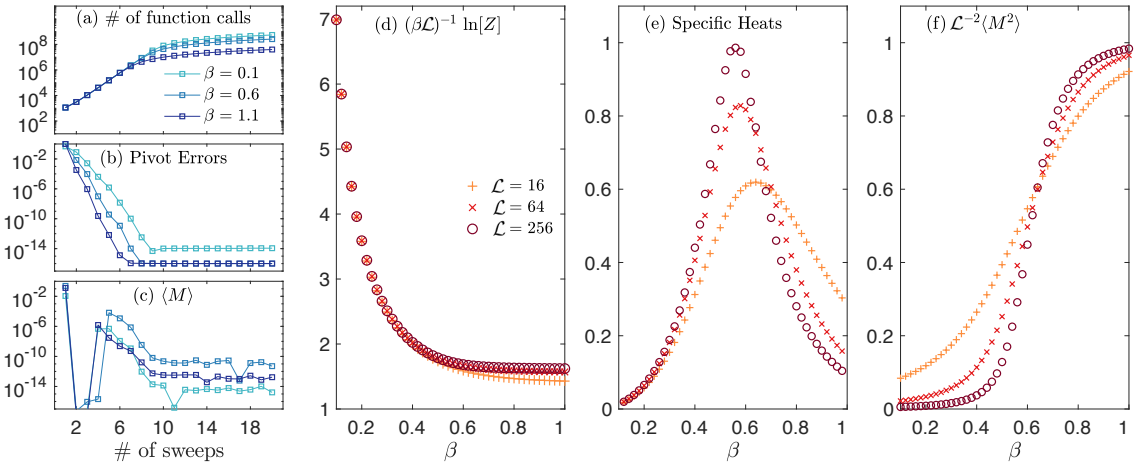


Figure 5: Unfolding the Boltzmann distribution function for the inverse square Ising chain via 2-site TCI in reset mode. The first three panels show the evolution of (a) the number of function calls, (b) pivot errors, and (c) magnetization with TCI full-sweeps for $\mathcal{L} = 64$ for $\beta = 0.1, 0.6$ and 1.1. (d) The free energy density, (e) specific heat, and (f) the second-order moment, for $\mathcal{L} = 16$, 64 and 256, computed for temperatures in the vicinity of the phase transition at $\beta_c \approx 0.62$. [Code: Listing 10 (C++)]

We consider the calculation of a classical partition function of the form $Z = \sum_{\boldsymbol{\sigma}} W_{\boldsymbol{\sigma}}$, where $W_{\boldsymbol{\sigma}} = e^{-\beta E_{\sigma}}$ and $E_{\boldsymbol{\sigma}}$ are the Boltzmann weight and energy, respectively, of a configuration $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_{\mathcal{L}})$ and $\beta = 1/T$ is the inverse temperature of the system. Once the Boltzmann weight has been put in TCI form, $W_{\boldsymbol{\sigma}} \simeq \widetilde{W}_{\boldsymbol{\sigma}} = \prod_{\ell=1}^{\mathcal{L}} M_{\ell}^{\sigma_\ell}$, the partition function can be expressed in factorized form, allowing its evaluation in polynomial time:

$$Z = \sum_{\boldsymbol{\sigma}} W_{\boldsymbol{\sigma}} \approx \sum_{\boldsymbol{\sigma}} \widetilde{W}_{\boldsymbol{\sigma}} = \sum_{\boldsymbol{\sigma}} \prod_{\ell=1}^{\mathcal{L}} M_{\ell}^{\sigma_\ell} = \prod_{\ell=1}^{\mathcal{L}} \sum_{\sigma_\ell} M_{\ell}^{\sigma_\ell} . \tag{66}$$

This direct access to Z stands in contrast to Monte Carlo approaches: these typically evaluate ratios of sums, giving easy access only to observables such as magnetization but not directly to the partition function itself. From Z, one can calculate the free energy per site, $F = (\beta\mathcal{L})^{-1} \ln Z$, and the specific heat, $C = \beta^2 \frac{\partial \ln Z}{\partial \beta^2}$ (evaluated through finite differences). Other quantities can also be calculated directly using appropriate weights.

Our example is a ferromagnetic Ising chain with a long-range interaction decaying as the inverse square of the distance. The energy of a configuration reads

$$E_{\boldsymbol{\sigma}} = -\sum_{\ell<\ell'} J_{\ell\ell'} \sigma_\ell \sigma_{\ell'} , \tag{67}$$

where $\sigma_\ell = \pm 1$ is a classical spin variable at site ℓ and $J_{\ell\ell'} = |\ell - \ell'|^{-2}$ the coupling constant between sites ℓ and ℓ'. This system is sufficiently complex to display a Kosterlitz-Thouless transition [55–57] at $\beta_c \approx 0.62$. Beyond the free energy, we also calculate the magnetization $M = \sum_{\ell=1}^{\mathcal{L}} \sigma_\ell / \mathcal{L}$ and its variance, using suitably modified versions of Eq. (66).

In Figs. 5(a–c), we first inspect the accuracy of the TCI at three different temperatures with $\mathcal{L} = 64$. Fig. 5(a) shows the accumulated number of function calls to the Boltzmannn weight $W_{\boldsymbol{\sigma}}$ over several sweeps. The total number of function calls initially grows exponentially, then the growth slows down significantly once the TCI's pivot error [Fig. 5(b)] approaches convergence. In Fig. 5(c), we see that irrespective of β, the average of the on-site magnetization reduces to almost zero (smaller than 10^{-8}) when the TCI's pivot error is sufficiently small.

This is due to symmetry since we did not use a (small) magnetic field to break the global Z_2 symmetry of the problem. To preserve this symmetry during TCI, we start the algorithm with two global pivots: $\boldsymbol{\sigma} = (1, 1, \ldots, 1)$ and $(-1, -1, \ldots, -1)$. This is very important at low temperature. Indeed, if we use only a single global pivot, then the pivot exploration gets stuck in the corresponding sector and we obtain the same result as if we *had* broken the symmetry with a small magnetic field. Even though the initial pivots correspond to fully polarized configurations ($\beta \to \infty$), TCI converges well at all temperatures, including in the paramagnetic phase. This is a indication of the robustness of the algorithm.

Figures 5(d–f) compare physical observables, such as the free energy, the specific heat, and the second magnetic moment, for $\mathcal{L} = 16$, 64 and 256. The smoothness of the free energy curve versus β [Fig. 5(d)] rules out the possibility of a first-order transition. Yet a phase transition is clearly seen in Fig. 5(e), as the specific heat develops an increasingly sharp peak when increasing the system size. Figure 5(f), showing the second magnetic moment, likewise indicates that a phase transition occurs at $\beta \approx 0.62$, where the three sets of data points for different system sizes intersect.

6 Application: Quantics representation of functions

When working with functions $f(\mathbf{x})$ for which a very high resolution of the variables $\mathbf{x}$ is desired, e.g. functions having structures with widely different length scales, using the *quantics tensor representation* [18,19] can be advantageous. It achieves exponential resolution by representing the function variables $\mathbf{x}$ through binary digits $\boldsymbol{\sigma}$. The resulting binary representation of the function can be viewed as a tensor, $F_{\boldsymbol{\sigma}} = f(\mathbf{x}(\boldsymbol{\sigma}))$. Many functions are represented by a low-rank tensor, including some functions involving vastly different scales [15,21,49]. This section discusses various applications of quantics TCI.

6.1 Definition

We begin by discussing the quantics representation of a function of one variable, $f(x)$. The variable is rescaled such that $x \in [0, 1)$ and discretized on a uniform grid $x(m) = m/M$, with $M = 2^{\mathcal{R}}$ and $m = 0, 1, \ldots, M-1$. We express the grid index m in binary form using $\mathcal{R}$ bits $\sigma_r \in \{0, 1\}$ as follows (the second expression is standard binary notation)

$$m(\sigma_1, \ldots, \sigma_{\mathcal{R}}) = (\sigma_1 \sigma_2 \cdots \sigma_{\mathcal{R}})_2 \equiv \sum_{r=1}^{\mathcal{R}} \sigma_r 2^{\mathcal{R}-r} \,. \tag{68}$$

We define $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_{\mathcal{R}})$ and $x(\boldsymbol{\sigma}) = x(m(\boldsymbol{\sigma}))$. Bit σ_r now resolves x at the scale 2^{-r}. Thus, the discretized function f is a tensor $F_{\boldsymbol{\sigma}} = f(x(\boldsymbol{\sigma}))$, the *quantics* representation of f. It has $\mathcal{L} = \mathcal{R}$ indices, each of dimension $d = 2$.

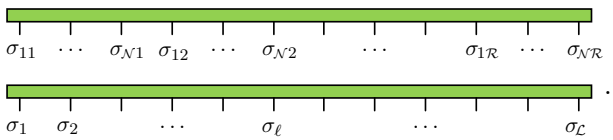
For a function of $\mathcal{N}$ variables, $f(\mathbf{x}) = f(x_1, \ldots, x_{\mathcal{N}})$, we rescale and discretize each variable as $x_n(m_n) = m_n/M = m_n/2^{\mathcal{R}}$, then express m_n through $\mathcal{R}$ bits $\sigma_{nr} \in \{0, 1\}$ as

$$m_n(\sigma_{n1}, \ldots, \sigma_{n\mathcal{R}}) = (\sigma_{n1} \sigma_{n2} \cdots \sigma_{n\mathcal{R}})_2 = \sum_{r=1}^{\mathcal{R}} \sigma_{nr} 2^{\mathcal{R}-r} \,. \tag{69}$$

The vector $\mathbf{x}$ is represented by a tuple of $\mathcal{L} = \mathcal{N}\mathcal{R}$ bits, where bit σ_{nr} resolves x_n at the scale 2^{-r}. The rank of the tensor train $\widetilde{F}_{\boldsymbol{\sigma}}$ obtained by unfolding $F_{\boldsymbol{\sigma}}$ can strongly depend on the way we order the different bits. In the *interleaved* quantics representation, we group all the bits that address the same scale together and relabel the bits as $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_{\mathcal{L}})$, with $\sigma_{\ell(n,r)} = \sigma_{nr}$

and $\ell = n+(r-1)\mathcal{N} = 1,\ldots,\mathcal{L}$, such that

$$F_{\boldsymbol{\sigma}} = f\big(\mathbf{x}(\boldsymbol{\sigma})\big) =$$



(70)

If the variables at the same scale are strongly entangled, which is the case in many physical applications, using the interleaved quantics representation can lead to a more compressible tensor [15, 18, 19, 21]. An alternative is the *fused quantics* representation, $F_{\tilde{\boldsymbol{\sigma}}} = f\big(\mathbf{x}(\tilde{\boldsymbol{\sigma}})\big)$, where we "fuse" all bits for scale 2^{-r} into a single variable

$$\tilde{\sigma}_r = (\sigma_{\mathcal{N}r}\cdots\sigma_{2r}\sigma_{1r})_2 = \sum_{n=1}^{\mathcal{N}} 2^{n-1}\sigma_{nr}\,, \tag{71}$$

taking the values $0,\ldots,2^{\mathcal{N}}-1$, and arrange these variables as $\tilde{\boldsymbol{\sigma}} = (\tilde{\sigma}_1,\ldots,\tilde{\sigma}_{\mathcal{R}})$. One can also group together all bits addressing a given variable x_n, as done in the natural representation.

Once a quantics representation F of f has been defined, TCI can be applied to F to obtain a tensor train $\widetilde{F}$ interpolating f with exponential resolution. We dub this algorithm *quantics TCI* (QTCI), and the resulting tensor train a *quantics tensor train* (QTT) [18–20].

Some simple analytic functions are approximated well as a QTT with $\chi < 10$. For instance, a pure exponential, $f(x) = e^{\lambda x}$, has $\chi = 1$, since its quantics tensor factorizes completely, $F_{\boldsymbol{\sigma}} = \prod_{r=1}^{\mathcal{R}} e^{\lambda\sigma_r 2^{\mathcal{R}-r}}$. Similarly, sine and cosine functions have $\chi = 2$, since they can be expressed as sums of two exponentials, i.e. sums of two rank-1 tensors. Some discontinuous functions likewise have low-rank in quantics representations, such as the Dirac delta ($\chi = 1$) and Heaviside step function ($\chi = 2$) [19]. By contrast, random noise is incompressible and leads to $\chi \sim d^{\mathcal{L}/2}$. More generally, if a function has low quantics rank χ, the sites representing different scales are not strongly "entangled". In this sense, the quantics rank of a function quantifies the degree of scale separation inherent in the function [15, 21].

An interesting example of low-rank analytic functions of two variables is the Kronecker delta function $f(m_1, m_2) = \delta_{m_1 m_2}$ defined on a discrete 2D grid. Its matrix representation, the $2^{\mathcal{R}}\times 2^{\mathcal{R}}$ unit matrix, is incompressible (in the sense of SVD) because all its singular values are 1. In the quantics representation, $f(m_1, m_2) = \delta_{\sigma_{11}\sigma_{21}}\cdots\delta_{\sigma_{1r}\sigma_{2r}}\cdots\delta_{\sigma_{1\mathcal{R}},\sigma_{2\mathcal{R}}}$, which can be regarded as a rank-1 MPS by fusing σ_{1r} and σ_{2r}.

Other examples for functions of multiple variables that can be approximated as a low-rank QTT are multivariate analogues of the 1D examples above, with $\mathbf{x}\in\mathbb{R}^{\mathcal{N}}$: a single exponential $f(\mathbf{x}) = \exp(\mathbf{v}\cdot\mathbf{x})$ with arbitrary $\mathbf{v}$ has bond dimension $\chi = 1$; a Dirac delta $\delta(\mathbf{x})$ reduces to the Kronecker delta above and therefore has bond dimension $\chi = 1$ as well; a step function $f(\mathbf{x}) = \theta(\mathbf{v}\cdot\mathbf{x}-\mathbf{b})$ for given $\mathbf{v}$ and $\mathbf{b}$ has bond dimension $\chi = 2$. In all examples mentioned here, the small bond dimension is due to separability of length scales. An example where length scales are not separable is the function $f(\mathbf{x}) = \theta(1-\|\mathbf{x}\|_2)$, which is equal to 1 inside the unit sphere and 0 outside. Since the surface of the sphere is curved, the maximum bond dimension, $\chi_{\max}$, needed to represent this function with a QTT will depend on the resolution with which the surface is resolved, increasing as the resolution is refined. This is illustrated in Fig. 6 for the case $\mathcal{N} = 2$.

6.2 Operating on quantics tensor trains

Given a function represented by a quantics tensor train, various operations on these functions can be performed within the tensor train form. In the following, we describe how to calculate integrals, convolutions and symmetry transforms within the quantics representation; quantics

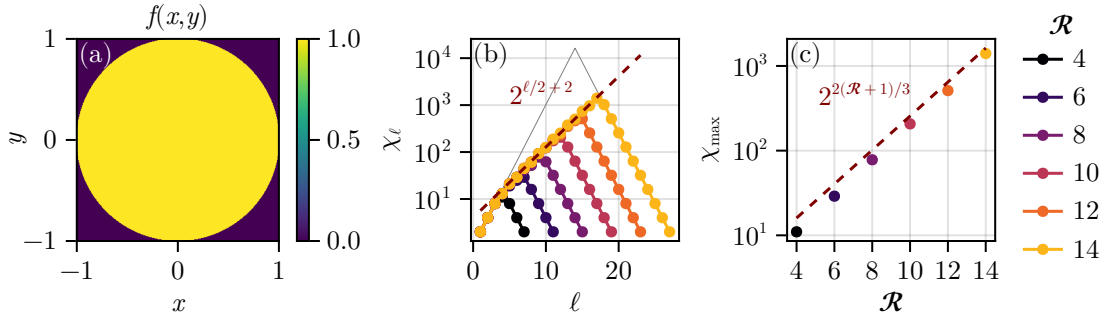


Figure 6: (a) The function $f(\mathbf{x}) = \theta(1 - \|\mathbf{x}\|_2)$, in $\mathcal{N} = 2$ dimensions. (b) The bond dimension χ_ℓ of a QTT representation of f with interleaved index ordering, plotted for several values of $\mathcal{R}$. Along the chain, the bond dimension scales as $\chi_\ell \sim 2^{\ell/2}$. Intuitively, this is because each additional pair of bits σ_{1r}, σ_{2r} doubles the number of points close to the circle, which are those that contain additional information. (c) The maximal bond dimension, $\chi_{\max}$, increases exponentially with $\mathcal{R}$, as $\chi_{\max} \approx 2^{2(\mathcal{R}+1)/3}$. This behavior is independent of the specified tolerance, because the step function changes abruptly. If the step function is broadened, the maximum bond dimension decreases significantly, in a manner depending on the tolerance.

Fourier transforms are described in detail in Sec. 6.2. In addition, the methods for element-wise operations and addition of tensor trains that have already been introduced in Sec. 4.7 work just as well here. These basic 'building blocks' can be combined to formulate more complicated algorithms entirely within the quantics tensor train form.

Integrals are approximated as Riemann sums, then factorized over the quantics bits as

$$\int_{[0,1]^\mathcal{N}} d^\mathcal{N}\mathbf{x} f(\mathbf{x}) \approx \frac{1}{2^\mathcal{L}} \sum_{\boldsymbol{\sigma}} f\big(\mathbf{x}(\boldsymbol{\sigma})\big) = \frac{1}{2^\mathcal{L}} \sum_{\boldsymbol{\sigma}} F_{\boldsymbol{\sigma}} \approx \frac{1}{2^\mathcal{L}} \sum_{\boldsymbol{\sigma}} \widetilde{F}_{\boldsymbol{\sigma}} = \frac{1}{2^\mathcal{L}} \prod_{\ell=1}^{\mathcal{L}} \Big[\sum_{\sigma_\ell=1}^{2} M_\ell^{\sigma_\ell} \Big], \tag{72}$$

where $1/2^\mathcal{L}$ is the integration volume element. Since the number of discretization points is exponential in $\mathcal{L}$, the discretization error of this integral decreases as $O(1/2^\mathcal{L})$, whereas the cost of the factorized sum is $\mathcal{O}(\chi^2 d\mathcal{L})$, i.e. linear in $\mathcal{L}$.

Matrix products of the form $f(\mathbf{x}, \mathbf{z}) = \int_D d^\mathcal{N}\mathbf{y} g(\mathbf{x}, \mathbf{y}) h(\mathbf{y}, \mathbf{z})$ can be performed as follows. We use quantics representations for each of the variables $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$, e.g. $\mathbf{x} = \mathbf{x}(\boldsymbol{\sigma}_x)$ with $\boldsymbol{\sigma}_x = (\sigma_{1x}, \ldots, \sigma_{\mathcal{L}x})$ and $\mathcal{L} = \mathcal{N}\mathcal{R}$. We unfold the tensors for g and h as MPOs,

$$\widetilde{G}_{\boldsymbol{\sigma}_x\boldsymbol{\sigma}_y} = \quad , \qquad \widetilde{H}_{\boldsymbol{\sigma}_y\boldsymbol{\sigma}_z} = \quad , \tag{73}$$

with indices at matching scales, $(\sigma_{\ell x}, \sigma_{\ell y})$ or $(\sigma_{\ell y}, \sigma_{\ell z})$, assigned to the same site ℓ. We then approximate the integral $\int d^\mathcal{N}\mathbf{y}$ by a factorized sum over each $\sigma_{\ell y}$, cf. (72), $f(\mathbf{x}, \mathbf{y}) \approx 2^{-\mathcal{L}} \sum_{\boldsymbol{\sigma}_y} \widetilde{G}_{\boldsymbol{\sigma}_x\boldsymbol{\sigma}_y} \widetilde{H}_{\boldsymbol{\sigma}_y\boldsymbol{\sigma}_z}$, which can be computed and compressed in several ways, see Sec. 4.7.

Quantics Fourier transform can be performed using a simple MPO-MPS contraction, where the MPO is of surprisingly low rank ($\chi \approx 11$ for machine precision in one dimension) [21, 58]. This means that taking the Fourier transform of a function that has a low-rank quantics tensor train can be done exponentially faster than with FFT. Calculating $\hat{f}(\mathbf{k}) = \int d\mathbf{x} f(\mathbf{x}) e^{-i\mathbf{k}\cdot\mathbf{x}}$

on a quantics tensor train representing $f(\mathbf{x})$ is equivalent to the quantum Fourier transform algorithm ubiquitous in quantum computing [18].

Consider a discrete function $f_m \in \mathbb{C}^M$, e.g. the discretization, $f_m = f(x(m))$, of a one-dimensional function $f(x)$ on a grid $x(m)$. Its discrete Fourier transform (DFT) is

$$\hat{f}_k = \sum_{m=0}^{M-1} T_{km} f_m \,, \qquad T_{km} = \tfrac{1}{\sqrt{M}} e^{-i2\pi k\cdot m/M} \,. \tag{74}$$

For a quantics grid, $M = 2^{\mathcal{R}}$ is exponentially large and the DFT exponentially expensive to evaluate. We seek a quantics tensor train representing T, because then $\hat{f} = Tf$ can be computed by simply contracting the tensor trains for T and f and recompressing [18,19,21].

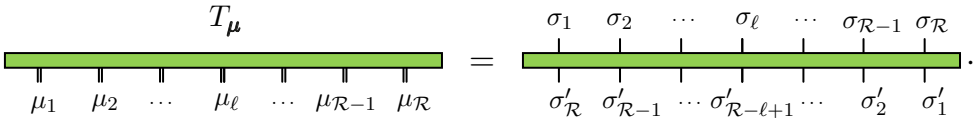
We start by expressing m and k in their quantics form

$$m(\boldsymbol{\sigma}) = (\sigma_1\sigma_2\cdots\sigma_{\mathcal{R}})_2 = \sum_{\ell=1}^{\mathcal{R}} \sigma_\ell 2^{\mathcal{R}-\ell} \,, \qquad k(\boldsymbol{\sigma}') = (\sigma'_1\sigma'_2\cdots\sigma'_{\mathcal{R}})_2 = \sum_{\ell'=1}^{\mathcal{R}} \sigma_{\ell'} 2^{\mathcal{R}-\ell'} \,. \tag{75}$$

Then, T has the quantics representation

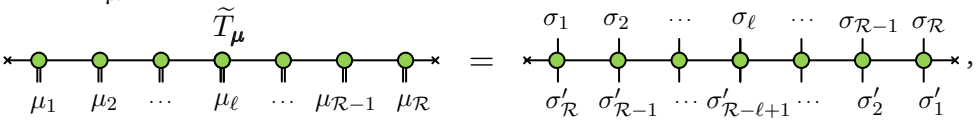
$$T_{\boldsymbol{\mu}} = T_{\boldsymbol{\sigma}'\boldsymbol{\sigma}} = T_{k(\boldsymbol{\sigma}')m(\boldsymbol{\sigma})} = \tfrac{1}{\sqrt{M}} \exp\Big[-i2\pi \sum_{\ell\ell'} 2^{\mathcal{R}-\ell'-\ell} \sigma'_{\ell'}\sigma_\ell\Big], \tag{76}$$

where we introduced the fused index $\boldsymbol{\mu} = (\mu_1, \ldots \mu_{\mathcal{R}})$, with $\mu_\ell = (\sigma'_{\mathcal{R}-\ell+1}, \sigma_\ell)$. We thereby arrange $\boldsymbol{\sigma}'$ and $\boldsymbol{\sigma}$ indices in *scale-reversed* order [21], so that $\sigma'_{\mathcal{R}-\ell+1}$, describing the scale $2^{\ell-1}$ in the k domain, is fused with σ_ℓ, describing the scale $2^{\mathcal{R}-\ell}$ in the m domain, in accordance with Fourier reciprocity (small k scales match large m scales and vice versa):



(77a)

The tensor $T_{\boldsymbol{\mu}}$ turns out to have a remarkably *low rank* [21,58]: when unfolded as a MPO $\widetilde{T}_{\boldsymbol{\mu}}$,



(77b)

a rank of $\chi = 11$ suffices to yield machine precision, i.e. errors $|T_{\boldsymbol{\mu}} - \widetilde{T}_{\boldsymbol{\mu}}|_\infty / |T_{\boldsymbol{\mu}}|_\infty < 10^{-10}$, irrespective of $\mathcal{R}$ [21,58]. By contrast, if a scale-reversed order is not used (i.e., $\mu_\ell = (\sigma'_\ell, \sigma_\ell)$), the resulting tensor $T_{\boldsymbol{\mu}}$ has exponentially large rank [59]. An intuitive explanation for the scale-reversed order is given in Appendix A.5, which also verifies through numerical experiment that the small-rank representation is found by TCI.

It follows that for a 1D function with rank χ' in quantics representation, the DFT can be obtained in $O(\chi^2\chi'^2\mathcal{R}) = O(\chi^2\chi'^2 \log M)$ operations, where $M = 2^{\mathcal{R}}$ is the number of points in the grid. This is exponentially faster than the $O(M \log M)$ of the fast Fourier transform [21, 58, 60].

6.3 Example: High-resolution compression of functions

In this section we illustrate the use of quantics TCI for representing functions in 1, 2 and 3 dimensions, and for computing multi-dimensional integrals.

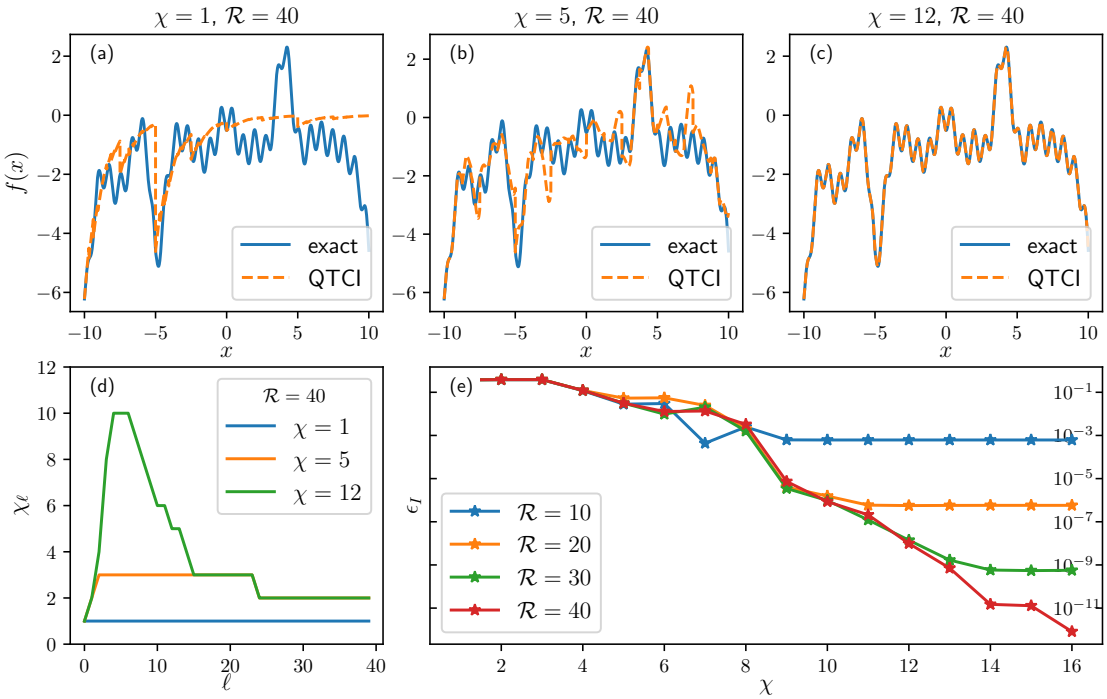


Figure 7: (a-c) The function $f(x)$ of Eq. (78) (solid blue) and its quantics representation (orange dashed) with $\mathcal{R} = 40$ for $\chi = 1$, 5 and 12. Although the TCI is performed on $2^{40} \approx 10^{12}$ points, only a small fraction are actually shown in the plot. (d) Bond dimension χ_ℓ as a function of ℓ, for $\chi = 1, 5, 12$. (e) Error on the integral $I = \int_{-10}^{10} \mathrm{d}x f(x)$ calculated from its QTCI approximation, $\widetilde{I}$. The plot shows the relative error $\epsilon_I = |\widetilde{I}/I - 1|$, plotted versus the rank χ of the QTCI approximation, for $\mathcal{R} = 10$, 20, 30, 40. [Code: Listing 2 (Python), 3 (Julia)]

6.3.1 Oscillating functions in 1, 2 and 3 dimensions

1d oscillating function As a first example, we consider a simple function with large oscillations:

$$f(x) = \mathrm{sinc}(x) + 3e^{-0.3(x-4)^2}\,\mathrm{sinc}(x-4) - \cos(4x)^2 - 2\,\mathrm{sinc}(x+10)e^{-0.6(x+9)} + 4\cos(2x)e^{-|x+5|} + \frac{6}{x-11} + \sqrt{(|x|)}\arctan(x/15)\,, \tag{78}$$

where $\mathrm{sinc}(x) = \sin x/x$ is the sinus cardinal and $x \in [-10, 10]$.

The code of Listing 2 discretizes the function on a quantics grid of 2^{40} points $\{x(\boldsymbol{\sigma})\}$, defines the tensor $F_{\boldsymbol{\sigma}} = f\big(x(\boldsymbol{\sigma})\big)$ and uses `xfac` to TCI it, $\widetilde{F}_{\boldsymbol{\sigma}} \approx F_{\boldsymbol{\sigma}}$. Listing 3 shows `TCI.jl` code performing the same task. Figs. 7(a–c) show the resulting QTCI approximation: it converges very quickly with increasing χ. Fig. 7(d) shows the bond dimension χ_ℓ as a function of ℓ. It remains small (≤ 10), hence the function is strongly compressible. Figure 7(e) shows that the integral $I = \int_{-10}^{+10} \mathrm{d}x f(x)$ converges rapidly with increasing χ even though the function is highly oscillatory.

```
import xfacpy
import numpy as np

# Grid parameters
R = 40                          # Number of bits <@$\cR$@>
M = 2**R                        # Number of grid points <@$M$@>
```

```
xmin, xmax = -10.0, +10.0   # Domain of function <@$f$@>


def m_to_sigma(m):          # Convert grid index <@$m$@> to quantics multi-index <@$\bsigma(m)$@>
    return [int(k) for k in np.binary_repr(m, width=R)]


def sigma_to_x(sigma):      # Convert quantics multi-index <@$\bsigma$@> to grid point <@$x(\bsigma)$@>
    ind = int(''.join(map(str, sigma)), 2)
    return xmin + (xmax-xmin)*ind/M


def f(x):                   # Function of interest <@$f(x)$@>
    return (np.sinc(x)+3*np.exp(-0.3*(x-4)**2)*np.sinc(x-4)-np.cos(4*x)**2 -
            2*np.sinc(x+10)*np.exp(-0.6*(x+9))+4*np.cos(2*x)*np.exp(-abs(x+5)) +
            6*1/(x-11)+abs(x)**0.5*np.arctan(x/15))


def f_tensor(sigma):        # Quantics tensor <@$F_\bsigma$@>
    return f(sigma_to_x(sigma))


# Set first pivot to <@$\bar\bsigma=(0, \ldots, 0)$@> and initialize TCI <@$\tF_\bsigma$@>
p = xfacpy.TensorCI1Param()
p.pivot1 = [0 for ind in range(R)]
f_tci = xfacpy.TensorCI1(f_tensor, [2]*R, p)

# Optimize <@$\tF_\bsigma$@>
for sweep in range(12):
    f_tci.iterate()         # Perform a half sweep

f_tt = f_tci.get_TensorTrain()  # Obtain the TT <@$M_1 M_2 \ldots M_\scR$@>
# Print a table to compare <@$f(x)$@> and <@$\tF_\bsigma$@> on some regularly spaced points
print("x\t f(x)\t f_tt(x)")
for m in range(0, M, 2**(R-5)):
    sigma = m_to_sigma(m)
    x = xmin + (xmax-xmin)*m/M
    print(f"{x}\t{f(x)}\t{f_tt.eval(sigma)}")
```

Listing 2: Python code using `xfac` to compute construct a quantics tensor train for the function $f(x)$ of Eq. (78), shown in Fig. 7, using 2^{40} grid points. Note that this code could also have used pre-defined functions that are part of `xfac` to generate quantics grids $x(\boldsymbol{\sigma})$ and convert between x, m, and $\boldsymbol{\sigma}$, see App. B.1.3.

2d oscillating function The above construction generalizes straightforwardly to more than one dimension. Let us consider the following simple 2d function with features at vastly different scales:

$$f(x,y) = 1 + e^{-0.4\left(x^2+y^2\right)} + \sin(xy)\, e^{-x^2} + \cos(3xy)\, e^{-y^2} + \cos(x+y) \\ + 0.05\cos\left[10^2\cdot(2x-4y)\right] + 5\cdot 10^{-4}\cos\left[10^3\cdot(-2x+7y)\right] + 10^{-5}\cos\left(2\cdot 10^8 x\right). \tag{79}$$

We use a quantics unfolding of $f(x,y)$ with $\mathcal{R} = 40$, which discretizes f on a $10^{12}\times 10^{12}$ grid. The corresponding quantics tensor $F_{\boldsymbol{\sigma}}$ has $\mathcal{L} = 2\mathcal{R}$ indices, interleaved so that even indices $\sigma_{2\ell}$ encode x and odd indices $\sigma_{2\ell+1}$ encode y. A tensor train approximation is then obtained using standard TCI, which yields an efficient low-rank representation that rapidly converges, as shown in Fig. 8 (a–d). At rank $\chi \approx 110$, the MPS becomes a numerically exact (within machine precision) representation of the original function at all scales. It requires only 10^5 numbers ($\sim$ 1 MB of RAM), which is trivial to store in memory, and 19 orders of magnitude smaller than needed for a naive regular grid ($\sim 10^{13}$ TB of RAM). Furthermore,

```
using QuanticsTCI
import QuanticsGrids as QG

R = 40                                          # Number of bits <@$\cR$@>
M = 2^R                                         # Number of discretization points <@$M$@>
xgrid = QG.DiscretizedGrid{1}(R, -10, 10)       # Discretization grid <@$x(\bsigma)$@>

function f(x)                                   # Function of interest <@$f(x)$@>
    return (
        sinc(x) + 3 * exp(-0.3 * (x - 4)^2) * sinc(x - 4) - cos(4 * x)^2 -
        2 * sinc(x + 10) * exp(-0.6 * (x + 9)) + 4 * cos(2 * x) * exp(-abs(x + 5)) +
        6 * 1 / (x - 11) + sqrt(abs(x)) * atan(x / 15))
end

# Construct and optimize quantics TCI <@$\tF_\bsigma$@>
f_tci, ranks, errors = quanticscrossinterpolate(Float64, f, xgrid; maxbonddim=12)
# Print a table to compare <@$f(x)$@> and <@$\tF_\bsigma$@> on some regularly spaced
    points
println("x\t f(x)\t\t\t f_tt(x)")
for m in 1:2^(R-5):M
    x = QG.grididx_to_origcoord(xgrid, m)
    println("$x\t$(f(x))\t$(f_tci(m))")
end
```

Listing 3: Julia code using `TCI.jl` to construct a quantics tensor train for $f(x)$ of Eq. (78), plotted in Fig. 7. The function `quanticscrossinterpolate` includes code to convert f to quantics form, see Sec. 8.3. The `xgrid` object constructed on line 6 is a lazy object that does not create an exponentially large object.

it can be manipulated exponentially faster than for the regular grid, including most common operations such as Fourier transform, convolution or integration.

3d integral Figure 9 shows the last example of this series: the computation of the 3D integral $I = \int_{\mathbb{R}^3} d^3\mathbf{x} e^{-\sqrt{x^2+y^2+z^2}}$ using the quantics representation. TCI in both accumulative and reset mode converges exponentially fast towards the exact integral $I = 8\pi$, almost reaching machine precision, an indication of excellent numerical stability.

6.3.2 Quantics for multi-dimensional integration

Let us return in this section to the example of the multi-dimensional integral from Sec. 5.2. In the initial approach, a quadrature has been choosen in order to factorize the function on the quadrature grid, and then, in a second step, to perform the one-dimensional integrations. Here, we consider the interleaved quantics representation described in Sec. 6.1, with $\mathcal{L} = \mathcal{N}\mathcal{R}$ legs of dimension $d = 2$. After obtaining the QTT from TCI, the integral can be evaluated efficiently by a factorized sum over the MPS tensors, as shown in Eq. (72). This corresponds to a Riemann sum with exponentially many discretization points. The results are shown in Fig. 10. In this example, we observe a fast convergence of the results. Note that using the fused instead of interleaved representation here would lead to $d = 2^{\mathcal{N}}$, which quickly becomes prohibitive for large $\mathcal{N}$.

Listing 8 in App. B.1.3 contains the python code (using `xfac`) yielding the results shownin Fig. 10. The code is very similar to Listing 1, but replaces the Gauss–Kronrod helper functions with corresponding functions for a quantics grid. A more detailed discussion can be found in App. B.1.3. Listing 13 contains an equivalent code using `TCI.jl`.

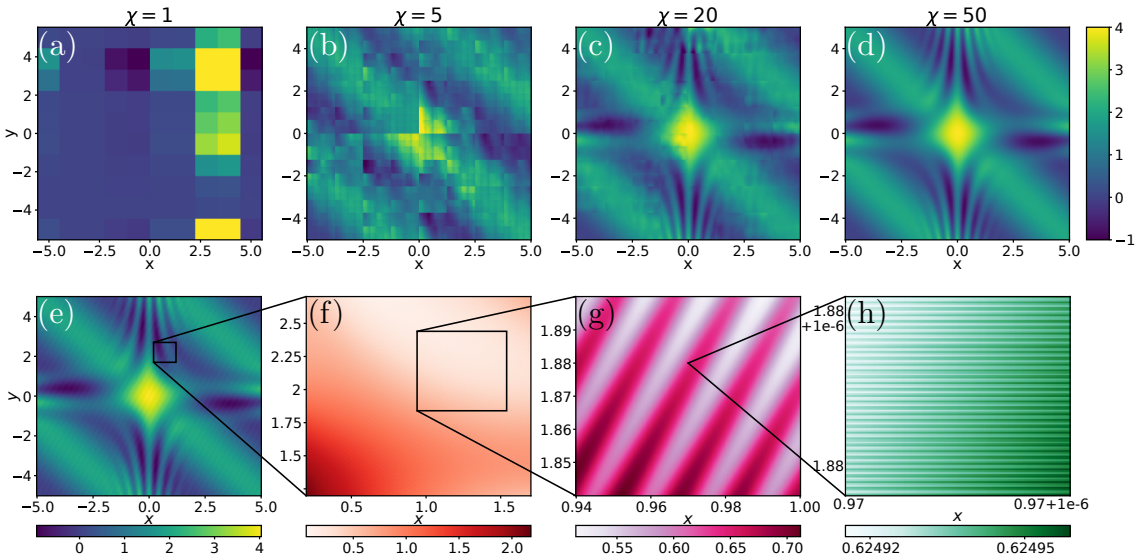


Figure 8: Quantics TCI (QTCI) representation of the function $f(x,y)$ defined in Eq. (79). (a–d) Approximations obtained for 4 different values of the MPS rank χ using $\mathcal{R} = 40$ plotted on a coarse grid of 300 × 300 points. (e–h) From left to right, the panels show different levels of zoom into the QTCI at $\chi = 50$ from coarse to very fine. At this rank, the compressed representation is numerically exact at all scales. [Code: Listing 7 (Python), 12 (Julia)]

6.4 Example: Heat equation using superfast Fourier transforms

In this section, we show how the different operations described earlier can be combined for a nontrivial application: solving a partial differential equation on a grid with exponentially many grid points [20].

Our example is the solution of the heat equation in 1D,

$$\partial_t u(x,t) = \partial_x^2 u(x,t)\,, \tag{80}$$

with a billion grid points and a complex initial condition with features at different scales. Since its solution is trivial in Fourier space, $u(k,t) = e^{-k^2 t} u(k,0)$, our strategy is simple: put $u(x,0)$ in quantics form using TCI, Fourier transform it (in ultrafast way), evolve it up to time t and Fourier transform back to real space.

We discretize the spatial variable as $x(m) = x_{\min} + m\delta$ with $\delta = (x_{\max} - x_{\min})/M$, $M = 2^{\mathcal{R}}$. Then, we view u as a vector with components $u_m(t) = u(x(m),t)$, satisfying the equation

$$\partial_t u_m(t) = \big(u_{m-1} - 2u_m + u_{m+1}\big)/\delta^2\,. \tag{81}$$

Taking the discrete Fourier transform of this equation using $u^{\mathrm{FT}} = Tu$ one obtains

$$\partial_t u_k^{\mathrm{FT}}(t) = -(2/\delta)^2 \sin^2(\pi k/M) u_k^{\mathrm{FT}}(t)\,. \tag{82}$$

For a given initial condition $u_m(0)$, with Fourier transform $u_k^{\mathrm{FT}}(0)$, this can be solved as

$$u_k^{\mathrm{FT}}(t) = g_k(t) u_k^{\mathrm{FT}}(0)\,, \qquad g_k(t) = \exp\!\big[-(2/\delta)^2 \sin^2(\pi k/M) t\big]\,. \tag{83}$$

The algorithm to solve the heat equation using the quantics representation is now straightforward. It is summarized through the following mappings:

$$u_m(0) \xrightarrow{\text{QTCI}} \widetilde{U}_{\boldsymbol{\sigma}}(0)\,, \qquad g_k(t) \xrightarrow{\text{QTCI}} \widetilde{G}_{\boldsymbol{\sigma}'}(t)\,, \qquad T_{km} \xrightarrow{\text{QTCI}} \widetilde{T}_{\boldsymbol{\sigma}'\boldsymbol{\sigma}}\,, \tag{84a}$$

$$\widetilde{U}_{\boldsymbol{\sigma}}(0) \xrightarrow{\times \widetilde{T}_{\boldsymbol{\sigma}'\boldsymbol{\sigma}}} \widetilde{U}^{\mathrm{FT}}_{\boldsymbol{\sigma}'}(0) \xrightarrow{\times \widetilde{G}_{\boldsymbol{\sigma}'}(t)} \widetilde{U}^{\mathrm{FT}}_{\boldsymbol{\sigma}'}(t) \xrightarrow{\times \widetilde{T}^{-1}_{\boldsymbol{\sigma}\boldsymbol{\sigma}'}} \widetilde{U}_{\boldsymbol{\sigma}}(t)\,. \tag{84b}$$

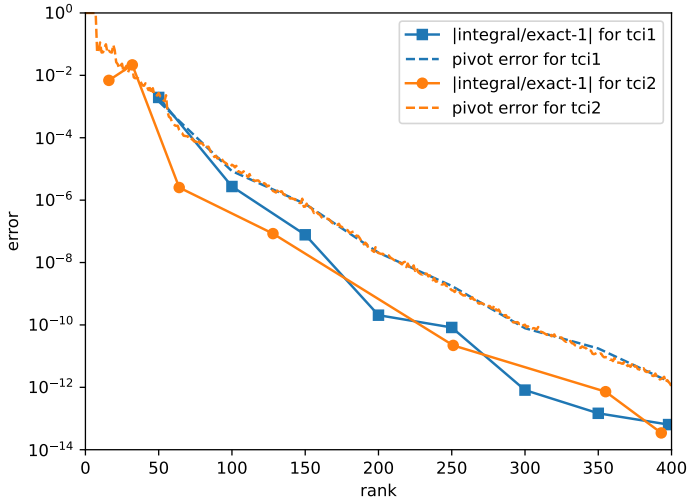


Figure 9: Consider the 3D integral $I = \int_{\mathbb{R}^3} d^3\mathbf{x}\, e^{-\sqrt{x^2+y^2+z^2}} = 8\pi$. We use `xfac` to compute its QTCI approximation, $\widetilde{I}$, on a uniform grid of $2^{3\mathcal{R}}$ points with $\mathcal{R} = 30$ in the cube $[-40, 40]^3$. The plot shows the error $\epsilon_I = |\widetilde{I}/I - 1|$ (solid lines with symbols) and the pivot error (dashed lines) as function of the MPS rank χ, computed using accumulative mode (blue) and reset mode (orange).

By Eq. (84a), we first QTCI all relevant objects; by Eq. (84b), we then Fourier transform the initial condition, time-evolve it in momentum space, and then Fourier transform it back to position space. The third step of Eq. (84b) involves element-wise multiplication of two tensor trains, $\widetilde{U}^{\mathrm{FT}}_{\boldsymbol{\sigma}'}(t) = \widetilde{G}_{\boldsymbol{\sigma}'}(t)\widetilde{U}^{\mathrm{FT}}_{\boldsymbol{\sigma}'}(0)$, performed separately for every $\boldsymbol{\sigma}'$. Note that the application of the tensor train operators $\widetilde{T}$ and $\widetilde{T}^{-1}$ are understood to each be followed by TCI recompressions.

We consider an initial condition with tiny, rapid oscillations added to a large, box-shaped background described by Heaviside θ-functions:

$$u(x,0) = \tfrac{1}{100}\left[1 + \cos(120x)\sin(180x)\right] + \theta\left(x - \tfrac{7}{2}\right)\left[1 - \theta\left(x - \tfrac{13}{2}\right)\right]. \tag{85}$$

Figure 11 shows the subsequent solution $u(x,t)$ at several different times. With increasing time, the initial oscillations die out (see inset) and in the long-time limit diffusive spreading is observed, as expected. The computation was performed for $\mathcal{R} = 30$, implying a very dense grid with $M = 2^{30}$ points, beyond the reach of usual numerical simulation techniques. Remarkably, however, the computational costs scale only linearly (not exponentially!) with $\mathcal{R}$. Indeed, even though the grid has around one billion points, obtaining the solution for one value of the time takes about one second on a single computing core.

The python code used to produce the data for Fig. 11 is shown in listing 9, App. B.1.4.

7 Application: Matrix product operators (MPOs)

A linear tensor operator $H_{\boldsymbol{\sigma}'\boldsymbol{\sigma}}$ can be unfolded into a matrix product operator (MPO) (also known as tensor train operator) using TCI. This is done by grouping the input and output indices together, $\mu_\ell = (\sigma'_\ell, \sigma_\ell)$, and performing TCI on the resulting tensor train $F_{\boldsymbol{\mu}} \equiv H_{\boldsymbol{\sigma}'\boldsymbol{\sigma}}$.

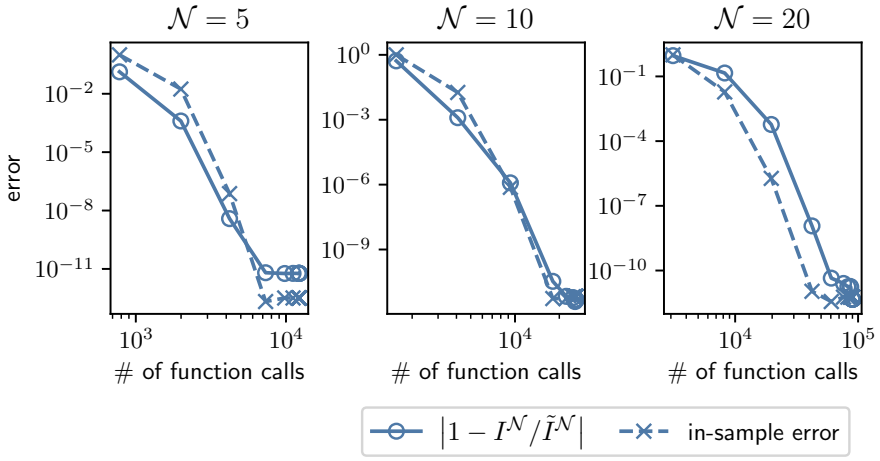
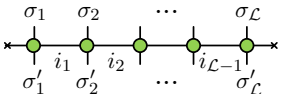


Figure 10: Performance metrics for the TCI2 computation of the integral $I^{(\mathcal{N})}$ of Eq. (64), for $\mathcal{N} = 5, 10, 20$, (left, middle, right). In all panels, the relative error $|1-I^{(\mathcal{N})}/\widetilde{I}^{(\mathcal{N})}|$ (straight lines with circles) and the relative in-sample error $|1-F_{\boldsymbol{\sigma}}/\widetilde{F}_{\boldsymbol{\sigma}}|$ (dashed lines with crosses) is plotted versus the number of function evaluations. The TCI2 computation of $\widetilde{I}^{(\mathcal{N})}$ has been obtained using $\mathcal{R} = 40$ quantics bits per variable in interleaved representation and a maximal bond dimension $\chi = 30$. [Code: Listing 8 (Python), 13 (Julia)]

One obtains an MPO, $H \approx \widetilde{H} = \prod_{\ell=1}^{\mathcal{L}} W_\ell$, with tensor elements of the form

$$[H]_{\boldsymbol{\sigma}'\boldsymbol{\sigma}} \approx \Big[\prod_{\ell=1}^{\mathcal{L}} W_\ell\Big]_{\boldsymbol{\sigma}'\boldsymbol{\sigma}} = [W_1]_{1i_1}^{\sigma_1'\sigma_1}[W_2]_{i_1i_2}^{\sigma_2'\sigma_2}\cdots[W_{\mathcal{L}}]_{i_{\mathcal{L}-1}1}^{\sigma_{\mathcal{L}}'\sigma_{\mathcal{L}}} =$$ (86)



In this section, we discuss a specific algorithm to perform this unfolding for the construction of the Hamiltonian MPO for quantum many-body problems. This construction is the first step of a DMRG many-body calculation. For this application, the Hamiltonian H is very sparse and a naive usage of TCI may fail there due to the ergodicity problem discussed in section 4.3.6. To avoid this issue, the algorithm and associated code (C++ header `autompo.h`) discussed below generates a MPO representation from a sum of rank-1 terms using element-wise tensor addition.

7.1 Formulation of the problem

Consider an $\mathcal{L}$-site quantum system whose many-body Hamiltonian is the sum of N_H rank-1 MPOs H_a,

$$H = \sum_{a=1}^{N_H} H_a\,, \qquad H_a = \prod_{\ell=1}^{\mathcal{L}} H_{a\ell}\,,$$ (87)

where each $H_{a\ell}$ is a local operator acting non-trivially only on site ℓ (see Eqs. (91) or (93) below for examples). Each term H_a in the sum is, by construction, a MPO of rank 1, but their sum $\sum_a H_a$ is not. The number of terms in the sum typically is exponentially smaller than the size of the Hilbert space in which the Hamiltonian lives, hence the operator of interest is very sparse. For instance, in quantum chemistry applications involving, say, $\mathcal{L}$ spin-orbitals, the number of terms is $\mathcal{O}(\mathcal{L}^4)$ while the size of the Hilbert space is $2^{\mathcal{L}}$. Naively, H is an MPO

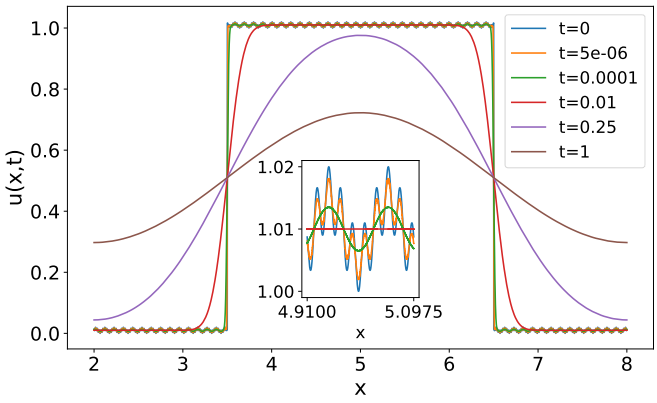


Figure 11: Solution of the heat equation (80) using quantics TCI. The plot shows $u(x,t)$ versus x for different times. We used a 1D grid with $M = 2^{\mathcal{R}}$ points and $\mathcal{R} = 30$, at a computational cost of $\mathcal{O}(\mathcal{R})$. The inset shows a zoom close to $x = 5$.

of rank N_H as one may express it as $\prod_{\ell=1}^{\mathcal{L}} W_\ell$, with

$$W_1 = (H_{1\mathcal{L}}, \ldots, H_{N_H\mathcal{L}}), \qquad W_{1<\ell<\mathcal{L}} = \begin{pmatrix} H_{1\ell} & 0 & \ldots & \ldots \\ 0 & H_{2\ell} & 0 & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & 0 & H_{N_H\ell} \end{pmatrix}, \qquad W_{\mathcal{L}} = \begin{pmatrix} H_{1\mathcal{L}} \\ H_{2\mathcal{L}} \\ \ldots \\ H_{N_H\mathcal{L}} \end{pmatrix}. \quad (88)$$

However, in many situation the actual rank is much smaller.

The problem of generating a compressed MPO from a sum of products of local operators is as old as the field of tensor networks itself. There are essentially three standard approaches to perform this task (see [61–64] for an in-depth discussion):

- Manual construction of the MPO, in particular using complementary operators. This method, pioneered in DMRG, is suitable for simple problems but not for general ones.
- Symbolic compression of the naive-sum MPO, in particular using bipartite graph theory. This powerful, automatic approach is exact. However, this approach makes implementing approximate compression (within a certain tolerance) rather complex and does not exploit specific relations between the values of matrix elements (all that matters is whether a term is present or not).
- Compression of the naive-sum MPO using SVD. This approach is widely used but has a well-known stability issue for large systems due to a numerical truncation error.[2]

The stability issue of the SVD compression can be understood as arising from the fact that SVD finds the best low-rank approximation of an $N \times N$ matrix A with respect to the *Frobenius* norm $|A|_F = (\sum_{ij} |A_{ij}|^2)^{1/2}$. When A is the sum of terms having very different Frobenius norms, numerical truncation errors may lead the algorithm to wrongly discard those with small norms. As an illustration, consider $A = \mathbb{1} + \psi\psi^\dagger$ where $\mathbb{1}$ is the identity matrix and ψ a normalized vector ($\psi^\dagger\psi = 1$). Here, we have $|A|_F = N + 3$. When N is very large (as in many-body problems, where $N \approx 2^{\mathcal{L}}$), the $\mathcal{O}(1)$ contribution from $\psi\psi^\dagger$ may be lost in numerical noise. By contrast, the prrLU does not suffer from this problem since it optimizes a different target norm (the *maximum* norm of the Schur complement).

[2] In Ref. [64], it is shown that this issue can be resolved for certain local Hamiltonians in the DMRG context by exploiting their specific structure.

7.2 MPO algorithm for quantum many-body problems

We propose to compress the naive-sum MPO using prrLU instead of SVD. Very importantly, in this approach, the full naive-sum MPO is never built. Our auto-MPO algorithm follows a divide-and-conquer strategy:

(1) Collect a fixed number $N_a \ll N_H$ of terms H_a.

(2) Construct the naive MPO of their sum using Eq. (88).

(3) Compress the resulting tensor train using CI canonicalization.

(4) Repeat steps (1-3) until all N_H terms have been processed.

(5) Sum and pairwise compress (formally in a binary tree) the N_H/N_a partial sums from (4).

The validity of the final MPO can be checked explicitly making use of the fact that H is a sparse matrix. Considering $F_\mu = H_{\sigma'\sigma}$ as a large vector, the tensor $\widetilde{F}_\mu$ is a correct unfolding of F_μ if and only if

$$\sum_\mu F_\mu^* \widetilde{F}_\mu = \sum_\mu |F_\mu|^2 = \sum_\mu |\widetilde{F}_\mu|^2 \tag{89}$$

(this guarantees that $|F - \widetilde{F}|_F = 0$ hence $F = \widetilde{F}$). This translates into

$$\sum_{a=1}^{N_H} \left(\sum_\mu [H_a]_\mu^* \widetilde{F}_\mu \right) = \sum_{a,a'=1}^{N_H} \left(\sum_\mu [H_{a'}^*]_\mu [H_a]_\mu \right) = \sum_\mu |\widetilde{F}_\mu|^2 \,. \tag{90}$$

Computing these expressions involves $\mathcal{O}(N_H)$ MPS contractions for the left side, enumerating the nonzero elements of the sparse matrices for the central part, and taking the trace of an MPO-MPO product for the right side. The same approach can be applied to the compression obtained by SVD or to compare the results of SVD and prrLU compressions.

We have tested the above algorithm against the same divide-and-conquer approach but with prrLU replaced by SVD, for the example $A = I_d + \psi\psi^\dagger$ where ψ is the rank-1 MPS $\psi_{\sigma_1 \cdots \sigma_\mathcal{L}} = \prod_\ell \delta_{\sigma_\ell,1}$. We found that SVD yields the correct rank-2 MPO for $\mathcal{L} < 103$ but fails for larger values of $\mathcal{L}$, incorrectly yielding a rank-1 MPO (the identity MPO). By contrast, the prrLU variant is stable for all values of $\mathcal{L}$ (up to 1000) that we have tested.

7.3 Illustration on Heisenberg and generic chemistry Hamiltonians

We illustrate the auto-MPO algorithm with two iconic Hamiltonian examples here: the Heisenberg Hamiltonian for a spin chain and a generic quantum chemistry Hamiltonian. The full code can be found in the folder `example/autoMPO/autoMPO.cpp` of the `xfac` library.

We start with the spin-$\frac{1}{2}$ Heisenberg Hamiltonian for an $\mathcal{L}$-site ring of spins:

$$H = \sum_{\ell=1}^{\mathcal{L}} S_\ell^z S_{\ell+1}^z + \tfrac{1}{2} \sum_{\ell=1}^{\mathcal{L}} \left(S_\ell^+ S_{\ell+1}^- + S_\ell^- S_{\ell+1}^+ \right), \tag{91}$$

$$S_\ell^\alpha = \underbrace{\mathbb{1} \otimes \mathbb{1} \otimes \cdots \otimes \mathbb{1}}_{\ell-1 \text{ times}} \otimes s^\alpha \otimes \underbrace{\mathbb{1} \otimes \cdots \otimes \mathbb{1}}_{\mathcal{L}-\ell \text{ times}} . \tag{92}$$

Here, the matrices $\mathbb{1} = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$, $s^z = \frac{1}{2}\left(\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right)$, $s^+ = \left(\begin{smallmatrix} 0 & 1 \\ 0 & 0 \end{smallmatrix}\right)$, $s^- = \left(\begin{smallmatrix} 0 & 0 \\ 1 & 0 \end{smallmatrix}\right)$ represent the single-site identity and spin operators for a spin-$\frac{1}{2}$ Hilbert space, while $S_\ell^{\alpha=z,\pm}$ represent site-ℓ spin operators for the full Hilbert space of the $\mathcal{L}$-site chain, acting non-trivially only on site ℓ. We use periodic boundary conditions, defining $S_{\mathcal{L}+1}^\alpha = S_1^\alpha$. Listing 4 shows a C++ code that first constructs the

Hamiltonian as a sum of local operators (an instance of the `polyOp` class), then generates the MPO (using the `to_tensorTrain()` method).

Our second example is a fully general quantum chemistry Hamiltonian of the form

$$H = \sum_{\ell_1 \ell_2} K_{\ell_1 \ell_2} c^\dagger_{\ell_1} c_{\ell_2} + \sum_{\ell_1 < \ell_2, \ell_3 < \ell_4} V_{\ell_1 \ell_2 \ell_3 \ell_4} c^\dagger_{\ell_1} c^\dagger_{\ell_2} c_{\ell_3} c_{\ell_4} \,. \tag{93}$$

The fermionic operators $c^\dagger_\ell$ (c_ℓ) create (destroy) an electron at spin-orbital ℓ. They satisfy standard anti-commutation relations, which we implement using a Jordan-Wigner transformation. We take all the coefficients $K_{\ell_1 \ell_2}$ and $V_{\ell_1 \ell_2 \ell_3 \ell_4}$ as random numbers for our benchmark (in a real application, the number of significant Coulomb elements would be smaller, typically $\mathcal{L}^3$ instead of $\mathcal{L}^4$ here). The example code is given in Listing 5 below. Table 3 shows the obtained ranks for up to $\mathcal{L} = 50$ orbitals which match the theoretical expectation. Note that the number of terms N_H for the larger size is greater than 10^6, hence a naive approach would fail here.

```
1  #include <xfac/tensor/auto_mpo.h>
2
3
4  using namespace std;
5  using namespace xfac;
6  using namespace xfac::autompo;
7
8
9  /// Heisenberg Hamiltonian (periodic boundary condition)
10 polyOp HeisenbergHam(int L)
11 {
12     auto Sz=[=](int i) { return prodOp {{ i%L, locOp {{-0.5,0},{0,0.5}} }}; };
13     auto Sp=[=](int i) { return prodOp {{ i%L, locOp {{0 ,0},{1,0}} }}; };
14     auto Sm=[=](int i) { return prodOp {{ i%L, locOp {{0 ,1},{0,0}} }}; };
15
16     polyOp H;
17     for(int i=0; i<L; i++) {
18         H += Sz(i)*Sz(i+1) ;
19         H += Sp(i)*Sm(i+1)*0.5 ;
20         H += Sm(i)*Sp(i+1)*0.5;
21     }
22     return H;
23 }
24
25
26 int main() {
27     int len=50;
28     auto H=HeisenbergHam(len);
29     TensorTrain mpo=H.to_tensorTrain();
30
31     cout<< "|1-<mpo|H>/<mpo|mpo>|=" << abs(1-H.overlap(mpo)/mpo.norm2()) << endl;
32
33     return 0;
34 }
```

Listing 4: C++ code to generate the MPO of the periodic Heisenberg Hamiltonian of Eq. (91). Lines 1–6 load the `xfac` library and namespaces. Lines 12–14 construct the spin operators of Eq. (92); note that only the single-site 2×2 matrices need to be specified explicitly. Lines 16–20 construct the sum $\sum_{\ell=1}^{\mathcal{L}}$ over all chain sites of the Hamiltonian Eq. (91). The maximun bond dimension obtained is 8 as it should be. This listing showcases the close similarity between formulae and corresponding code, which was one of the design goals of the `xfac` library.

Our C++ implementation is found in the namespace `xfac::autompo`. We define three classes: `locOp`, `prodOp`, and `polyOp`, corresponding to a local operator (i.e. a 2×2 matrix), a direct product of `locOp`, and a sum of `prodOp`, respectively. Our `prodOp` is a std::map going from int to locOp, while `polyOp` contains a std::vector of `prodOp`. The operators * and += are conveniently overloaded. Each of the classes `prodOp` and `polyOp` possesses the methods `to_tensorTrain()` (the actual algorithm to construct the MPO) and `overlap(mpo)` (to compute the left hand side of Eq. (89)).

```
polyOp ChemistryHam(arma::mat const& K, arma::mat const& Vijkl)
{
    auto Fermi=[=](int i, bool dagger)
    {
        locOp create={{0,1},{0,0}};
        auto ci=prodOp {{ i, dagger ? create : create.t() }};
        for(auto j=0; j<i; j++) ci[j]=locOp {{1,0},{0,-1}};    // fermionic sign
        return ci;
    };

    auto L=K.n_rows;
    polyOp H;

    for(auto i=0u; i<L; i++)
        for(auto j=0u; j<L; j++)
            if (fabs(K(i,j))>1e-14)
                H += Fermi(i,true)*Fermi(j,false)*K(i,j);   // kinetic energy

    for(auto i=0; i<L; i++)
        for(auto j=i+1; j<L; j++)
            for(auto k=0; k<L; k++)
                for(auto l=k+1; l<L; l++)
                    if (fabs(Vijkl(i+j*L,k+l*L))>1e-14)
                        H += Fermi(i,true)*Fermi(j,true)*Fermi(k,false)*
                             Fermi(l,false)*Vijkl(i+j*L,k+l*L);
    return H;
}
```

Listing 5: C++ code to generate the MPO of the quantum chemistry Hamiltonian of Eq. (93).

Table 3: Performance of our Auto-MPO construction for the quantum chemistry Hamiltonian of Eq. (93), for $\mathcal{L}$ orbitals, computed with an error tolerance of $\tau = 10^{-9}$. The third column is the bond dimension found with our approach. As a check, the fourth column gives the expected bond dimension obtained via the complementary-operator approach [63]. A naively constructed MPO would have bond dimension equal to the number of terms (2nd column), making it practically impossible to compress using SVD for $\mathcal{L} = 50$. [Code: Listing 5 (C++)]

$\mathcal{L}$	number of terms	bond dimension	$\mathcal{L}^2/2 + 3\mathcal{L}/2 + 2$
10	2125	67	67
30	190125	497	497
50	1503125	1327	1327

Table 4: Features supported by the main algorithms in `xfac/TCI.jl`.

	`TensorCI1`	`TensorCI2`		
feature		0-site	1-site	2-site
update mode	accumulative	accumulative & reset		
pivot search	full & rook	full	full	full & rook
nesting condition	full	no	full	partial
environment error	supported	no	no	planned support
recompression	not supported	supported	supported	supported
global pivots	not supported	supported	supported	supported

8 API and implementation details

We have presented a variety of use cases for our libraries `xfac/TCI.jl` in the examples above. After reading the present section, prospective users should be able to use our libraries in their own applications. In Sec. 8.1, we overview common features of the `xfac/TCI.jl` libraries. In Secs. 8.2 and 8.3 we provide language-specific information for C++ and Julia, respectively. We refer the reader to the tensor4all website [65] for the full documentation of the libraries.

Code for most examples contained in this paper is shown in Appendix B, and can be used as a starting point for implementations of new use cases. For more advanced use cases, it may be necessary to refer to the online documentation. We also encourage the readers to directly read the code of the library, in either language. It is indeed rather compact and often conveys the algorithms more transparently than lenghty explanations.

8.1 Implementation

For legacy reasons, `xfac/TCI.jl` contain two main classes for computing TCIs: `TensorCI1` and `TensorCI2`. `TensorCI1` is a variation of algorithm 5 of Ref. [12] and has been discussed in great detail in Ref. [13, Sec. III] (for a summary, see Sec. S-2 of the supplemental material of Ref. [15]). It is based on the conventional CI formula [38] and iteratively adds pivots one by one without ever removing any pivots (accumulative mode). `TensorCI2` is based on the more stable prrLU decomposition and implements 2-, 1- and 0-site TCI as described in this paper.

The numerical stability of the prrLU decomposition is inherited by `TensorCI2`, which often shows more reliable convergence. It is therefore used as a default in our codes. Nevertheless, since all TCI algorithms involve sampling, none of them is fully immune against missing some features of the tensor of interest, as already discussed above. Therefore, it may be necessary to enrich the sampling by proposing relevant global pivots before or during iteration (see Sec. 4.3.5). For instance, for the results shown in Fig. 9, we proposed 8 initial pivots according to the symmetry of the problem. Because of their different sampling patterns, it may also happen that `TensorCI1` finds much better approximations than `TensorCI2`. We have found at least one example where this was the case, and manual addition of some global pivots during initialization of `TensorCI2` solved the issue. There are minor differences in other features supported by `TensorCI1` and `TensorCI2`, which are summarized in Table 4. Most importantly, 0-site and 1-site optimization is only available in `TensorCI2`. Therefore, we offer convenient conversion between both classes.

General tensor trains, possibly obtained from an external source, are represented by a class `TensorTrain`. It supports related algorithms that are agnostic to the specific index structure of a TCI, such as evaluation, summation or compression using LU, CI or SVD. It also serves as an interface to other tensor network algorithms, such as those implemented in ITensor [33], to allow for quick incorporation of the TCI libraries into existing code. A `TensorCI1`/`TensorCI2` object

Table 5: Features supported by the main TCI algorithms in `xfac`. In the code, ci1 is a `TensorCI1`, ci2 is a `TensorCI2`, p is a `TensorCIParam` used to build a `TensorCI`, and tt is a `TensorTrain`. The method iterate(nIter, nSite) receives the number of iterations nIter to perform and the number of physical sites nSite to use for the matrix CI (can be 0, 1, or 2).

Section	feature	variant	example C++ code
4.2	nesting	no	ci2.iterate()
		full	ci1.iterate()
			ci2.makeCanonical()
4.3.3	pivot update	accumulative	ci1.iterate()
		reset	ci2.iterate()
4.3.7	environment	active if	p.weight=...
4.3.4	pivot search	rook	p.fullPiv=false
		full	p.fullPiv=true
4.3.5	global pivots		ci2.addGlobalPivots(...)
4.4	0-site		ci2.iterate(1,0)
	1-site		ci2.iterate(1,1)
4.5	compression	SVD	tt.compressSVD()
		LU	tt.compressLU()
		CI	tt.compressCI()
4.5	conversion	tci1 → tci2	to_tci2(tci1)
		tci2 → tci1	to_tci1(tci2)

can be trivially converted to a `TensorTrain` object. Conversion in the inverse direction is done by making the `TensorTrain` CI-canonical using the algorithm described in Sec. 4.5, resulting in a `TensorCI2` object.

8.2 C++ API (`xfac`)

The file "readme.txt" explains the installation procedure and how to generate the detailed documentation. The main components of the library are represented in Figure 12. As mentioned above, the classes `TensorCI1` and `TensorCI2` build a TCI of an input function. The main output is the tensor train, stored in the class `TensorTrain`, which represents a list of 3-leg tensors.

Below, we summarize the C++ API especially focusing on `TensorCI2`; the API for `TensorCI1` is similar and can be found in the documentation. A `TensorCI2` can be constructed from a tensor function $f : (x_1, x_2, \ldots, x_{\mathcal{L}}) \to \mathbb{C}$ and its local dimensions $\{d_\ell\}$ where the index $x_\ell \in \{1, \ldots, d_\ell\}$ with $\ell = 1, 2, \ldots, \mathcal{L}$. This is the main constructor:

```
TensorCI2(
  function<T(vector<int>)> f,
  vector<int> localDim,
  TensorCI2Param param={}
);
```

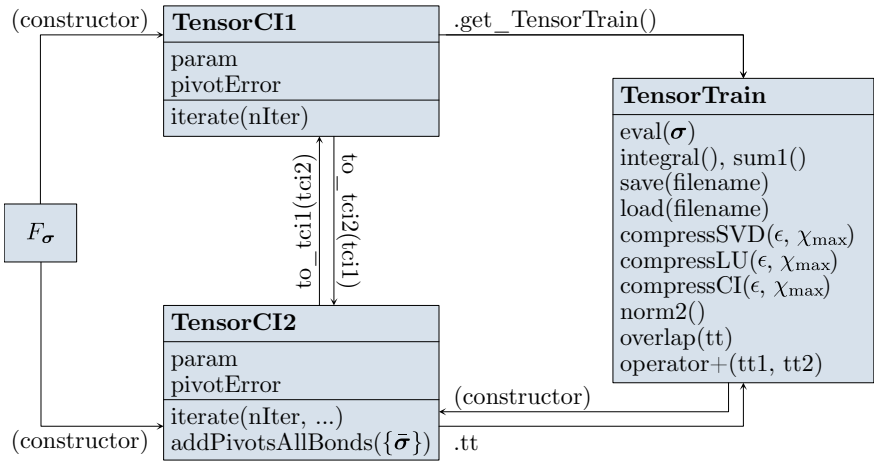


Figure 12: Scheme of the main conversions implemented in xfac.

The parameters of the cross interpolation can be set in the constructor by the class `TensorCI2Param`:

```
struct TensorCI2Param {
  int bondDim=30;                    ///< max bond dimension of tensor train
  double reltol=1e-12;               ///< expected relative tolerance of CI
  vector<int> pivot1;                ///< first pivot (optional)
  bool fullPiv=false;                ///< whether to use full pivoting
  int nRookIter=3;                   ///< number of rook pivoting iterations
  vector<vector<double>> weight;     ///< activates the ENV learning
  function<bool(vector<int>)> cond;  ///< cond(x)=false when x should not be a pivot
  bool useCachedFunction=true;       ///< whether to use internal caching
};
```

For `TensorCI1`, `TensorCI1Param` is used to set the parameters. We refer to the documentation for more details.

To factorize a continuous function $f : \mathbb{R}^{\mathcal{L}} \to \mathbb{R}$, `xfac` introduces the class `CTensorCI2` (or `CTensorCI1`). `CTensorCI2` is a `TensorCI2` that can be constructed from a multidimensional function f by providing also the grid of points for each component:

```
CTensorCI2(
  function<T(vector<double>)> f,
  vector<vector<double>> const& xi,
  TensorCI2Param param={}
);
```

The main output of `CTensorCI2` is a continuous tensor train `CTensorTrain`, which can be evaluated at any point in $\mathbb{R}^{\mathcal{L}}$, including those outside the original grid (cf. App. A.4).

As discussed in Sec. 6.1, functions of continuous variables can also be discretized using the quantics representation. For that, `xfac` introduces the helper class `QTensorCI2`, currently available only for `TensorCI2`. It can be constructed from a multidimensional function f by providing the quantics grid in addition to the parameters required for `TensorCI2`:

```
QTensorCI2(
  function<T(vector<double>)> f,
  grid::Quantics const& qgrid,
  TensorCI2Param param={}
);
```

Specifically, the `grid::Quantics` type represents an uniform grid on the hypercube $[a, b)^{\mathcal{N}}$ with $2^{\mathcal{R}\mathcal{N}}$ points:

```
struct Quantics {
    double a=0, b=1;          ///< start and end points of interval
    int nBit=10;              ///< number of bits for each variable
    int dim=1;                ///< dimension of hypercube
    bool fused=false;         ///< whether to fuse the bits for the same scale (default:
        false)
}
```

The main output of `QTensorCI2` is a quantics tensor train `QTensorTrain`, which is a cheap representation of the function that can be evaluated, and saved/loaded to file.

8.3 Julia libraries

The Julia implementation of TCI is subdivided into several parts:

- `TensorCrossInterpolation.jl` (referred to as `TCI.jl`) contains only TCI and associated algorithms for tensor cross interpolation.
- `QuanticsGrids.jl` contains functionality to construct quantics grids, and to convert indices between direct and quantics representations.
- `QuanticsTCI.jl` is a thin wrapper around `TCI.jl` and `QuanticsGrids.jl` to allow for convenient quantics tensor cross interpolation in the most common use cases.
- `TCIITensorConversion.jl` is a small helper library to convert between tensor train objects and MPS/MPO objects of the `ITensors.jl` library.

All four libraries are available through Julia's general registry and can thus be installed by

```
import Pkg; Pkg.install("TensorCrossInterpolation")
```

and analogous commands. Below, we present only the main functionalities that were used for the examples in this paper. A complete documentation can be found online [65].

8.3.1 TensorCrossInterpolation.jl

Similar to `xfac`, `TCI.jl` has classes `TensorCI1` and `TensorCI2` that build a TCI of an input function, as well as a general-purpose `TensorTrain` class. These three classes and their main functions are shown in Fig. 13. Given a function of interest, $f : (x_1, x_2, \ldots, x_{\mathcal{L}}) \to \mathbb{C}$ and its local dimensions $\{d_\ell\}$, the most convenient way to obtain a `TensorCI1`/`TensorCI2` is by calling `crossinterpolate1`/`crossinterpolate2`. Since the algorithm based on prrLU is usually more stable, we recommend using `crossinterpolate2` as a default.

```
function crossinterpolate2(
  ::Type{ValueType},        # Return type of f, usually Float64 or ComplexF64
  f,                        # Function of interest: <@$f_\bsigma$@>
  localdims::Union{Vector{Int},NTuple{N,Int}},  # Local dimensions <@$(d_1, \ldots,
      d_\scL)$@>
  initialpivots::Vector{MultiIndex};  # List of initial pivots <@$\{\hat\bsigma\}$@>.
      Default: <@$\{(1, \ldots, 1)\}$@>
  tolerance::Float64,       # Global error tolerance <@$\tau$@> for TCI. Default:
      <@$10^{-8}$@>
```

```
    pivottolerance::Float64,  # Local error tolerance <@$\tau_{\mathrm{loc}}$@> for prrLU.
        Default: <@$\tau$@>
    maxbonddim::Int,          # Maximum bond dimension <@$\chi_{\max}$@>. Default: no limit
    maxiter::Int,             # Maximum number of half-sweeps. Default: <@$20$@>
    pivotsearch::Symbol,      # Full or rook pivot search? Default: :full
    normalizeerror::Bool,     # Normalize <@$\varepsilon$@> by <@$\max_{\bsigma \in
        \mathrm{samples}} F_\bsigma$@>? Default: true
    ncheckhistory::Int        # Convergence criterion: <@$\varepsilon < \tau$@> for how
        many iterations? Default: 3
) where {ValueType,N}
```

The three required positional arguments specify basic features of the tensor to be approximated. `f` is a function that produces tensor components when called with a vector of indices. For instance, `f([1, 2, 3, 4])` should return the value of f_{1234}. If appropriate pivots are known beforehand, they can be put in the list `initialpivots`, which is used to initialize the TCI. The convergence of TCI is controlled by mainly by the arguments `tolerance`, which is the global error tolerance τ of the TCI approximation, and `pivottolerance`, which determines the local error tolerance during 2-site updates. Usually, it is best to set `pivottolerance` to `tolerance` or slightly below `tolerance`. Both should be larger than the numerical accuracy, else the cross approximation may become numerically unstable. The maximum number of sweeps, controlled by `maxiter`, can be chosen rather small in reset mode, as the algorithm requires only a few sweeps.

After convergence, `crossinterpolate2` returns an object of type `TensorCI2` that represents the tensor train, as well as two vectors: `ranks` contains the bond dimension χ, and `errors` the error estimate ε, both as a function of iteration number. For example, a possible call to `crossinterpolate2` to approximate a complex tensor $f_{\sigma_1 \cdots \sigma_4}$ with 4 indices $\sigma_\ell \in \{1, 2, \ldots, 8\}$ up to tolerance 10^{-5} with TCI would be:

```
tci, ranks, errors = crossinterpolate2(ComplexF64, f, fill(8, 4); tolerance=1e-5)
```

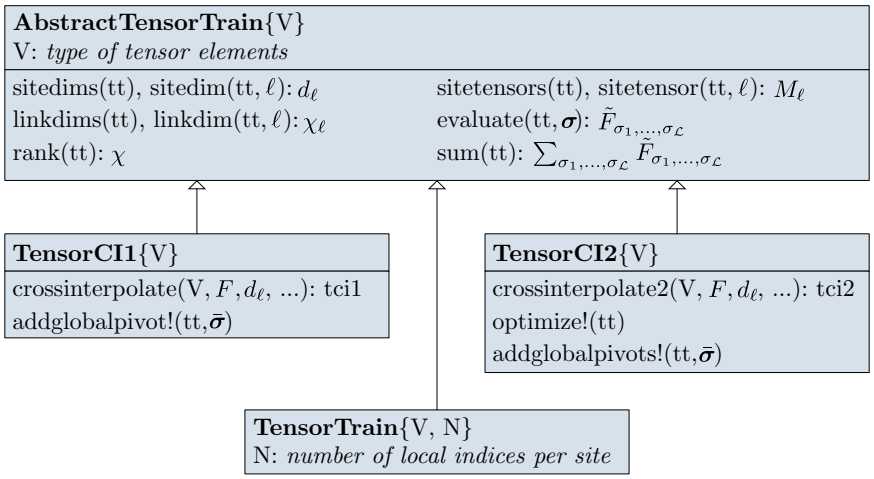


Figure 13: Schematic of the relations between the most important types in `TensorCrossInterpolation.jl`. Here, functions associated to types do not signify member functions, but rather functions operating on these types. By convention, functions ending with an exclamation mark '!' modify the object, while all other functions leave the object unchanged.

Table 6: Features supported by the main TCI algorithms in `TCI.jl`. In the code, tci1 is a `TensorCI1`, tci2 is a `TensorCI2`, and tt is a `TensorTrain`.

Section	feature	variant	example julia code
4.2	nesting	no	crossinterpolate2(ValueType, f; ...)
		full	crossinterpolate1(ValueType, f; ...)
			makecanonical!(tci2, f; ...)
4.3.3	pivot update	accumulative	crossinterpolate1(ValueType, f; ...)
		reset	crossinterpolate2(ValueType, f; ...)
4.3.4	pivot search	rook	crossinterpolate2(..., pivotsearch=:rook, ...)
		full	crossinterpolate2(..., pivotsearch=:full, ...)
4.3.5	global pivots		addglobalpivot!(tci1, ...)
			addglobalpivots!(tci2, ...)
4.4	0-site		sweep0site!(tci2, ...)
	1-site		sweep1site!(tci2, ...)
4.5	compression	SVD	compress!(tt, :SVD, ...)
		LU	compress!(tt, :LU, ...)
		CI	compress!(tt, :CI, ...)
4.5	conversion	tci1 → tci2	TensorCI1{ValueType}(tci2, f; ...)
		tci2 → tci1	TensorCI2{ValueType}(tci1)

To evaluate the resulting TCI, call the object as a functor in the same way as the original function. For example, `tci([1, 2, 3, 4])` should be approximately equal to `f([1, 2, 3, 4])`. This is equivalent to a call `evaluate(tci, [1, 2, 3, 4])`. A sum over the TCI, e.g. to calculate an integral, is obtained by calling `sum(tci)`. If the only objective is to calculate an integral, it is more convenient to use the function `integrate(...)`, which calculates the integral of a function by building a weighted TCI on a Gauss–Kronrod grid and performing efficient weighted summation. Alternatively, quantics schemes described in the next section can be used for this task.

To apply more complicated tensor network algorithms to `tci`, it is useful to convert it into a `TensorTrain` object, which gives access to functions that do not preserve the CI-canonical gauge, such as SVD-based compression. With `TCIITensorConversion.jl`, all tensor train like objects can also be converted to actual `MPS` and `MPO` objects of the `ITensors.jl` library, which contains much more functionality [33].

8.3.2 Quantics grids and QTCI

Two associated libraries, `QuanticsGrids.jl` and `QuanticsTCI.jl`, offer convenient functionality to perform computations in quantics representation. `QuanticsGrids.jl` offers conversion between quantics indices, linear indices and function variables on (multidimensional) quantics grids. For example, fused quantics indices for a $\mathcal{R} = 10$ bit quantics grid on a hypercube $[-1, +1]^3$ can be obtained using the following code:

```
import QuanticsGrids as QG
grid = QG.DiscretizedGrid{3}(10, (-1.0, -1.0, -1.0), (1.0, 1.0, 1.0);
    unfoldingscheme=:fused)
sigma = QG.grididx_to_quantics(grid, (3, 4, 5)) # Translate <@$\vec{m} = (3, 4, 5)
    \rightarrow \bsigma(\vec{m})$@>
m = QG.quantics_to_grididx(grid, sigma)         # <@$\vec{m}(\bsigma)$@>
x = QG.quantics_to_origcoord(grid, sigma)       # <@$\vec{x}(\bsigma)$@>
```

To create a quantics TCI of a user-supplied function, these grids can be used together with `quanticscrossinterpolate(...)` from `QuanticsTCI.jl`, which translates a given function $f(x_1, \ldots, x_N)$ to its quantics representation F_σ, and applies the `crossinterpolate2` to F_σ in a single call. The function signature is

```
function quanticscrossinterpolate(
  ::Type{ValueType},        # Return type of f, usually Float64 or ComplexF64
  f,                        # Function of interest <@$f(x)$@>
  grid,                     # Discretization grid, as QuanticsGrids.Grid, Array, or Range
  initialpivots::Vector{MultiIndex};  # List of initial pivots <@$\{\bar{\vec{m}}\}$@>.
      Default: <@$\{(1, \ldots, 1)\}$@>
  unfoldingscheme::Symbol,  # Fused or interleaved representation? Default: :interleaved
  kwargs...                 # All other arguments are forwarded to crossinterpolate2().
) where {ValueType}
```

The vector `initialpivots` enumerates the pivots used to initialize the TCI, as indices into `xvals` that are automatically translated to quantics form. Thus, this function takes care of all conversions to quantics representation that the user would otherwise have to do manually. It returns a QTCI, a vector of ranks, and a vector of errors, similar to `crossinterpolate2`, for example:

```
qtci, ranks, errors = quanticscrossinterpolate(Float64, f, grid, tolerance=1e-5)
```

Here, `qtci` is a `QuanticsTensorCI2` object, a thin wrapper around `TensorCI2` that translates between regular indices and their quantics representation. Similar to `TensorCI2`, objects of this type can be evaluated using function call syntax. For example, `qtci(m)` should be approximately equal to `f(QG.grididx_to_origcoord(grid, m))`. The object's components can be accessed as `qtci.tci` and `qtci.grid`. For a complete documentation of all functionality, see the online documentation of the respective libraries [65].

9 Perspectives

In this article, we have presented old and new variants of the tensor cross interpolation (TCI) algorithm, their open source C++, python and Julia implementations as well as a wide range of applications (integration in high dimension, solving partial differential equations, construction of matrix product operators, ...).

TCI has a very peculiar position among other tensor network algorithms: it provides an automatic way to map a very large variety of physics and applied mathematics problems onto the MPS toolbox. Of course not all mathematical objects admit a low-rank representation. But some problems do, and those will strongly benefit from being mapped onto the tensor network framework. Progress in computational sciences often corresponds to exploiting a particular structure of the problem. TCI belongs to the rare class of algorithms capable of discovering such structures for us. We surmise that TCI and related tools will play a major role in extending the scope of the MPS toolbox to applications beyond its original purpose of manipulating many-body wavefunctions.

An interesting side aspect of TCI is that offers a simple Go/No-Go test for the feasibility of speeding up computations using the MPS toolbox. Suppose, e.g., that one is in possession of a solver for a partial differential equation. One can feed some typical solutions into TCI to check whether they are strongly compressible—if so, a faster solver can likely be built using MPS tools. Using this very simple approach, the authors of this article have already identified numerous compressible objects in a wide range of contexts.

Acknowledgments

J.v.D., H.S. and M.R. thank Markus Wallerberger for very interesting discussions on a relation between prrLU and CI decompositions. X.W. and O.P. thank Miles Stoudenmire for valuable feedback on the manuscript. M.R. thanks the Center for Computational Quantum Physics at the Flatiron Institute of the Simons Foundation for hospitality during an extended visit.

Author contributions Y.N.F. and X.W. initiated the project. Y.N.F. conceived the TCI-via-prrLU algorithms with the help of X.W. and developed the xfac library. M.R., S.T. and H.S. developed the TCI.jl library based on xfac. Y.N.F., M.R., M.J., J.W.L., T.L. and T.K. contributed examples of applications of these libraries. X.W., M.R., O.P., J.v.D., Y.N.F. and H.S. wrote the paper and contributed to the proofs. Y.N.F. and M.R. contributed comparable amounts of work.

Funding information H.S. was supported by JSPS KAKENHI Grants No. 21H01041, No. 21H01003, No. 22KK0226, and No. 23H03817, as well as JST PRESTO Grant No. JPMJPR2012 and JST FOREST Grant No. JPMJFR2232, Japan. This work was supported by Institute of Mathematics for Industry, Joint Usage/Research Center in Kyushu University. (FY2023 CATEGORY "IUse of Julia in Mathematics and Physics" (2023a015).) X.W. acknowledges the funding of Plan France 2030 ANR-22-PETQ-0007 "EPIQ", the PEPR "EQUBITFLY", the ANR "DADI" and the CEA-FZJ French-German project AIDAS for funding. J.v.D. acknowledges funding from the Deutsche Forschungsgemeinschaft under grant INST 86/1885-1 FUGG and under Germany's Excellence Strategy EXC-2111 (Project No. 390814868), and the Munich Quantum Valley, supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus. The Flatiron Institute is a division of the Simons Foundation.

A Proofs of statements in the main text

A.1 Proof of the quotient identity for the Schur complement

Below, we give a simple proof of the quotient identity (17), i.e. that taking the Schur complement with respect to multiple blocks either simultaneously or sequentially yields the same result. Consider two block matrices

$$A = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix}, \qquad B \equiv \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \tag{A.1}$$

where A_{11} and B are invertible submatrices of A. From (13), we factorize the B matrix as

$$B = \begin{pmatrix} \mathbb{1}_{11} & 0 \\ A_{21}A_{11}^{-1} & \mathbb{1}_{22} \end{pmatrix} \begin{pmatrix} A_{11} & 0 \\ 0 & [B/A_{11}] \end{pmatrix} \begin{pmatrix} \mathbb{1}_{11} & A_{11}^{-1}A_{12} \\ 0 & \mathbb{1}_{22} \end{pmatrix},$$

which is easy to invert as

$$B^{-1} = \begin{pmatrix} \mathbb{1}_{11} & -A_{11}^{-1}A_{12} \\ 0 & \mathbb{1}_{22} \end{pmatrix} \begin{pmatrix} A_{11}^{-1} & 0 \\ 0 & [B/A_{11}]^{-1} \end{pmatrix} \begin{pmatrix} \mathbb{1}_{11} & 0 \\ -A_{21}A_{11}^{-1} & \mathbb{1}_{22} \end{pmatrix}.$$

This implies that

$$\left(B^{-1}\right)_{22} = [B/A_{11}]^{-1}. \tag{A.2}$$

We then form the Schur complement

$$\begin{aligned}[A/B] &= A_{33} - (A_{31}, A_{32}) B^{-1} \begin{pmatrix} A_{13} \\ A_{23} \end{pmatrix} \\ &= A_{33} - (A_{31}, A_{32}) \begin{pmatrix} \mathbb{1}_{11} & -A_{11}^{-1}A_{12} \\ 0 & \mathbb{1}_{22} \end{pmatrix} \begin{pmatrix} A_{11}^{-1} & 0 \\ 0 & [B/A_{11}]^{-1} \end{pmatrix} \begin{pmatrix} \mathbb{1}_{11} & 0 \\ -A_{21}A_{11}^{-1} & \mathbb{1}_{22} \end{pmatrix} \begin{pmatrix} A_{13} \\ A_{23} \end{pmatrix} \\ &= A_{33} - \big(A_{31}, (A_{32} - A_{31}A_{11}^{-1}A_{12})\big) \begin{pmatrix} A_{11}^{-1} & 0 \\ 0 & [B/A_{11}]^{-1} \end{pmatrix} \begin{pmatrix} A_{13} \\ A_{23} - A_{21}A_{11}^{-1}A_{13} \end{pmatrix} \\ &= A_{33} - A_{31}A_{11}^{-1}A_{13} - (A_{32} - A_{31}A_{11}^{-1}A_{12})[B/A_{11}]^{-1}(A_{23} - A_{21}A_{11}^{-1}A_{13}) . \end{aligned} \tag{A.3}$$

On the other hand, the Schur complement $[A/A_{11}]$ has the explicit block form

$$\begin{aligned}[A/A_{11}] &= \begin{pmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{pmatrix} - \begin{pmatrix} A_{21} \\ A_{31} \end{pmatrix} (A_{11})^{-1} (A_{12} \;\; A_{13}) \\ &= \begin{pmatrix} [B/A_{11}] & A_{23} - A_{21}A_{11}^{-1}A_{13} \\ (A_{32} - A_{31}A_{11}^{-1}A_{12}) & A_{33} - A_{31}A_{11}^{-1}A_{13} \end{pmatrix} . \end{aligned} \tag{A.4}$$

Taking the Schur complement of the above matrix with respect to its upper left block $[B/A_{11}]$ yields an expression which we recognize as the last line of Eq.(A.3). This proves the Schur quotient identity (17):

$$\big[[A/A_{11}]/[B/A_{11}]\big] = [A/B] . \tag{A.5}$$

A.2 Convergence and rook conditions in block rook search

This section proves that the block rook search Algorithm 1 (see p. 14) converges, and that upon convergence, the pivots satisfy rook conditions.

Definition: Block rook conditions Given lists $\mathcal{I} = (i_1, \dots, i_\chi)$ and $\mathcal{J} = (j_1, \dots, j_\chi)$ of pivots, the block rook search algorithm alternates between factorizing $A(\mathbb{I}, \mathcal{J})$ and $A(\mathcal{I}, \mathbb{J})$, updating $\mathcal{I}$ and $\mathcal{J}$ after each factorization. In odd iterations, the block rook search obtains lists $\mathcal{I}' = (i'_1, \dots, i'_\chi)$ and $\mathcal{J}' = (j'_1, \dots, j'_\chi)$ from a prrLU factorization of $A(\mathbb{I}, \mathcal{J})$. Since the matrix $A(\mathbb{I}, \mathcal{J})$ has more rows than columns, the new column indices $\mathcal{J}'$ are a permutation of the old column indices $\mathcal{J}$, whereas $\mathcal{I}'$ may contain new elements that are not in $\mathcal{I}$. During a prrLU, we denote by A_r the pivot matrix after the inclusion of the first r pivots, i.e. $A_r = A((i'_1, \dots, i'_r), (j'_1, \dots, j'_r))$. These pivots satisfy,

$$(i'_r, j'_r) = \operatorname{argmax}[A(\mathbb{I}, \mathcal{J})/A_{r-1}] . \tag{A.6}$$

For even iterations, one factorizes $A(\mathcal{I}, \mathbb{J})$, the new pivots satisfy

$$(i'_r, j'_r) = \operatorname{argmax}[A(\mathcal{I}, \mathbb{J})/A_{r-1}] , \tag{A.7}$$

and $\mathcal{I}'$ is a permuation of $\mathcal{I}$. After each prrLU, the pivots lists are updated $\mathcal{I} \leftarrow \mathcal{I}', \mathcal{J} \leftarrow \mathcal{J}'$. The process ends when $\mathcal{I}' = \mathcal{I}$ and $\mathcal{J}' = \mathcal{J}$.

Definition: Rook conditions. The pivots generated by sequential rook search (i.e. the standard rook search algorithm known from literature) fulfill the following set of rook conditions:

$$i_r = \operatorname{argmax}([A/A_{r-1}](\mathbb{I}, j_r)) , \tag{A.8}$$

$$j_r = \operatorname{argmax}([A/A_{r-1}](i_r, \mathbb{J})) , \tag{A.9}$$

where $A_r = A((i_1, \dots, i_r), (j_1, \dots, j_r))$.

Statement 1. The pivots $(i_1, j_1), \ldots, (i_\chi, j_\chi)$ found by a converged block rook search satisfy the rook conditions (A.8),(A.9). The proof follows from the restriction property of the Schur complement. At convergence, $(i'_r, j'_r) = (i_r, j_r)$ for each $r = 1, \ldots, \chi$. Applying Eq. (26) to the Schur complement of Eq.(A.6), one immediately gets Eq. (A.8). Similarly, one gets Eq. (A.9) from Eq. (A.7).

Statement 2. The block rook search must converge in a finite number of steps. The proof can be done iteratively. The search of (i_1, j_1) correspond to looking for the maximum of $A(\mathbb{I}, \mathcal{J})$ (odd iterations) or $A(\mathcal{I}, \mathbb{J})$ (even iterations). For odd iterations $i'_1 = i_1$ unless new columns (that have never been seen by the algorithm) have been introduced in the previous even iteration. Since there are only a finite number of columns, this process must terminate in a finite number of iterations. The same argument works for j_1 and the even iterations.

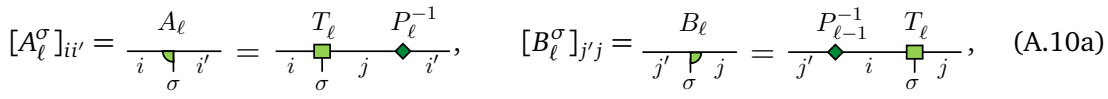
To show that the search for (i_2, j_2) must terminate, one applies the same reasoning to $[A/A_1]$ after (i_1, j_1) has converged. One continues the proof iteratively for all (i_r, j_r). In case the matrix $[A/(1, \ldots, r-1)]$ has multiple entries with the same maximum value, the ambiguity must be lifted to guarantee that the algorithm terminates. A solution is to choose $(i'_r, j'_r) = (i_r, j_r)$ whenever the previously seen pivot (i_r, j_r) is among the maximum elements of that matrix.

A.3 Nesting properties

Consider a tensor train $\widetilde{F}$ in TCI form (34). If its pivots satisfy nesting conditions, the T^σ_ℓ and P_ℓ matrices have certain useful properties, derived in Ref. [13, App. C] and invoked in the main text. Here, we summarize them and recapitulate their derivations.

For each ℓ we define the matrices $A^\sigma_\ell = T^\sigma_\ell P^{-1}_\ell$ and $B^\sigma_\ell = P^{-1}_{\ell-1} T^\sigma_\ell$, with elements

$$[A^\sigma_\ell]_{ii'} = \quad = \quad , \qquad [B^\sigma_\ell]_{j'j} = \quad = \quad , \tag{A.10a}$$

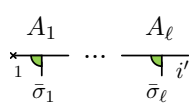


for $i \in \mathcal{I}_{\ell-1}$, $i' \in \mathcal{I}_\ell$, $j' \in \mathcal{J}_\ell$, $j \in \mathcal{J}_{\ell+1}$, $\sigma \in \mathbb{S}_\ell$. If the unprimed indices $i \oplus (\sigma)$ or $(\sigma) \oplus j$ are restricted to $\mathcal{I}_\ell$ or $\mathcal{J}_\ell$, respectively, we obtain Kronecker symbols, in analogy to Eq. (10):

$$[A^\sigma_\ell]_{ii'} = \delta_{i\oplus(\sigma),i'}\,, \quad \forall\, i\oplus(\sigma) \in \mathcal{I}_\ell\,, \qquad [B^\sigma_\ell]_{j'j} = \delta_{j',(\sigma)\oplus j}\,, \quad \forall\, (\sigma)\oplus j \in \mathcal{J}_\ell\,. \tag{A.11}$$

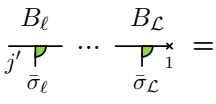
If the pivots are left-nested up to ℓ, and if $\bar{\imath}_\ell = (\bar\sigma_1, \ldots, \bar\sigma_\ell)$ is an index from a row pivot list, $\bar{\imath}_\ell \in \mathcal{I}_\ell$, the same is true for any of its subindices, $\bar{\imath}_{\ell'} \in \mathcal{I}_{\ell'}$ for $\ell' < \ell$. Hence, iterative use of Eq. (A.11), starting from $A_1 A_2$, yields a telescope collapse of the following product:

$$\quad = [A_1^{\bar\sigma_1} \cdots A_\ell^{\bar\sigma_\ell}]_{1i'} = \delta_{\bar{\imath}_\ell, i'}\,, \quad \forall\, \bar{\imath}_\ell \in \mathcal{I}_\ell \quad \text{if} \quad \mathcal{I}_0 < \mathcal{I}_1 < \cdots < \mathcal{I}_\ell\,. \tag{A.12a}$$



Similarly, if the pivots are right-nested up to ℓ, and $\bar\jmath_\ell = (\bar\sigma_\ell, \ldots, \bar\sigma_\mathcal{L}) \in \mathcal{J}_\ell$, we obtain

$$\quad = [B_\ell^{\bar\sigma_\ell} \cdots B_\mathcal{L}^{\bar\sigma_\mathcal{L}}]_{j'1} = \delta_{j',\bar\jmath_\ell}\,, \quad \forall\, \bar\jmath_\ell \in \mathcal{J}_\ell \quad \text{if} \quad \mathcal{J}_\ell > \cdots > \mathcal{J}_\mathcal{L} > \mathcal{J}_{\mathcal{L}+1}\,. \tag{A.12b}$$

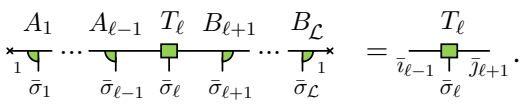


We stress that such collapses do not apply for all configurations, only for pivots from left- or right-nested lists, respectively. Thus, the As and Bs are not isometries: $\sum_\sigma [A^{\sigma\dagger}_\ell A^\sigma_\ell]_{ii'} \neq \delta_{ii'}$ and $\sum_\sigma [B^\sigma_\ell B^{\sigma\dagger}_\ell]_{j'j} \neq \delta_{j'j}$, because the $\sum_\sigma$ sums involve non-pivot configurations.

The above telescope collapses are invoked in the following arguments:

- *1-Site nesting w.r.t. T_ℓ:* We say that pivots are *nested w.r.t.* T_ℓ if they are left-nested up to $\ell-1$ and right-nested up $\ell+1$. Then, if $\bar{\boldsymbol{\sigma}}$ is a configuration from the 1d slice on which T_ℓ is built, $\bar{\sigma} \in \mathcal{I}_{\ell-1} \times \mathbb{S}_\ell \times \mathcal{J}_{\ell+1}$, the tensor train can be collapsed telescopically using Eqs. (A.12):

$$\widetilde{F}_{\bar{\boldsymbol{\sigma}}} = \left[A_1^{\bar{\sigma}_1} \cdots A_{\ell-1}^{\bar{\sigma}_{\ell-1}} T_\ell^{\bar{\sigma}_\ell} B_{\ell+1}^{\bar{\sigma}_{\ell+1}} \cdots B_{\mathcal{L}}^{\bar{\sigma}_{\mathcal{L}}}\right]_{11} = \left[T_\ell^{\bar{\sigma}_\ell}\right]_{\bar{\imath}_{\ell-1}\bar{\jmath}_{\ell+1}} = F_{\bar{\boldsymbol{\sigma}}}\,, \tag{A.13}$$



  This proves that if the pivots of $\widetilde{F}$ are nested w.r.t. T_ℓ, then $\widetilde{F}$ is exact on the slice T_ℓ. It follows that if the pivots of $\widetilde{F}$ are nested w.r.t. *all* T_ℓ, i.e. if they are fully nested (cf. Eq. (37)), then $\widetilde{F}$ is exact on all slices T_ℓ (and their sublices P_ℓ), i.e. on all configurations $\bar{\boldsymbol{\sigma}}$ from which it was built. Hence, a fully nested $\widetilde{F}$ is an interpolation of F.

- *0-Site nesting w.r.t. P_ℓ*: We say that the pivots are *nested w.r.t.* P_ℓ if they are left-nested up to ℓ and right-nested up $\ell+1$. Then, P_ℓ is a subslice of both T_ℓ (since $\mathcal{I}_{\ell-1} < \mathcal{I}_\ell$) and $T_{\ell+1}$ (since $\mathcal{J}_{\ell+1} > \mathcal{J}_{\ell+2}$), and $\widetilde{F}$ is exact on both (by Eq. (A.13)), hence $\widetilde{F}$ is exact on P_ℓ. Moreover, if we view $\widetilde{F}_{\boldsymbol{\sigma}}$, with $\boldsymbol{\sigma} = (i_\ell, j_{\ell+1})$, as a matrix with elements $[\widetilde{F}]_{i_\ell\, j_{\ell+1}}$, then its rank, say r_ℓ, equals the dimension of P_ℓ, i.e. $r_\ell = \chi_\ell$. This matrix rank r_ℓ is an intrinsic property of $\widetilde{F}$: it will stay fixed under all exact manipulations on $\widetilde{F}$, i.e. ones that leave its values on all configurations unchanged, e.g. exact SVDs or exact TCIs.

- *2-Site nesting w.r.t. Π_ℓ*: We say that pivots are *nested w.r.t.* Π_ℓ if they are left-nested up to $\ell-1$ and right-nested up to $\ell+2$. Then, if $\bar{\boldsymbol{\sigma}}$ is a configuration from the 2d slice Π_ℓ, i.e. $\bar{\boldsymbol{\sigma}} \in \mathcal{I}_{\ell-1} \times \mathbb{S}_\ell \times \mathbb{S}_{\ell+1} \times \mathcal{J}_{\ell+2}$ so that $F_{\bar{\boldsymbol{\sigma}}} = [\Pi_\ell]_{\bar{\boldsymbol{\sigma}}}$, the tensor train can be collapsed telescopically to yield $\widetilde{F}_{\bar{\boldsymbol{\sigma}}} = \left[T_\ell^{\bar{\sigma}_\ell} P_\ell^{-1} T_{\ell+1}^{\bar{\sigma}_{\ell+1}}\right]_{\bar{\imath}_{\ell-1},\bar{\jmath}_{\ell+2}}$. On this slice the local error, $\left[\Pi_\ell - T_\ell P_\ell^{-1} T_{\ell+1}\right]_{\bar{\boldsymbol{\sigma}}}$, is therefore equal to the global error, $\left[F - \widetilde{F}\right]_{\bar{\boldsymbol{\sigma}}}$, of the TCI approximation. A local update reducing the local error will thus also reduce the global error (cf. Eq. (40)).

A.4 TCI in the continuum

This entire article is based on the cross interpolation of discrete tensors $F_{\boldsymbol{\sigma}}$. In this appendix, we briefly discuss how this concept can be extended to continuum functions $f(\mathbf{x})$, as alluded to in Sec. 2.2.

Consider the natural TCI representation of a function $f(\mathbf{x})$. Following the notations of Sec. 5.1, we suppose that $f(\mathbf{x})$ has been discretized on a grid $\{\mathbf{x}(\boldsymbol{\sigma})\}$ and is represented by a tensor $F_{\boldsymbol{\sigma}} = f(\mathbf{x}(\boldsymbol{\sigma}))$. Its TCI approximation $\widetilde{F}_{\boldsymbol{\sigma}}$ is constructed from tensors T_ℓ that are slices of $F_{\boldsymbol{\sigma}}$, i.e. with elements $[T_\ell^\sigma]_{i_{\ell-1} j_{\ell+1}}$ given by function values of $f(\mathbf{x}(\boldsymbol{\sigma}))$,

$$[T_\ell^\sigma]_{i_{\ell-1} j_{\ell+1}} = f\big(x_1(\sigma_1), \ldots, x_{\ell-1}(\sigma_{\ell-1}), x_\ell(\sigma), x_{\ell+1}(\sigma_{\ell+1}), \ldots, x_{\mathcal{L}}(\sigma_{\mathcal{L}})\big)\,. \tag{A.14}$$

In order to extend the associated TCI form to the continuum, we can simply extend $x_\ell(\sigma)$ to new values. In other words, one may perform the TCI on a grid and evaluate the obtained MPS on another, larger, grid. Formally, one simply replaces the matrix T_ℓ^σ by a matrix $T_\ell(x)$ defined as

$$[T_\ell(x)]_{i_{\ell-1} j_{\ell+1}} = f\big(x_1(\sigma_1), \ldots, x_{\ell-1}(\sigma_{\ell-1}), x, x_{\ell+1}(\sigma_{\ell+1}), \ldots, x_{\mathcal{L}}(\sigma_{\mathcal{L}})\big)\,. \tag{A.15}$$

The obtained MPS $\tilde{f}(\mathbf{x}) = T_1(x_1) P_1^{-1} T_2(x_2) P_2^{-1} \cdots T_n(x_n)$ can be evaluated for any $\mathbf{x}$ in the continuum. In practice, it may be convenient to write the matrices $T_\ell(x)$ as an expansion over, say, Chebychev polynomials. This can be readily done if the initial grid is constructed from the corresponding Chebychev roots.

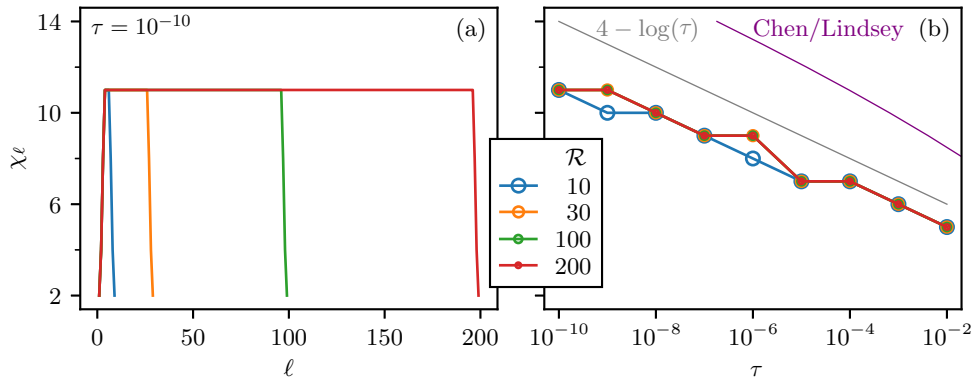


Figure 14: Bond dimensions of the discrete Fourier transform in quantics representation. (a) Bond dimensions χ_ℓ along the MPS. Different colors signify different number of bits $2^{\mathcal{R}}$. Except at both ends of the MPS, the bond dimension χ_ℓ is constant with a value independent of $\mathcal{R}$. (b) Dependence of the maximum bond dimension χ on the tolerance τ. Bond dimensions increase logarithmically with decreasing tolerance (gray curve), and are independent of $\mathcal{R}$. The error bound from Ref. [66] is shown for comparison (purple curve).

A.5 Small rank of the quantics Fourier transform

There is an intuitive explanation of the fact that we need to reverse the ordering of the indices of k with respect to those of m: large-scale properties in real space (big shifts of m, associated with changes of σ_ℓ with $\ell \approx 1$) correspond to the Fourier transform at small momentum k (i.e. changes of σ'_ℓ with $\ell \approx \mathcal{R}$), and indices that relate to the same scales should be fused together. More technically, the fact that the scale-reversed encodings (75) yield a tensor T_μ of low rank stems from the factor $2^{\mathcal{R}-\ell'-\ell}$ in its phase. This factor is an integer for $\mathcal{R}-\ell' \geq \ell$ and $\simeq 0$ for $\mathcal{R}-\ell' \ll \ell$, hence $\exp[-i2\pi 2^{\mathcal{R}-\ell'-\ell}\sigma'_{\ell'}\sigma_\ell] = 1$ or $\simeq 1$, respectively, irrespective of the values of $\sigma'_{\ell'}$ and σ_ℓ. Therefore, $T_{\boldsymbol{\sigma}'\boldsymbol{\sigma}}$ has a strong dependence on the index combinations $(\sigma'_{\ell'}, \sigma_\ell)$ only if neither of the above-mentioned inequalities apply, i.e. only if $\mathcal{R}-\ell'+1$ is equal to or just slightly smaller than ℓ; in this sense, the dependence of $T_{\boldsymbol{\sigma}'\boldsymbol{\sigma}}$ on $|(\mathcal{R}-\ell'+1)-\ell|$ is rather short-ranged. This is illustrated in Figure 15 by the color-scale plot of $(2^{\mathcal{R}-\ell'-\ell}) \bmod 1$ as a function of ℓ and $\mathcal{R}-\ell'+1$: only a small set of coefficients is not close to an integer, namely those on or slightly below the diagonal, where $\mathcal{R}-\ell'+1 = \ell$ or $\lesssim \ell$. This is the reason for defining μ_ℓ as $(\sigma'_{\mathcal{R}-\ell+1}, \sigma_\ell)$, not $(\sigma'_\ell, \sigma_\ell)$. Then, tensor train unfoldings $\widetilde{T}_\mu$ of T_μ involve, in quantum information parlance, only *short-range entanglement* and have low rank [21, 58].

To show explicitly that TCI is able to find this low-rank representation, numerical experiments are shown in Fig. 14. The resulting tensor train has a rank of $\chi = 11$ for a tolerance of $\tau = 10^{-10}$, independent of $\mathcal{R}$. With decreasing tolerance, we observe that the bond di-

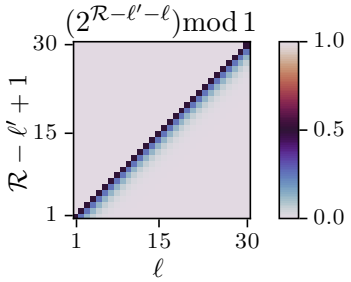


Figure 15: Fractional part of the phase factor of the Fourier transform.

mension increases slightly slower than logarithmically. This is similar to the results found in Refs. [21,58,59] for SVD-based truncation, and below the error bound obtained by Chen and Lindsey in Ref. [66].

B Code listings of examples discussed in the text

All the examples discussed in the text are associated with a runnable script (in one or more language) that can be found in the supplementary materials. Below, we show the most important parts of these scripts.

B.1 Python scripts

B.1.1 Integration of multivariate functions in environment mode

In the environment mode discussed in section 4.3.7, the TCI factorization aims to minimize the error of the integral (whereas with the usual bare mode, it aims to minimize the error on the intergrand). In `xfac`, the environment mode is switched on when the `CTensorCI()` class is instanciated with weights $w_\ell(\sigma_\ell)$. Providing these weights actually triggers two things: the activation of the environment mode and the fact that one uses the weighted unfolding Eq. (63). To perform the computation in environment mode, simply replace the call to `CTensorCI()` in line 19 of code Listing 1 by the lines of code in Listing 6.

```
# TCI1 Tensor factorization in "environment mode"
par = xfacpy.TensorCIParam()
par.weight = [well] * N
tci = xfacpy.CTensorCI(f, [xell] * N, par)
```

Listing 6: Python code snippet to perform a TCI factorization in environment mode combined with weighted unfolding Eq. (63) with weights $\{w_\ell(\sigma_\ell)\}$. To activate environment mode, the weights are passed to the `CTensorCI()` class using the attribute `weights` of the optional parameter `par`, which itself is an instance of class `xfacpy.TensorCIParam()`. The weights `well` must be provided for each of the `N` legs of the tensor. In this example we chose the weights to be identical for each leg (Gauss–Kronrod weights) and use the shorthand notation `[well] * N`, to generate a list of `N` lists of weights. Note that above code is generic and works similarly with all other TCI classes.

B.1.2 Quantics for 2-dimensional integration

Listing 7 shows the quantics unfolding of the 2D function defined in Eq. (79) (see Fig. 8 in Sec. 6.3.1). Here, the variables x and y are both discretized onto grids of $M = 2^{\mathcal{R}}$ points each, with $\mathcal{R} = 40$. The corresponding MPS $\widetilde{F}_\sigma$ has $\mathcal{L} = 2\mathcal{R}$ indices, interleaved so that even indices $\sigma_{2\ell}$ encode x and odd indices $\sigma_{2\ell+1}$ encode y. Lines 9–18 define conversions between grid indices and interleaved quantics indices. Lines 21–28 then define the function from Eq. (79) and the corresponding tensor in quantics representation, which is then TCI-unfolded using `xfac` in lines 31–38.

```
import xfacpy
import numpy as np

R = 40  # number of bits
```

```
M = 2**R  # number of grid points per variable
xmin, xmax, ymin, ymax = -5, +5, -5, +5  # domain of function f


def m_to_sigma(m_x, m_y):  # convert grid index (m_x,m_y) to quantics multi-index sigma
    b1, b2 = np.binary_repr(m_x, width=R), np.binary_repr(m_y, width=R)
    return np.ravel(list(zip(b1, b2))).astype('int')


def sigma_to_xy(sigma):  # convert quantics multi-index sigma to grid point (x,y)
    m_x, m_y = int(''.join(map(str, sigma[0::2])), 2), int(
        ''.join(map(str, sigma[1::2])), 2)
    x, y = xmin + m_x*(xmax-xmin)/M, ymin + m_y*(ymax-ymin)/M
    return x, y


def f(x, y):
    return (np.exp(-0.4*(x**2+y**2))+1+np.sin(x*y)*np.exp(-x**2) +
            np.cos(3*x*y)*np.exp(-y**2)+np.cos(x+y))


def f_tensor(sigma):  # quantics tensor
    x, y = sigma_to_xy(sigma)
    return f(x, y)


# load default parameters for initializing tci object T(1/P)T(1/P)T...
p = xfacpy.TensorCI1Param()
p.pivot1 = [0 for ind in range(2*R)]  # set first pivot to sigma=(0,0,...0)
# use first pivot to initialize tci
f_tci = xfacpy.TensorCI1(f_tensor, [2]*2*R, p)

for sweep in range(40):
    f_tci.iterate()  # perform half-sweep

f_tt = f_tci.get_TensorTrain()  # get a TT object MMMM...
print("x\t f(x)\t f_tt(x)")

# evaluate the approximation on some regularly spaced points
for m_x in range(0, M, 2**(R-5)):
    for m_y in range(0, M, 2**(R-5)):
        sigma = m_to_sigma(m_x, m_y)
        x, y = xmin + (xmax-xmin)*m_x/M, ymin + (ymax-ymin)*m_y/M
        print(f"{x}\t{y}\t{f(x,y)}\t{f_tt.eval(sigma)}")
```

Listing 7: Python code, using `xfac` and `TensorCI1` to compute the quantics approximation of the 2D-function $f(x,y)$ of Eq. (79) for x and y between -5 and 5 using $2^{40} \times 2^{40}$ grid points, plotted in Fig. 8.

B.1.3 Quantics for multi-dimensional integration

```
import xfacpy
from math import log

N = 5
xmin, xmax = 0.0, 1.0
R = 40  # Number of bits


def f(x):  # Integrand function
    f.neval += 1
    return 2**N / (1 + 2 * sum(x))


f.neval = 0

# Exact integral value in 5 dimensions
i5 = (- 65205 * log(3) - 6250 * log(5) + 24010 * log(7) + 14641 * log(11)) / 24
```

```
18
19 # Define the multidim quantics grid
20 grid = xfacpy.QuanticsGrid(a=xmin, b=xmax, nBit=R, dim=N, fused=False)
21
22
23 def fq(sigma):  # Integrand function on quatics grid
24     return f(grid.id_to_coord(sigma))
25
26
27 # TCI2 Tensor factorization
28 tci = xfacpy.TensorCI2(fq, [grid.tensorLocDim] * grid.tensorLen)
29
30 # Estimate integral and error
31 for hsweep in range(14):
32     tci.iterate()
33     # calculate the integal over the hypercube
34     itci = tci.tt.sum1()*grid.deltaVolume
35     print("hsweep= {}, neval= {}, I_tci= {:e}, |I_tci - I_exact|= {:e}, in-sample err=
           {:e}"
36           .format(hsweep+1, f.neval, itci, abs(itci - i5), tci.pivotError[-1]))
```

Listing 8: Python code to compute the integral $I^{(\mathcal{N}=5)}$ of Eq. (64) numerically using the multi-dimensional quantics integration from section 6.3.2. The integrand is formally discretized on 2^{40} points per variable x_n, while the factorization is performed on a multi-dimensional quantics representation using a tensor of $2^{5\times 40}$ legs holding 2 sites each. The mapping between the original coordinate space $\mathbf{x}$ and the quantics representation is performed with the helper class `xfacpy.QuanticsGrid()`. The maximal bond dimension is 30 (default value of `xfacpy.TensorCI2`).

Listing 8 contains the code to compute the multi-dimensional integral Eq. (64) using quantics. The code is very similar to Listing 1, but replaces the Gauss–Kronrod helper functions with corresponding functions for a quantics grid. The helper class `xfacpy.QuanticsGrid()` in line 16 performs the mapping between the original coordinate space $(x_1, \ldots, x_\mathcal{N})$ and the quantics representation. `a=0` and `b=1` specify the bounds of the integration interval, the dimension is $\mathcal{N} = 5$ and we have chosen $\mathcal{R} = 40 \equiv$ `nBit`. The last argument `fused=False` indicate that the variable should not be fused, as described above.

The function `fq(sigma)` defined in line 18 and 19 evaluates the integrand function `f(x)` in the quantics representation. The method `QuanticsGrid.id_to_coord(sigma)` provides the mapping from the index position $\boldsymbol{\sigma}$ in the interleaved representation, written in terms of a binary number, onto the corresponding point $(x_1, \ldots, x_\mathcal{N})$ in the original argument space of the function `f(x)`. In our implementation $\boldsymbol{\sigma}$ is a Python list consisting of $2^{\mathcal{R}\mathcal{N}}$ binary elements, each either 0 or 1. The first or last element of $\boldsymbol{\sigma}$ represents the left-most or right-most bit, respectively. The tensor is instantiated in line 22. We have chosen TCI2 in this example as opposed to TCI1 in Listing 1, to demonstrate that both implementations of TCI1 and TCI2 are easily interchanged as their interfaces are similar. The overall result will be similar in both cases. The second argument of `TensorCI2`, namely `[grid.tensorLocDim] * grid.tensorLen`, creates a list `[2, 2, 2, ...]` with 200 elements, where each element is equal to 2 (our tensor has $\mathcal{N}\mathcal{R} = 200$ legs, with dimension 2 per leg). The rest of the script is similar to Listing 1, printing the result and the error for each iteration of the TCI algorithm.

B.1.4 Heat equation using superfast Fourier transforms

```
1 import numpy as np
2 import xfacpy
3
4 # Grid parameters
5 R = 30
6 M = 2**R
```

```
xmin, xmax = 2., 8.

# Manipulation of indices and grid
def m_to_sigma(m): # convert grid index to quantics multi-index
    return [int(k) for k in np.binary_repr(m, width=R)]

def sigma_to_m(sigma): # reversed transform
    return int(''.join(map(str, sigma)), 2)

def sigma_to_x(sigma): # convert quantics multi-index to position
    return xmin + (xmax-xmin) * sigma_to_m(sigma) / 2**R

# Contraction
def contract_tt_MPO_MPS(tt_mpo, tt_mps):
    mpo = tt_mpo.core
    mps = tt_mps.core
    res = xfacpy.TensorTrain_complex(len(mpo))
    for i in range(len(mpo)):
        aux = np.reshape( mpo[i], (mpo[i].shape[0], 2, 2, mpo[i].shape[2]))
        m = np.tensordot(mps[i], aux, axes=([1], [2]))
        m = np.transpose(m, (0, 2, 3, 1, 4))
        newshape = (mps[i].shape[0]*mpo[i].shape[0], 2,
                    mps[i].shape[2]*mpo[i].shape[2])
        m = np.reshape(m, newshape)
        res.setCoreAt(i, m)
    res.compressSVD()
    return res

# TCI
def build_TCI2_complex(fun, d, pivot1, pivots):
    p = xfacpy.TensorCI2Param()
    p.pivot1 = pivot1
    p.useCachedFunction = True
    p.fullPiv = True
    ci = xfacpy.TensorCI2_complex(fun, [d]*R, p)
    ci.addPivotsAllBonds(pivots)

    nsweep = 3
    for chi in [4,8,16,32,64]:
        ci.param.bondDim = chi
        for i in range(1, nsweep+1):
            ci.iterate()
            rank = np.max([x.shape[2] for x in ci.tt.core])
        if (rank < chi) or (ci.pivotError[-1] < 1e-10):
            break
    return ci.tt


# Fourier transform MPOs
def qft(mu):
    m1 = sigma_to_m( [mu[i]%2 for i in range(R)] )
    m2_swapped = sigma_to_m(reversed( [mu[i]//2 for i in range(R)] ))
    res = 1/(2**(R/2)) * np.exp(-1j * 2*np.pi * m1 * m2_swapped / 2**R)
    return res
qft_mpo = build_TCI2_complex(qft, 4, pivot1=[3]*R, pivots=[])

def iqft(mu):
    m1_swapped = sigma_to_m(reversed( [mu[i]%2 for i in range(R)] ))
    m2 = sigma_to_m( [mu[i]//2 for i in range(R)] )
    res = 1/(2**(R/2)) * np.exp(1j * 2*np.pi * m1_swapped * m2 / 2**R)
    return res
iqft_mpo = build_TCI2_complex(iqft, 4, pivot1=[3]*R, pivots=[])


# Initial temperature distribution
def u0(x):
    door = np.where(abs(x-5) <= 1.5, 1, 0)
    oscillations = (1 + np.cos(120*x) * np.sin(180*x))
    return door + 0.01 * oscillations

# Quantics representation of u0
def u0_tensor(sigma):
```

```
79      return u0(sigma_to_x(sigma))
80  pivot1= [np.random.randint(2) for i in range(R)]
81  while u0_tensor(pivot1) == 0:
82      pivot1 = [np.random.randint(2) for i in range(R)]
83  pivots = [m_to_sigma(m) for m in [0, M//4, M//2-2, M//2, 3*M//4, M-1]]
84  u0_mps = build_TCI2_complex(u0_tensor, 2, pivot1, pivots)
85
86
87  # Time propagator of the Heat equation in Fourier Space
88  def heat_kernel(sigma,t):
89      k = sigma_to_m(reversed(sigma) ) # work with swapped bits in Fourier space
90      delta = (xmax - xmin) / M
91      g_k = np.exp(- (2/delta * np.sin(np.pi * k / M))**2 * t)
92      return g_k
93
94  # MPO representation of the Heat kernel
95  def build_mpo_heat_kernel(t):
96      # build a Quantics MPS
97      pivot1 = [0]*R
98      heat_kernel_t = lambda sigma : heat_kernel(sigma, t)
99      heat_mps =  build_TCI2_complex(heat_kernel_t, 2, pivot1, pivots)
100     # convert to a diagonal MPO
101     heat_mps = heat_mps.core
102     res = xfacpy.TensorTrain_complex(R)
103     for i in range(R):
104         s = heat_mps[i].shape
105         aux = np.zeros((s[0],4,s[2]),dtype='complex')
106         aux[:,0,:] = heat_mps[i][:,0,:]
107         aux[:,3,:] = heat_mps[i][:,1,:]
108         res.setCoreAt(i,aux)
109     return res
110
111
112
113 # Time evolution
114 ft_u0 = contract_tt_MPO_MPS(qft_mpo,u0_mps)
115 ts = [5e-6, 0.0001, 0.01, 0.25, 1] # times list
116 samples_lists = []
117 m_list = [m for m in range(0,M,2**(R-4))]
118 x_list = [xmin + (xmax-xmin)/M * m for m in m_list]
119 for t in ts:
120     heat_k_mpo = build_mpo_heat_kernel(t)
121     ft_ut = contract_tt_MPO_MPS(heat_k_mpo, ft_u0)
122     ut = contract_tt_MPO_MPS(iqft_mpo, ft_ut)
123     samples_lists.append([np.real(ut.eval(m_to_sigma(m))) for m in m_list])
124
125 # print the evolution of temperature on some regularly spaced points
126 print(*(['x\t'] + [f'u(x,{t})' for t in ts]), sep='\t')
127 for i,x in enumerate(x_list):
128     print(*([f'{x:.3f}'] + [f'\t{samples_lists[j][i]:.3f}' for j in range(5)]),
129           sep='\t')
```

Listing 9: Python code using `TensorCI2` and the quantics representation defined in section 6 to build a superfast Fourier transform and solve the heat equation on a billion points grid, as shown in Fig. 11.

Listing 9 shows the code to solve the heat equation (80) on a 2^{30} points grid using quantics and the ultrafast Fourier transform MPO representation, as described in section 6.2.

The `contract_tt_MPO_MPS` defined line 20 performs the contraction of an MPO with an MPS. The `build_TCI2_complex` function defined line 36 calls TCI to build either a MPS when the second argument is $d = 2$ or an MPO when $d = 4$. We define an MPO as a tensor with dimension 4 per leg by fusing the input an output indices σ, σ' following: $\mu = 2\sigma' + \sigma$.

The Fourier and inverse Fourier transform MPO representations are defined line 56 and 63. The initial temperature distribution (85) is defined line 72 and mapped to a quantics MPS. The `build_mpo_heat_kernel` method line 95 builds the MPO representation of the heat kernel operator (83) to perform time evolution in Fourier space for a given time t.

The final temperature distribution is then computed at 5 different times following (84a) and (84b). The code prints a temperature values on some regularly spaced grid points for visualization.

B.2 C++ code

B.2.1 Computation of partition functions

Listing 10 shows the C++ code to compute the partition function using TCI2 for classical Ising model with $|\ell - \ell'|^{-2}$ interaction detailed in Eq. (66). We increase the maximum bond dimension by `incD` (=5) step by step until the error is below the tolerance (=10^{-10}).

```
#include <iostream>
#include <iomanip>
#include <vector>
#include <cmath>
#include "xfac/tensor/tensor_ci_2.h"

using namespace std;
using namespace xfac;

// function for compute energy for given spin configuration
double energy(vector<double> const& spin, vector<double> const& cpln, vector<int> const& config){
  const int len= config.size();
  vector<double> vecS(len);
  transform(config.begin(), config.end(), vecS.begin(), [&spin](int i) {return spin.at(i);});

  double sum2 = 0;
  for (int ii=0; ii<len; ii++) {
    const double si = vecS.at(ii);
    for (int jj=ii+1; jj<len; jj++) {
      const double sj = vecS.at(jj);
      sum2 += -(si*sj) * cpln.at(jj-ii-1);
    }
  }
  return sum2;
}

int main(int argc, char *argv[]){
  vector<double> spin = {-1,1}; // down:-1; up:+1
  const double beta = 0.6;      // inverse temperature
  const double len = 32;        // system size

  // cpln: coupling constant is |i-j|^(-2)
  vector<double> cpln(len-1);
  iota(cpln.begin(), cpln.end(), 1);
  for_each(cpln.begin(), cpln.end(), [] (double& val) {
    val = pow(val,-2);
  });

  // TT parameters
  const int niter = 100; // # of tci sweeps
  const int minD = 5;    // minimal bond dimension
  const int incD = 5;    // increment of bond dimension
  const int dim = spin.size(); //dim of the local space

  // Define partition function
  long count = 0;
  auto prob=[=,&spin,&beta,&count](vector<int> const& config) {
    count++;
    return exp( -beta * energy(spin, cpln, config) );
  };

  // Initialize TCI
  TensorCI2Param pp;
  pp.bondDim = minD;
  auto tci = TensorCI2<double>(prob, vector(len,dim),pp);
```

```
56
57    // Initialize PIVOTS
58    auto init1 = vector(len, 1);
59    auto init2 = vector(len, 0);
60    vector<vector<int>> seed = {init1,init2};
61    tci.addPivotsAllBonds(seed);
62
63    // TCI sweep
64    for (int iter=0; iter<niter; iter++){
65      tci.iterate();
66      cout << setw( 6) << fixed << iter << " "
67           << setw( 6) << fixed << tci.param.bondDim << " "
68           << setw(12) << fixed << count << " "
69           << setw(20) << scientific << setprecision(4) <<
                 tci.pivotError.back()/tci.pivotError.front() << " "
70           << endl;
71      if (tci.pivotError.back() / tci.pivotError.front() <1e-10) {
72        break;
73      }
74      tci.pivotError.clear();
75      tci.param.bondDim += incD;
76    }
77
78    // Measure local moments
79    vector<vector<double>> ones = vector(len, vector(dim,1.0));
80    vector<double> m2(dim);
81    transform(spin.begin(), spin.end(), m2.begin(), [&len](double i) {return pow(i,2);});
82    const double norm = tci.tt.sum(ones);
83    // compute <M>
84    double aM1 = 0;
85    for (int ss=0; ss<len; ss++){
86      auto tmp = ones;
87      tmp.at(ss) = spin;
88      aM1 = aM1 + tci.tt.sum(tmp);
89    }
90    // compute <M^2>
91    double aM2 = 0;
92    for (int s1=0; s1<len; s1++){
93      for (int s2=s1+1; s2<len; s2++){
94        auto tmp = ones;
95        tmp.at(s1) = spin;
96        tmp.at(s2) = spin;
97        aM2 = aM2 + 2*tci.tt.sum(tmp);
98      }
99    }
100   for (int ss=0; ss<len; ss++){
101     auto tmp = ones;
102     tmp.at(ss) = m2;
103     aM2 = aM2 + tci.tt.sum(tmp);
104   }
105
106   // Print resutls
107   const double FE = log(norm)/len; //free energy
108   const double M1 = aM1/norm/len;
109   const double M2 = aM2/norm/len/len;
110   cout << "Beta: "
111        << setw( 6) << fixed << setprecision(2) << beta
112        << " | Free Energy: "
113        << setw(20) << fixed << setprecision(16) << FE
114        << " | M1: "
115        << setw(12) << fixed << setprecision(8)  << M1
116        << " | M2: "
117        << setw(12) << fixed << setprecision(8)  << M2
118        << " | # calls: " << setw(12) << fixed << count
119        << endl;
120   return 0;
121 }
```

Listing 10: C++ code to compute the partition function for classical Ising model with $|\ell - \ell'|^{-2}$ interactions; see Eq. (66).

B.3 Julia scripts

B.3.1 TCI for high-dimensional Gauss–Kronrod quadrature

Listing 11 contains the Julia script for numerical integration of Eq. (64) using TCI for $\mathcal{N} = 5$ using bare error estimate (refer to Sec. 4.3.7).

```
import TensorCrossInterpolation as TCI

N = 5               # Number of dimensions <@$\cN$@>
tolerance = 1e-10   # Tolerance of the internal TCI
GKorder = 15        # Order of the Gauss-Kronrod rule to use

f(x) = 2^N / (1 + 2 * sum(x))   # Integrand
integralvalue = TCI.integrate(Float64, f, fill(0.0, N), fill(1.0, N); tolerance, GKorder)

# Exact value of integral for <@$\cN = 5$@>
i5 = (-65205 * log(3) - 6250 * log(5) + 24010 * log(7) + 14641 * log(11)) / 24
error = abs(integralvalue - i5)

@info "TCI integration with GK$GKorder: " integralvalue i5 error
```

Listing 11: Julia code to numerically compute the integral $I^{(\mathcal{N}=5)}$ of Eq. (64) using `TCI.jl`. Results are shown in Figs. 4.

B.3.2 Quantics TCI for 2-dimensional integration

Listing 12 shows the Julia script for the Julia code to compute a quantics TCI of the 2D-function $f(x, y)$ of Eq. (79) for x and y between -5 and 5 using $\mathcal{R} = 40$. In practice, we use `QuanticsTCI.jl`, which is a thin wrapper around `TCI.jl` that provides a more user-friendly interface for quantics TCI.

```
using QuanticsTCI
import QuanticsGrids as QG

R = 40  # Number of bits <@$\cR$@>
xygrid = QG.DiscretizedGrid{2}(R, (-5.0, -5.0), (5.0, 5.0)) # Discretization grid
    <@$\vec{x}(\bsigma)$@>

function f(x, y) # Function of interest <@$f(x)$@>
    return exp(-0.4*(x^2 + y^2)) + 1 + sin(x * y) * exp(-x^2) +
        cos(3*x*y) * exp(-y ^ 2) + cos(x+y)
end

# Construct and optimize quantics TCI <@$\tF_\bsigma$@>
f_tci, ranks, errors = quanticscrossinterpolate(Float64, f, xygrid; tolerance=1e-10)

# Print a table to compare <@$f(x)$@> and <@$\tF_\bsigma$@> on some regularly spaced
    points
println("x\t y\t f(x)\t\t\t f_tt(x)")
for index in CartesianIndices((10, 10))
    m = Tuple(index) .* div(2^R, 10)
    x, y = QG.grididx_to_origcoord(xygrid, m)
    println("$x\t$y\t$(f(x, y))\t$(f_tci(m))")
end

println("Value of the integral: $(integral(f_tci))")
```

Listing 12: Julia code to compute a quantics TCI of the 2D-function $f(x, y)$ of Eq. (79) for $x, y \in [-5, 5)$ using $2^{40} \times 2^{40}$ grid points, plotted in Fig. 8.

B.3.3 Quantics TCI for multi-dimensional integration

Listing 13 shows the Julia script to compute the integral $I^{(\mathcal{N}=5)}$ [Eq. (64)] numerically using the multi-dimensional quantics integration from Sec. 6.3.2. The code is equivalent to the Python script in Listing 8.

```
import QuanticsGrids as QG
import TensorCrossInterpolation as TCI

N = 5               # Number of dimensions <@$\cN$@>
tolerance = 1e-10   # Tolerance of the internal TCI
R = 40              # Number of bits <@$\cR$@>

f(x) = 2^N / (1 + 2 * sum(x))    # Integrand <@$f(\vec{x})$@>

# Discretization grid with <@$2^{\scN \scR}$@> points
grid = QG.DiscretizedGrid{N}(R, Tuple(fill(0.0, N)), Tuple(fill(1.0, N)),
    unfoldingscheme=:interleaved)
quanticsf(sigma) = f(QG.quantics_to_origcoord(grid, sigma)) # <@$f(\vec{x}(\bsigma))$@>

# Obtain the QTCI representation and evaluate the integral via factorized sum
tci, ranks, errors = TCI.crossinterpolate2(Float64, quanticsf, QG.localdimensions(grid);
    tolerance)

# Integral is sum multiplied with discretization volumne
integralvalue = TCI.sum(tci) * prod(QG.grid_step(grid))

# Exact value of integral for <@$\cN = 5$@>
i5 = (-65205 * log(3) - 6250 * log(5) + 24010 * log(7) + 14641 * log(11)) / 24
error = abs(integralvalue - i5)     # Error for <@$\cN = 5$@>

@info "Quantics TCI integration with R=$R: " integralvalue i5 error
```

Listing 13: Julia code to compute the integral $I^{(\mathcal{N}=5)}$ of Eq. (64) numerically using the multi-dimensional quantics integration from Sec. 6.3.2. The integrand is formally discretized on 2^{40} points per variable x_n, the factorization is performed on a multi-dimensional quantics representation of a tensor of $2^{5\times 40}$ elements.

B.3.4 Compressing existing data with TCI

In the example below, we illustrate how to apply (Q)TCI to some existing typical datasets. Let `dataset` be some pre-generated dataset (e.g. read from a file) in the form of an $\mathcal{N}$-dimensional array. Listing 14 shows a test for TCI compressibility. Listing 15 shows a similar test for QTCI compressibility.

```
import TensorCrossInterpolation as TCI

# Replace this line with the dataset to be tested for compressibility.
grid = range(-pi, pi; length=200)
dataset = [cos(x) + cos(y) + cos(z) for x in grid, y in grid, z in grid]

# Construct TCI
tolerance = 1e-5
tt, ranks, errors = TCI.crossinterpolate2(
    Float64, i -> dataset[i...], collect(size(dataset)), tolerance=tolerance)

# Check error
ttdataset = [tt([i, j, k]) for i in axes(grid, 1), j in axes(grid, 1), k in axes(grid, 1)]
errors = abs.(ttdataset .- dataset)
println(
    "TCI of the dataset with tolerance $tolerance has link dimensions
        $(TCI.linkdims(tt)), "
    * "for a max error of $(maximum(errors))."
)
```

Listing 14: Julia code to test an existing dataset for TCI compressibility.

```
using QuanticsTCI
import TensorCrossInterpolation as TCI

# Number of bits
R = 8

# Replace with your dataset
grid = range(-pi, pi; length=2^R+1)[1:end-1] # exclude the end point
dataset = [cos(x) + cos(y) + cos(z) for x in grid, y in grid, z in grid]

# Perform QTCI
tolerance = 1e-5
qtt, ranks, errors = quanticscrossinterpolate(
    dataset, tolerance=tolerance, unfoldingscheme=:fused)

# Check error
qttdataset = [qtt([i, j, k]) for i in axes(grid, 1), j in axes(grid, 1), k in axes(grid,
    1)]
error = abs.(qttdataset .- dataset)
println(
    "Quantics TCI compression of the dataset with tolerance $tolerance has " *
    "link dimensions $(TCI.linkdims(qtt.tci)), for a max error of $(maximum(error))."
)
```

Listing 15: Julia code to test an existing dataset for QTCI compressibility.

B.3.5 Adding global pivots

We provide a simple example demonstrating the ergodicity problem discussed in Sec. 4.3.5 and how to fix it by adding a global pivot. We consider a function that takes a finite value at the first and last grid points, but is zero elsewhere (see Fig. 16):

$$f_m = \delta_{m,0} + \delta_{m,M-1}, \tag{B.1}$$

where $m = 0, 1, \cdots, M-1$ and $M = 2^{\mathcal{R}}$. When we interpolate this function using a 2-site TCI in the quantics representation with an initial pivot $\boldsymbol{\sigma} = (0, 0, \cdots, 0)$ ($m = 0$), the interpolation fails to capture the function at the last grid point for $\mathcal{R} \geq 3$ [67]. We can fix this by adding a global pivot at the last grid point. Listing 16 shows the Julia code to demonstrate this.

```
import TensorCrossInterpolation as TCI
import Random
import QuanticsGrids as QD
using PythonPlot: pyplot as plt
import PythonPlot
using LaTeXStrings

PythonPlot.matplotlib.rcParams["font.size"] = 15

# Number of bits
R = 4
tol = 1e-4

# f(q) = 1 if q = (1, 1, ..., 1) or q = (2, 2, ..., 2), 0 otherwise
f(q) = (all(q .== 1) || all(q .== 2)) ? 1.0 : 0.0

localdims = fill(2, R)

# Perform TCI with an initial pivot at (1, 1, ..., 1)
firstpivot = ones(Int, R)
tci, ranks, errors = TCI.crossinterpolate2(
    Float64,
    f,
    localdims,
    [firstpivot];
```

```
    tolerance=tol,
    nsearchglobalpivot=0 # Disable automatic global pivot search
)

# TCI fails to capture the function at (2, 2, ..., 2)
globalpivot = fill(2, R)
@assert isapprox(TCI.evaluate(tci, globalpivot), 0.0)

# Add (2, 2, ..., 2) as a global pivot
tci_globalpivot = deepcopy(tci)
TCI.addglobalpivots2sitesweep!(
    tci_globalpivot, f, [globalpivot],
    tolerance=tol
)
@assert isapprox(TCI.evaluate(tci_globalpivot, globalpivot), 1.0)

# Plot the function and the TCI reconstructions
grid = QD.InherentDiscreteGrid{1}(R)
ref = [f(QD.grididx_to_quantics(grid, i)) for i in 1:2^R]
reconst_tci = [tci(QD.grididx_to_quantics(grid, i)) for i in 1:2^R]
reconst_tci_globalpivot = [tci_globalpivot(QD.grididx_to_quantics(grid, i)) for i in
    1:2^R]

fig, ax = plt.subplots(figsize=(6.4, 3.0))
ax.plot(ref, label="ref", marker="", linestyle="--")
ax.plot(reconst_tci, label="TCI without global pivot", marker="x", linestyle="")
ax.plot(reconst_tci_globalpivot, label="TCI with global pivot", marker="+", linestyle="")
ax.set_xlabel(L"Index $m$")
ax.set_ylabel(L"f_m")
ax.legend(frameon=false)
plt.tight_layout()
fig.savefig("global_pivot.pdf")
```

Listing 16: Julia code demonstrating how to add a global pivot. We first construct a TCI object using 2-site TCI with an initial pivot at the first grid index. This fails to interpolate the function at the last grid index due to the local nature of 2-site TCI. This is fixed by adding a global pivot at the last grid index.

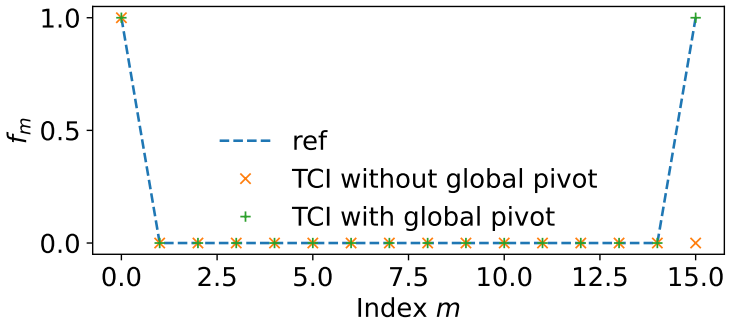


Figure 16: Comparison of the reference function (B.1), the result of TCI without an added global pivot, and the result of TCI with an added global pivot. The global pivot was added at the last grid index.

References

[1] S. R. White, *Density matrix formulation for quantum renormalization groups*, Phys. Rev. Lett. **69**, 2863 (1992), doi:10.1103/PhysRevLett.69.2863.

[2] U. Schollwöck, *The density-matrix renormalization group in the age of matrix product states*, Ann. Phys. **326**, 96 (2011), doi:10.1016/j.aop.2010.09.012.

[3] R. Orús, *A practical introduction to tensor networks: Matrix product states and projected entangled pair states*, Ann. Phys. **349**, 117 (2014), doi:10.1016/j.aop.2014.06.013.

[4] S. Montangero, *Introduction to tensor network methods: Numerical simulations of low-dimensional many-body quantum systems*, Springer, Cham, Switzerland, ISBN 9783030014087 (2018), doi:10.1007/978-3-030-01409-4.

[5] T. Xiang, *Density matrix and tensor network renormalization*, Cambridge University Press, Cambridge, UK, ISBN 9781009398701 (2023), doi:10.1017/9781009398671.

[6] I. Oseledets, S. V. Dolgov, A. Boyko, D. Savostyanov, A. Novikov and T. Mach, *TT-toolbox: Matlab implementation of tensor train decomposition* (2011), https://github.com/oseledets/TT-Toolbox.

[7] I. Oseledets, *ttpy: Python implementation of the TT-toolbox* (2012), https://github.com/oseledets/ttpy.

[8] I. Oseledets and E. Tyrtyshnikov, *TT-cross approximation for multidimensional arrays*, Linear Algebra Appl. **432**, 70 (2010), doi:10.1016/j.laa.2009.07.024.

[9] I. V. Oseledets, *Tensor-train decomposition*, SIAM J. Sci. Comput. **33**, 2295 (2011), doi:10.1137/090752286.

[10] D. Savostyanov and I. Oseledets, *Fast adaptive interpolation of multi-dimensional arrays in tensor train format*, *International workshop on multidimensional (nD) systems*, Poitiers, France, doi:10.1109/nDS.2011.6076873.

[11] D. V. Savostyanov, *Quasioptimality of maximum-volume cross interpolation of tensors*, Linear Algebra Appl. **458**, 217 (2014), doi:10.1016/j.laa.2014.06.006.

[12] S. Dolgov and D. Savostyanov, *Parallel cross interpolation for high-precision calculation of high-dimensional integrals*, Comput. Phys. Commun. **246**, 106869 (2020), doi:10.1016/j.cpc.2019.106869.

[13] Y. N. Fernández, M. Jeannin, P. T. Dumitrescu, T. Kloss, J. Kaye, O. Parcollet and X. Waintal, *Learning Feynman diagrams with tensor trains*, Phys. Rev. X **12**, 041018 (2022), doi:10.1103/PhysRevX.12.041018.

[14] K. Sozykin, A. Chertkov, R. Schutski, A.-H. Phan, A. Cichocki and I. Oseledets, *TTOpt: A maximum volume quantized tensor train-based optimization and its application to reinforcement learning*, in *Advances in neural information processing systems 36*, Curran Associates, Red Hook, USA, ISBN 9781713871088 (2022).

[15] M. K. Ritter, Y. N. Fernández, M. Wallerberger, J. von Delft, H. Shinaoka and X. Waintal, *Quantics tensor cross interpolation for high-resolution parsimonious representations of multivariate functions*, Phys. Rev. Lett. **132**, 056501 (2024), doi:10.1103/PhysRevLett.132.056501.

[16] N. Jolly, Y. N. Fernández and X. Waintal, *Tensorized orbitals for computational chemistry*, (arXiv preprint) doi:10.48550/arXiv.2308.03508.

[17] R. Sakurai, H. Takahashi and K. Miyamoto, *Learning parameter dependence for Fourier-based option pricing with tensor networks*, (arXiv preprint) doi:10.48550/arXiv.2405.00701.

[18] I. V. Oseledets, *Approximation of matrices with logarithmic number of parameters*, Dokl. Math. **80**, 653 (2009), doi:10.1134/s1064562409050056.

[19] B. N. Khoromskij, *$O(d \log N)$-quantics approximation of $N-d$ tensors in high-dimensional numerical modeling*, Constr. Approx. **34**, 257 (2011), doi:10.1007/s00365-011-9131-1.

[20] B. N. Khoromskij, *Tensor numerical methods in scientific computing*, De Gruyter, Berlin, Germany, ISBN 9783110370133 (2018), doi:10.1515/9783110365917.

[21] H. Shinaoka, M. Wallerberger, Y. Murakami, K. Nogaki, R. Sakurai, P. Werner and A. Kauch, *Multiscale space-time ansatz for correlation functions of quantum systems based on quantics tensor trains*, Phys. Rev. X **13**, 021015 (2023), doi:10.1103/PhysRevX.13.021015.

[22] H. Takahashi, R. Sakurai and H. Shinaoka, *Compactness of quantics tensor train representations of local imaginary-time propagators*, SciPost Phys. **18**, 007 (2025), doi:10.21468/SciPostPhys.18.1.007.

[23] H. Ishida, N. Okada, S. Hoshino and H. Shinaoka, *Low-rank quantics tensor train representations of Feynman diagrams for multiorbital electron-phonon models*, (arXiv preprint) doi:10.48550/arXiv.2405.06440.

[24] M. Murray, H. Shinaoka and P. Werner, *Nonequilibrium diagrammatic many-body simulations with quantics tensor trains*, Phys. Rev. B **109**, 165135 (2024), doi:10.1103/PhysRevB.109.165135.

[25] M. Środa, K. Inayoshi, H. Shinaoka and P. Werner, *High-resolution nonequilibrium GW calculations based on quantics tensor trains*, (arXiv preprint) doi:10.48550/arXiv.2412.14032.

[26] N. Gourianov, M. Lubasch, S. Dolgov, Q. Y. van den Berg, H. Babaee, P. Givi, M. Kiffner and D. Jaksch, *A quantum-inspired approach to exploit turbulence structures*, Nat. Comput. Sci. **2**, 30 (2022), doi:10.1038/s43588-021-00181-1.

[27] N. Gourianov, *Exploiting the structure of turbulence with tensor networks*, PhD thesis, University of Oxford, Oxford, UK (2022).

[28] R. D. Peddinti, S. Pisoni, A. Marini, P. Lott, H. Argentieri, E. Tiunov and L. Aolita, *Complete quantum-inspired framework for computational fluid dynamics*, Commun. Phys. **7**, 135 (2024), doi:10.1038/s42005-024-01623-8.

[29] E. Ye and N. F. G. Loureiro, *Quantum-inspired method for solving the Vlasov-Poisson equations*, Phys. Rev. E **106**, 035208 (2022), doi:10.1103/PhysRevE.106.035208.

[30] K. Sakaue, H. Shinaoka and R. Sakurai, *Learning tensor trains from noisy functions with application to quantum simulation*, (arXiv preprint) doi:10.48550/arXiv.2405.12730.

[31] G. H. Golub and C. F. Van Loan, *Matrix computations*, John Hopkins University Press, Baltimore, USA, ISBN 9780801854132 (1996).

[32] A. Erpenbeck et al., *Tensor train continuous time solver for quantum impurity models*, Phys. Rev. B **107**, 245135 (2023), doi:10.1103/PhysRevB.107.245135.

[33] M. Fishman, S. White and E. Stoudenmire, *The ITensor software library for tensor network calculations*, SciPost Phys. Codebases 4 (2022), doi:10.21468/SciPostPhysCodeb.4.

M. Fishman, S. White and E. Stoudenmire, *Codebase release 0.3 for ITensor*, SciPost Phys. Codebases 4-r0.3 (2022), doi:10.21468/SciPostPhysCodeb.4-r0.3.

[34] C.-T. Pan, *On the existence and computation of rank-revealing LU factorizations*, Linear Algebra Appl. **316**, 199 (2000), doi:10.1016/S0024-3795(00)00120-8.

[35] M. Bebendorf, *Approximation of boundary element matrices*, Numer. Math. **86**, 565 (2000), doi:10.1007/PL00005410.

[36] M. Bebendorf and S. Rjasanow, *Adaptive low-rank approximation of collocation matrices*, Computing **70**, 1 (2003), doi:10.1007/s00607-002-1469-6.

[37] M. Bebendorf and R. Grzhibovskis, *Accelerating Galerkin BEM for linear elasticity using adaptive cross approximation*, Math. Methods Appl. Sci. **29**, 1721 (2006), doi:10.1002/mma.759.

[38] M. Bebendorf, *Adaptive cross approximation of multivariate functions*, Constr. Approx. **34**, 149 (2010), doi:10.1007/s00365-010-9103-x.

[39] S. A. Goreinov and E. E. Tyrtyshnikov, *Quasioptimality of skeleton approximation of a matrix in the Chebyshev norm*, Dokl. Math. **83**, 374 (2011), doi:10.1134/S1064562411030355.

[40] J. Schneider, *Error estimates for two-dimensional cross approximation*, J. Approx. Theory **162**, 1685 (2010), doi:10.1016/j.jat.2010.04.012.

[41] E. Tyrtyshnikov, *Incomplete cross approximation in the mosaic-skeleton method*, Computing **64**, 367 (2000), doi:10.1007/s006070070031.

[42] A. Cortinovis, D. Kressner and S. Massei, *On maximum volume submatrices and cross approximation for symmetric semidefinite and diagonally dominant matrices*, Linear Algebra Appl. **593**, 251 (2020), doi:10.1016/j.laa.2020.02.010.

[43] D. E. Crabtree and E. V. Haynsworth, *An identity for the Schur complement of a matrix*, Proc. Am. Math. Soc. **22**, 364 (1969), doi:10.1090/S0002-9939-1969-0255573-1.

[44] C. Brezinski and M. R. Zaglia, *A Schur complement approach to a general extrapolation algorithm*, Linear Algebra Appl. **368**, 279 (2003), doi:10.1016/S0024-3795(02)00686-9.

[45] L. Miranian and M. Gu, *Strong rank revealing LU factorizations*, Linear Algebra Appl. **367**, 1 (2003), doi:10.1016/S0024-3795(02)00572-4.

[46] L. Neal and G. Poole, *A geometric analysis of Gaussian elimination. II*, Linear Algebra Appl. **173**, 239 (1992), doi:10.1016/0024-3795(92)90432-A.

[47] I. V. Oseledets, D. V. Savostianov and E. E. Tyrtyshnikov, *Tucker dimensionality reduction of three-dimensional arrays in linear time*, SIAM J. Matrix Anal. Appl. **30**, 939 (2008), doi:10.1137/060655894.

[48] G. Poole and L. Neal, *The Rook's pivoting strategy*, J. Comput. Appl. Math. **123**, 353 (2000), doi:10.1016/S0377-0427(00)00406-4.

[49] I. V. Oseledets, *Approximation of $2^d \times 2^d$ matrices using tensor decomposition*, SIAM J. Matrix Anal. Appl. **31**, 2130 (2010), doi:10.1137/090757861.

[50] L. Li, W. Yu and K. Batselier, *Faster tensor train decomposition for sparse data*, J. Comput. Appl. Math. **405**, 113972 (2022), doi:10.1016/j.cam.2021.113972.

[51] F. Verstraete and J. I. Cirac, *Renormalization algorithms for quantum-many body systems in two and higher dimensions*, (arXiv preprint) doi:10.48550/arXiv.cond-mat/0407066.

[52] E. M. Stoudenmire and S. R. White, *Minimally entangled typical thermal state algorithms*, New J. Phys. **12**, 055026 (2010), doi:10.1088/1367-2630/12/5/055026.

[53] B.-B. Chen, L. Chen, Z. Chen, W. Li and A. Weichselbaum, *Exponential thermal tensor network approach for quantum lattice models*, Phys. Rev. X **8**, 031082 (2018), doi:10.1103/PhysRevX.8.031082.

[54] J. Burkardt, *Test_nint, multi-dimensional integration test functions* (2012), https://people.math.sc.edu/Burkardt/f_src/test_nint/test_nint.html.

[55] E. Bayong, H. T. Diep and V. Dotsenko, *Potts model with long-range interactions in one dimension*, Phys. Rev. Lett. **83**, 14 (1999), doi:10.1103/PhysRevLett.83.14.

[56] E. Luijten and H. Meßingfeld, *Criticality in one dimension with inverse square-law potentials*, Phys. Rev. Lett. **86**, 5305 (2001), doi:10.1103/PhysRevLett.86.5305.

[57] K. Fukui and S. Todo, *Order-N cluster Monte Carlo method for spin systems with long-range interactions*, J. Comput. Phys. **228**, 2629 (2009), doi:10.1016/j.jcp.2008.12.022.

[58] J. Chen, E. M. Stoudenmire and S. R. White, *Quantum Fourier transform has small entanglement*, PRX Quantum **4**, 040318 (2023), doi:10.1103/PRXQuantum.4.040318.

[59] K. J. Woolfe, C. D. Hill and L. C. L. Hollenberg, *Scale invariance and efficient classical simulation of the quantum Fourier transform*, (arXiv preprint) doi:10.48550/arXiv.1406.0931.

[60] S. Dolgov, B. Khoromskij and D. Savostyanov, *Superfast Fourier transform using QTT approximation*, J. Fourier Anal. Appl. **18**, 915 (2012), doi:10.1007/s00041-012-9227-4.

[61] G. K.-L. Chan, A. Keselman, N. Nakatani, Z. Li and S. R. White, *Matrix product operators, matrix product states, and ab initio density matrix renormalization group algorithms*, J. Chem. Phys. **145**, 014102 (2016), doi:10.1063/1.4955108.

[62] C. Hubig, I. P. McCulloch and U. Schollwöck, *Generic construction of efficient matrix product operators*, Phys. Rev. B **95**, 035129 (2017), doi:10.1103/PhysRevB.95.035129.

[63] J. Ren, W. Li, T. Jiang and Z. Shuai, *A general automatic method for optimal construction of matrix product operators using bipartite graph theory*, J. Chem. Phys. **153**, 084118 (2020), doi:10.1063/5.0018149.

[64] D. E. Parker, X. Cao and M. P. Zaletel, *Local matrix product operators: Canonical form, compression, and control theory*, Phys. Rev. B **102**, 035147 (2020), doi:10.1103/PhysRevB.102.035147.

[65] Y. N. Fernández et al., *Tensor4all*, https://tensor4all.org/.

[66] J. Chen and M. Lindsey, *Direct interpolative construction of the discrete Fourier transform as a matrix product operator*, (arXiv preprint) doi:10.48550/arXiv.2404.03182.

[67] Y. Yu, *Private communication* (2024).

Chapter Four

Implementing diagrammatic methods with tensor trains

4.1 Introduction

As shown in Chapter 2, correlators such as the vertex can be highly complex functions of many arguments. Diagrammatic method thus encounter a *curse of dimensionality* problem, where the amount of memory required to represent some correlators on a dense grid grows exponentially with the number of arguments, which in turn scales with order in perturbation theory, number of bands or orbitals in the Hamiltonian, or other physical quantities [74]. This limits the scope of applicability of these methods. Since TCI is tailored to these curse of dimensionality problems, it is a quite natural idea to apply it to diagrammatic equations as well. This chapter shows how to perform a self-consistent parquet calculation entirely within the quantics TCI formalism.

Prior attempts to reduce the computational complexity of correlator representation below that of a naive dense grid approach are mainly based on basis sets that are designed for the specific quantity to be represented. For Matsubara frequencies, these are the closely related intermediate representation [74–76] and discrete Lehmann representation [77, 78]. The momentum dependency can be expanded in form factors in the truncated unity fRG [79, 80]. Having expanded a particular dependency, it is then possible—and often required—to truncate the basis set, and neglect coefficients that have small values. Ideally, one would systematically increase the number of components taken into account until convergence is reached up to some tolerance. In many cases, this is unfeasible due to the limited amount of memory typically available on current computing clusters. An example for this are vertices in the Hubbard model, where it is typically necessary to approximate the frequency dependence with < 10 components in order to converge the momentum dependence in the number of form factors [79, 81].

This type of approach is often cumbersome due to its inflexibility: each basis set can only be applied to a specific parameter dependence, and finding

a new basis set for a different type of argument requires considerable effort. If an algorithm such as fRG is to be set up in the new basis, it should also be convenient to manipulate the functions and evaluate equations in the new basis set. In comparison to basis sets, TT are much more flexible in their applicability, and manipulations of TT correspond to tensor networks that are often analogous to ones known from the tensor networks literature [P3, 30]. Prior to our work, Refs. [19, 30] showed that TT are a compact, flexible, and practical representation of correlators. In this chapter, we show that diagrammatic self-consistent equations can be solved with QTT. In parallel to us, Refs. [31, 82] showed that DMFT and GW can be implemented efficiently for non-equilibrium systems in the real-frequency Keldysh formalism.

4.2 Publication 4: Two-particle calculations with quantics tensor trains: Solving the parquet equations

In this section, the following publication is reprinted:

P4 *Two-particle calculations with quantics tensor trains: Solving the parquet equations*,
Stefan Rohshap, Marc K. Ritter, Hiroshi Shinaoka, Jan von Delft, Markus Wallerberger, and Anna Kauch,
Physical Review Research **7**, 023087 (2025),
doi:10.1103/PhysRevResearch.7.023087.
Reprinted on pages 120–142.

Two-particle calculations with quantics tensor trains: Solving the parquet equations

Stefan Rohshap,[1,*] Marc K. Ritter,[2,†] Hiroshi Shinaoka,[3] Jan von Delft,[2] Markus Wallerberger,[1] and Anna Kauch[1,‡]
[1]*Institute of Solid State Physics, TU Wien, 1040 Vienna, Austria*
[2]*Arnold Sommerfeld Center for Theoretical Physics, Center for NanoScience, and Munich Center for Quantum Science and Technology, Ludwig-Maximilians-Universität München, 80333 Munich, Germany*
[3]*Department of Physics, Saitama University, Saitama 338-8570, Japan*

(Received 3 December 2024; accepted 11 March 2025; published 24 April 2025)

We present an application of quantics tensor trains (QTTs) and tensor cross interpolation (TCI) to the solution of a full set of self-consistent equations for multivariate functions, the so-called parquet equations. We show that the steps needed to evaluate the equations (Bethe-Salpeter equations, parquet equation, and Schwinger-Dyson equation) can be decomposed into basic operations on the QTT-TCI compressed objects. The repeated application of these operations does not lead to a loss of accuracy beyond a specified tolerance and the iterative scheme converges even for numerically demanding parameters. As examples, we take the Hubbard model in the atomic limit and the single impurity Anderson model, where the basic objects in parquet equations, the two-particle vertices, depend on three frequencies, but not on momenta. The results show that this approach is able to overcome major computational bottlenecks of standard numerical methods. The applied methods allow for an exponential increase of the number of grid points included in the calculations, and a corresponding exponential reduction of the computational error, for a linear increase in computational cost.

DOI: 10.1103/PhysRevResearch.7.023087

I. INTRODUCTION

The understanding of many important excitations of electronic systems—magnons, excitons, or other composite objects—requires understanding correlations at the two-particle level. Two-particle quantities—correlation functions or scattering amplitudes (vertices)—are inherently large objects, with multiple dependencies: If we consider scattering of two particles, the amplitude will depend on the energies, momenta, and spin orbitals of two incoming and two outgoing particles. The number of independent variables can be reduced using conservation laws, but each independent spin-orbital combination still depends on three momenta and three frequencies. Numerical representation of these multivariate functions on uniform grids is very expensive due to the third power scaling of memory in the number of discrete momenta or energies. On the other hand, large ranges are required to faithfully represent complicated structures which the vertices show in all their dependencies [1–3]. When the vertices are themselves variables in diagrammatic equations, as is the case in parquet equations [4–6], the required computation time becomes prohibitive [7]. Several solutions to this problem have been proposed so far, either based on partial reduction of the number of frequency and/or momentum variables that need to be treated on grids [8–12] or based on compact representation of the frequency dependence in a suitable basis [13–16]. The former still do not lead to true dimensional reduction of the full parquet equation problem. The latter are very promising and provide another path to dimensional reduction, alternative to the one described in this paper. Recently, a wavelet-based decomposition for efficiently compressing two-particle quantities has also been proposed [17–19].

In this paper, we present a full computation of the self-consistent solution of parquet equations in the quantics tensor train (QTT) representation. This representation, based on length or energy scale separation, leads to significant dimensional reduction of the problem, removing memory bottlenecks. The computational cost becomes logarithmic in grid size and depends strongly only on the maximum bond dimension, which is small enough in many physics applications. Hence, the overall computational cost is significantly reduced.

The QTT representation of multivariate functions has already been around for a decade or so [20–23], but it was only recently applied to various fields of natural science such as turbulence [24–28], plasma physics [29], quantum chemistry [30], and quantum field theory of the many electron problem [31]. For quantum field theories, the QTT representation provides a compact representation of the space-time dependence of the correlation functions [31]. First many-body calculations of Feynman diagrams with the QTT representation in imaginary time [32] and in nonequilibrium [33] already show the potential of the method. A very favorable scaling of the QTT representation with temperature has been conjectured in Ref. [34]. In parallel, the tensor cross interpolation (TCI)

*Contact author: stefan.rohshap@tuwien.ac.at
†Contact author: ritter.marc@lmu.de
‡Contact author: kauch@ifp.tuwien.ac.at

method was applied to evaluations of diagrams in many-body physics [35–38]. TCI can be combined with the quantics tensor train representation to form QTT+TCI=QTCI, a powerful approach with diverse applications [39].

To apply QTCI to parquet equations, we break down these equations [Bethe-Salpeter equation (BSE), parquet equation, and Schwinger–Dyson equation (SDE)] into basic operations on QTTs, represented by matrix product operators (MPOs). We use MPO-MPO contractions for matrix and elementwise multiplications and construct a new MPO for affine transformations needed to perform channel transformations (variable shifts) occurring in the parquet equation. Our approach scales as $\mathcal{O}(D_{\max}^4 R)$, with maximum bond dimension of $D_{\max}$ and grid size 2^{3R}. The computational cost is only logarithmic in grid size and the main bottleneck is shifted to the maximum bond dimension. We have verified this scaling in two benchmark models: the Hubbard atom and the single impurity Anderson model (SIAM). In both models, the two-particle vertices are fully dynamical (dependent on three frequencies) but local (independent of momentum). In both cases, we empirically find that an overall accuracy $<10^{-3}$ of the full self-consistent solution can be achieved with a bond dimension up to 200 even for challenging parameters close to a divergence line in the Hubbard atom.

This paper is organized as follows. In Sec. II, we introduce the concrete Hamiltonians (Hubbard atom and single impurity Anderson model) for which we will present the results. Further, in Sec. III we first recall definitions of one- and two-particle Green's functions and vertices and set the notations used in the paper. Then we provide in detail the full set of parquet equations that we solve. Additional information on the equations and notations is also provided in Appendix A. In Sec. IV, we introduce quantics tensor trains, the tensor cross interpolation method, and matrix product operators. These techniques are used to construct efficient implementations of the parquet equations in Sec. V and Appendixes B and C. We also provide results for the compression of the vertices and scaling of the bond dimension for each of the operations needed to complete one loop of parquet equations in Sec. V. More details and additional plots can be found in Appendixes D and E. Next, in Sec. VI, we show results for the full self-consistent iterative scheme and its technical limitations (with details also in Appendix F). In the last section, Sec. VII, we conclude and provide an outlook.

II. MODELS

In the current paper, we focus on the solution of equations for two-particle vertices in the (Matsubara) frequency space. Although, in general, the vertices are also dependent on momentum and orbital degrees of freedom, we limit ourselves to simple models for which the vertices depend only on frequency but not on momentum. We present results for two benchmark models: the Hubbard atom, where exact analytical expressions for the vertex functions are known [40], and the single impurity Anderson model [41], where high-quality numerical data is available [3]. The treatment of the frequency dependence of vertices presented in this paper can be directly extended to models with additional orbital and momentum dependencies. The possibility of such extensions will be discussed in Appendix G.

A. Hubbard atom

The Hubbard atom is an extreme simplification of the Hubbard model in which the hopping amplitudes of the electrons between sites are put to zero. Although this is a drastic change, the Hubbard atom represents many of the features of the strong-coupling limit of the Hubbard model [40]. Without hopping, each atom is independent and described by the following Hamiltonian:

$$\hat{\mathcal{H}} = U\hat{n}_\uparrow\hat{n}_\downarrow - \mu(\hat{n}_\uparrow + \hat{n}_\downarrow), \tag{1}$$

with $\hat{n}_\sigma = \hat{c}_\sigma^\dagger \hat{c}_\sigma$ and the fermionic annihilation (creation) operator $\hat{c}_\sigma^{(\dagger)}$ that annihilates (creates) an electron with spin σ. The on-site Coulomb repulsion between two electrons is given by U and the chemical potential is set to $\mu = \frac{U}{2}$ (half filling). The only other energy scale beside U in this model is the temperature T, which we define in the same units as U, setting $k_B \equiv 1$ and $\hbar \equiv 1$.

B. Single-impurity Anderson model

In the SIAM, the interacting atom is not isolated, but coupled to a bath of non-interacting electrons. The SIAM Hamiltonian is [41]

$$\begin{aligned}\hat{\mathcal{H}} = &\sum_{\boldsymbol{k}\sigma} \varepsilon_{\boldsymbol{k}} \hat{c}_{\boldsymbol{k},\sigma}^\dagger \hat{c}_{\boldsymbol{k},\sigma} + \sum_{\boldsymbol{k}\sigma} (V_{\boldsymbol{k}} \hat{c}_{\boldsymbol{k},\sigma}^\dagger \hat{d}_\sigma + V_{\boldsymbol{k}}^* \hat{d}_\sigma^\dagger \hat{c}_{\boldsymbol{k},\sigma}) \\ &+ U\hat{n}_{d,\uparrow}\hat{n}_{d,\downarrow} + \varepsilon_d(\hat{n}_{d,\uparrow} + \hat{n}_{d,\downarrow}),\end{aligned} \tag{2}$$

where the impurity is described by the fermionic annihilation (creation) operators $\hat{d}_\sigma^{(\dagger)}$, the number operator $\hat{n}_{d,\sigma} = \hat{d}_\sigma^\dagger \hat{d}_\sigma$, the impurity one-particle energy level ε_d and the on-site repulsion U. The bath is described by the kinetic term $\hat{c}_{\boldsymbol{k},\sigma}^{(\dagger)}$ with one-particle energies $\varepsilon_{\boldsymbol{k}}$. The hybridization between the impurity and the bath is given by $V_{\boldsymbol{k}}$. The bath parameters jointly determine the frequency-dependent hybridization function:

$$\Delta(\nu) = \sum_{\boldsymbol{k}} \frac{|V_{\boldsymbol{k}}|^2}{i\nu - \varepsilon_{\boldsymbol{k}}}. \tag{3}$$

In this paper, we use the following hybridization function:

$$\Delta(\nu) = -\frac{iV^2}{D} \arctan\left(\frac{D}{\nu}\right), \tag{4}$$

which corresponds to a flat density of states of the bath electrons $\rho(\epsilon) = \theta(D - |\epsilon|)/(2D)$ with bandwidth D and $V_{\boldsymbol{k}} = V$. We will present results for $V = 2$, $D = 10$, and half filling, i.e., with $\varepsilon_d = -U/2$.

III. PARQUET EQUATIONS

The parquet equations are a set of exact relations between different classes of two-particle vertices and between the self-energy and the full two-particle vertex [4,5]. A good introduction to the formalism is provided in Ref. [6]. Here we only recall the equations, using the notations of Refs. [1,42–44]. Before we introduce the equations themselves, we first recapitulate some definitions to set the notations.

A. One-particle quantities

The one-particle Green's function in the Matsubara frequency space $G_\sigma(\nu)$ is defined as the Fourier transform of the (imaginary-time ordered) two-point correlation function:

$$G_\sigma(\nu) = -\int_0^\beta d\tau e^{i\nu\tau}\langle T_\tau \hat{c}_\sigma(\tau)\hat{c}_\sigma^\dagger(0)\rangle, \tag{5}$$

with τ denoting the imaginary time, $\beta \equiv 1/T$ the inverse temperature, and $\nu = (2n+1)\pi/\beta, n \in \mathbb{Z}$ denoting the (discrete) fermionic Matsubara frequencies. Through the Dyson equation, we further define the self-energy $\Sigma_\sigma(\nu)$,

$$G_\sigma(\nu) = \frac{1}{G_{0,\sigma}^{-1}(\nu) - \Sigma_\sigma(\nu)}, \tag{6}$$

where $G_{0,\sigma}(\nu)$ is the Green's function of the noninteracting system:

$$G_{0,\sigma}(\nu) = \frac{1}{i\nu + \frac{U}{2} - \Delta(\nu)}, \tag{7}$$

where we have set the following model-dependent parameters to values corresponding to half filling: For the Hubbard atom, $\Delta(\nu)$ vanishes, and the chemical potential is $\mu = U/2$. For the single-impurity Anderson model, we set the chemical potential to $\mu = 0$ and the one-particle energy level to $\epsilon_d = -U/2$.

B. Two-particle quantities

The two-particle Green's function in Matsubara frequencies is the Fourier transform of the (imaginary time ordered) four-point correlator:

$$G^{\nu_1\nu_2\nu_3}_{\sigma_1\ldots\sigma_4} = \int_0^\beta d\tau_1 \int_0^\beta d\tau_2 \int_0^\beta d\tau_3\, e^{i\nu_1\tau_1 + i\nu_2\tau_2 + i\nu_3\tau_3} \times \langle T_\tau \hat{c}_{\sigma_1}(\tau_1)\hat{c}^\dagger_{\sigma_2}(\tau_2)\hat{c}_{\sigma_3}(\tau_3)\hat{c}^\dagger_{\sigma_4}(0)\rangle. \tag{8}$$

In the above definition of the Fourier transform, the two-particle Green's function is dependent on three fermionic Matsubara frequencies ν_1, ν_2, ν_3. In the context of Bethe-Salpeter equations (defined below in Sec. III C), it is more convenient to parametrize two-particle quantities as a function of two fermionic frequencies ν, ν' and one bosonic Matsubara frequency $\omega = \frac{2n\pi}{\beta}, n \in \mathbb{Z}$. There are three important conventions of this parametrization: the particle-hole (ph) channel notation, where $\omega = \nu_1 + \nu_2$; the particle-particle (pp) channel notation, where $\omega = \nu_1 + \nu_3$; and the transversal particle-hole ($\overline{ph}$) channel notation, where $\omega = \nu_2 + \nu_3$. In the parquet approach, it is necessary to transform between these conventions using the so-called channel transformations outlined in Appendix A. The reason, as we will see in Sec. III D, is that the parquet equation mixes vertex functions that are represented in different frequency channel parametrizations.

In this paper, we use the SU(2) symmetry of the discussed models, which allows us, together with spin conservation, to reduce the number of spin components that need to be computed to the following: $G_{\sigma\sigma\sigma'\sigma'}$, which we will denote by $G_{\sigma\sigma'}$, and $G_{\sigma(-\sigma)(-\sigma)\sigma}$, which can be shown to be equal to $G_{\sigma\sigma} - G_{\sigma(-\sigma)}$. Furthermore, since $G_{\sigma\sigma'} = G_{(-\sigma)(-\sigma')}$, we only need to compute $G_{\uparrow\uparrow}$ and $G_{\uparrow\downarrow}$. We will proceed in a similar manner for the vertex F (see below). From here on, we will also drop the spin index from the one-particle objects G_0, G, and Σ, since $G_\uparrow = G_\downarrow$.

The full two-particle vertex F is the connected part of the two-particle Green's function with "amputated legs." In the particle-hole channel, it is related to the two-particle Green's function through

$$G^{\nu\nu'\omega}_{\sigma\sigma'} = G(\nu)G(\nu')\delta_{\omega 0} - G(\nu)G(\nu+\omega)\delta_{\nu\nu'}\delta_{\sigma\sigma'} - G(\nu)G(\nu+\omega)F^{\nu\nu'\omega}_{\sigma\sigma'}G(\nu')G(\nu'+\omega). \tag{9}$$

Apart from channels stemming from different frequency parametrizations (ph, pp, and $\overline{ph}$), it is convenient to introduce also linear combinations of spin components. The following spin combinations will be used for vertices in the ph frequency channel:

$$F_d = F_{\uparrow\uparrow} + F_{\uparrow\downarrow}, \qquad F_m = F_{\uparrow\uparrow} - F_{\uparrow\downarrow}, \tag{10}$$

which physically correspond to the density (d) and magnetic (m) spin components.

The same vertex F can be represented in the particle-particle channel frequency parametrization: F^{pp} (see Appendix A for details). In the pp channel, the convenient spin combinations are the following:

$$F_s = F^{pp}_{\uparrow\uparrow} - F^{pp}_{\uparrow\downarrow}, \quad F_t = F^{pp}_{\uparrow\uparrow}, \tag{11}$$

physically corresponding to the singlet (s) and triplet (t) spin components. In the following, we will predominantly use the spin-component notation, i.e., $d/m/s/t$, assuming that the d or m spin components are always in the ph frequency channel notation and the s or t spin components are always in the pp frequency channel notation.

As we will see later (in Sec. III C), in the above four spin combinations the Bethe-Salpeter equations decouple in the spin variable.

C. Bethe-Salpeter equations

The full vertex F contains all diagrams irrespective of their two-particle reducibility. The Bethe-Salpeter equations relate the full two-particle vertex to sets of two-particle irreducible diagrams. This is analogous to Dyson's equation (6), however, in the two-particle case the notion of irreducibility is not unique. Instead of one Dyson equation, we have independent BSEs in particle-hole and particle-particle channels. In the ph channel, we have the equations for density and magnetic components:

$$F_d^{\nu\nu'\omega} = \Gamma_d^{\nu\nu'\omega} - \frac{1}{\beta^2}\sum_{\nu_1\nu_2}\Gamma_d^{\nu\nu_1\omega}\chi^{\nu_1\nu_2\omega}_{0,ph}F_d^{\nu_2\nu'\omega}, \tag{12a}$$

$$F_m^{\nu\nu'\omega} = \Gamma_m^{\nu\nu'\omega} - \frac{1}{\beta^2}\sum_{\nu_1\nu_2}\Gamma_m^{\nu\nu_1\omega}\chi^{\nu_1\nu_2\omega}_{0,ph}F_m^{\nu_2\nu'\omega}; \tag{12b}$$

in the pp channel, we have the equations for the singlet and triplet components:

$$F_s^{\nu\nu'\omega} = \Gamma_s^{\nu\nu'\omega} + \frac{1}{\beta^2}\sum_{\nu_1\nu_2}F_s^{\nu\nu_1\omega}\chi^{\nu_1\nu_2\omega}_{0,pp}\Gamma_s^{\nu_2\nu'\omega}, \tag{12c}$$

$$F_t^{\nu\nu'\omega} = \Gamma_t^{\nu\nu'\omega} - \frac{1}{\beta^2}\sum_{\nu_1\nu_2}F_t^{\nu\nu_1\omega}\chi^{\nu_1\nu_2\omega}_{0,pp}\Gamma_t^{\nu_2\nu'\omega}. \tag{12d}$$

The above equations define four irreducible vertices Γ_r, $r = d/m/s/t$ that are irreducible in either the ph channel ($r = d/m$) or pp channel ($r = s/t$). We define vertices reducible in these channels simply as

$$\Phi_{d/m}^{\nu\nu'\omega} = F_{d/m}^{\nu\nu'\omega} - \Gamma_{d/m}^{\nu\nu'\omega}, \tag{13a}$$

$$\Phi_{s/t}^{\nu\nu'\omega} = F_{s/t}^{\nu\nu'\omega} - \Gamma_{s/t}^{\nu\nu'\omega}. \tag{13b}$$

The χ_0's are products of two one-particle Green's functions and are defined as follows:

$$\chi_{0,ph}^{\nu\nu'\omega} = -\beta G(\nu)G(\nu+\omega)\delta_{\nu\nu'}, \tag{14a}$$

$$\chi_{0,pp}^{\nu\nu'\omega} = -\frac{\beta}{2} G(\nu)G(-\nu-\omega)\delta_{\nu\nu'}. \tag{14b}$$

The pair propagators (also called bare generalized susceptibilities) χ_0's are diagonal in ν, ν', which means that the sum in Eqs. (12) runs over only one fermionic Matsubara frequency index. For convenience of actual numerical evaluations, we, however, keep the double fermionic frequency dependence in χ_0's.

Due to convenient parametrization of the frequency dependence of the vertices, i.e., the ph channel for d/m and pp channel for s/t, the BSEs (12) are diagonal both in the bosonic frequency ω and in the spin components $d/m/s/t$.

D. Parquet equation

Through Eqs. (12) and (13), we defined reducible vertices $\Phi_{d/m}$ and $\Phi_{s/t}$ in ph and pp channels, respectively. These vertices correspond to different physical processes that happen in the ph and pp scattering channels and are generated by the BSEs (12). The parquet equation mixes these processes, allowing for balance between contributions generated by all of the BSEs (12). In a simplified way, the parquet equation can be represented as the following sum of terms:

$$F = \Lambda + \Phi_{ph} + \Phi_{\overline{ph}} + \Phi_{pp}, \tag{15}$$

where Φ_{ph} denotes contributions coming from Φ_d or Φ_m, $\Phi_{\overline{ph}}$ contributions coming from $\Phi_{d/m}$, but in the $\overline{ph}$ frequency parametrization, and Φ_{pp} contributions from Φ_s or Φ_t (more details can be found in Appendix A or in Ref. [6]). The first summand, Λ, contains so-called fully two-particle irreducible diagrams, i.e., contributions which cannot be generated by the two-particle BSEs.

Since in the BSEs the reducible vertices Φ_r are in different frequency channel parametrizations, to sum the contributions we have to transform them into a common parametrization. The explicit form of the parquet equation for F_d is then the following [42]:

$$\begin{aligned} F_d^{\nu\nu'\omega} = {} & \Lambda_d^{\nu\nu'\omega} + \Phi_d^{\nu\nu'\omega} - \tfrac{1}{2}\Phi_d^{\nu(\nu+\omega)(\nu'-\nu)} \\ & - \tfrac{3}{2}\Phi_m^{\nu(\nu+\omega)(\nu'-\nu)} + \tfrac{1}{2}\Phi_s^{\nu\nu'(-\omega-\nu-\nu')} \\ & + \tfrac{3}{2}\Phi_t^{\nu\nu'(-\omega-\nu-\nu')}, \end{aligned} \tag{16}$$

where Λ_d is the density component of the fully irreducible vertex. Λ_d cannot be obtained from the BSEs and has to be provided from outside the parquet scheme. In the examples presented in Sec. VI, we either use the exact expression (it is known for the Hubbard atom) or we use a weak coupling approximation for it. Equation (15) can be used to generate equations analogous to Eq. (16) for F_m, F_s, and F_t. We provide them explicitly in Appendix A.

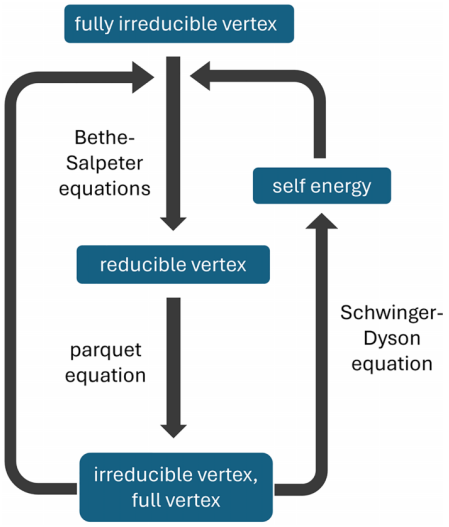


FIG. 1. Iterative parquet scheme.

E. Schwinger-Dyson equation

The last equation that belongs to the set of parquet equations is the SDE that relates the two-particle vertex F to the self-energy

$$\Sigma(\nu) = \frac{Un}{2} - \frac{U}{\beta^2}\sum_{\nu'\omega} F_{\uparrow\downarrow}^{\nu\nu'\omega} G(\nu')G(\nu'+\omega)G(\nu+\omega), \tag{17}$$

where n is the average particle density and $F_{\uparrow\downarrow}^{\nu\nu'\omega} = \frac{1}{2}(F_d^{\nu\nu'\omega} - F_m^{\nu\nu'\omega})$ [follows from Eq. (10)].

F. Iterative parquet scheme

Assuming we know the fully irreducible vertex Λ (or have a good approximation for it), the parquet equation (15) together with the BSEs (12) and the SDE (17), as well as the Dyson equation (6), applied iteratively, will generate all the vertices for the model given by the Hamiltonian and G_0: $\{F_{d/m}, F_{s/t}, \Gamma_{d/m}, \Gamma_{s/t}, \Phi_{d/m}, \Phi_{s/t}\}$ as well as the self-energy Σ and the Green's function G. The iterations are repeated until the difference between consecutive values falls below a given tolerance.

The input quantity that does not change in the iterations, namely, the fully irreducible vertex Λ, is known exactly (even analytically [40]) for the Hubbard atom and we use this exact expression. For the SIAM, we use the so-called parquet approximation which sets Λ equal to the bare interaction U (it is the lowest order diagram in Λ, the next order appearing in the diagrammatic expansion of Λ is U^4). Explicitly written out in spin components, the parquet approximation reads

$$\Lambda_d = U, \quad \Lambda_m = -U, \quad \Lambda_s = 2U, \quad \Lambda_t = 0. \tag{18}$$

In Fig. 1, we show how the actual iterated loop is implemented in this paper. In the first cycle of the iteration, we

either set $\Gamma_r = \Lambda_r, F_d = U, F_m = -U, F_s = 2U, F_t = \Lambda_t$ or use previous results for smaller values of βU. After initialization, the BSEs (12) are evaluated and the reducible vertices Φ_r are obtained. Then, from the parquet equation (16) (for all channels), the new F is computed and the irreducible vertices Γ_r are updated from (13). From the new F, the self-energy Σ can be obtained from SDE (17) and the one-particle Green's function from (6). The self-energy update does not have to happen in each iteration—depending on the value of U, it might be faster to update it every five or ten iterations. With updated F, Γ_r, and G, the cycle consisting of the nine equations is repeated until convergence is reached. In this paper, to keep things simple, we limit convergence acceleration to a linear mixing update to the reducible vertex, $\Phi_{n+1} = \alpha\Phi'_n + (1-\alpha)\Phi_n$ with mixing parameter α, where Φ_n is the reducible vertex in iteration n and Φ'_n is the result of applying a single cycle to Φ_n.

We conclude this section with some comments on convergence issues. First, the iterative parquet scheme is not guaranteed to converge. Second, there are cases where the iterative scheme leads to a false solution. This is the case, e.g., for the Hubbard atom for large βU beyond the first divergence of the irreducible vertex [45]. In this paper, we do not address such cases and focus on examples and parameter regimes where the parquet scheme should converge to the physical solution. Nevertheless, convergence can also be influenced by other factors. The first numerical solution for the Hubbard model on a 4×4 cluster [7] showed the difficulty of achieving convergence, particularly when the crossing symmetry (see Appendix A) was not obeyed. An important factor in improving stability of the iterative scheme turned out to be the inclusion of vertex asymptotics, i.e., a prediction for values of the vertex that fall beyond the frequency range used, either by introducing so-called kernels [8,10,44,46] or by removing the asymptotic parts of vertices altogether in the single-boson exchange reformulation [3,12]. In this paper, we do not use asymptotics and do not suffer from convergence problems mainly because we are able to use very large grids. In the future, it might be important to include asymptotic, as in Refs. [8,10,44,46] or [3,12], to improve stability but also to avoid some technical issues that are addressed in Appendix F.

IV. QUANTICS TENSOR TRAINS

Numerical solution of the iterative parquet scheme introduced in the previous section suffers from the curse of dimensionality: the vertex functions have multiple frequency (and in general also momentum) arguments. Discretizing these multivariate functions on naive grids requires a number of grid points that grows exponentially with the number of function arguments, which therefore becomes very expensive already for a moderate number of grid points in each argument. The solution to this problem that we propose is (i) to represent each variable through a set of binary numbers (hence quantics) corresponding to different length and energy scales and (ii) to factorize the dependence on each argument at each length and energy scale into a tensor train (TT), also known as a matrix product state (MPS). If the problem has some kind of scale separation, the resulting QTT is expected to have a small maximum bond dimension. Since this is the case in many physical problems, such an approach is potentially very powerful. It has already been shown to reduce computational costs significantly in several applications with high-dimensional functions [20,21,31,39,47].

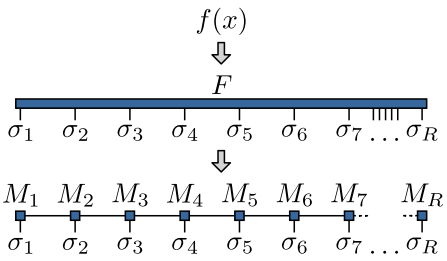


FIG. 2. Quantics representation and quantics tensor train of a univariate function.

In this section, we introduce the definition of the QTT representation, then present a method for efficient compression of multivariate functions into a QTT, namely, the TCI. Finally, we also introduce MPOs that are needed for computations with the QTTs. These methods are valid for any multivariate function and not specific to two-particle vertices.

In the remainder of the paper, the so-called grid parameter R will be of central importance, where the three-dimensional Matsubara frequency grid will consist of 2^{3R} grid points. Hence, this parameter governs the (exponential) number of points of the discretized grid and exponentially different length scales in the system and thus determines the length of the resulting QTT. This will become more clear in the following sections.

A. Quantics tensor train representation

Before discussing the three-dimensional Matsubara frequency case, let us introduce the QTT formalism for the one-dimensional case for educational purposes. In the quantics representation, a discrete function $f(m)$ with $m \in \{0, \ldots, M-1\}$ on a one-dimensional grid with $M = 2^R$ grid points is instead seen as a $2 \times 2 \times \ldots \times 2$ (R times) tensor $F_{\sigma_1,\ldots,\sigma_R}$ (see Fig. 2), where each tensor index $\sigma_1, \ldots, \sigma_R$ corresponds to a bit in a binary representation of m:

$$m = (\sigma_1\sigma_2 \ldots \sigma_R)_2 = \sum_{\ell=1}^{R} 2^{R-\ell}\sigma_\ell, \quad \sigma_\ell \in \{0, 1\}, \tag{19}$$

with the discussed grid parameter R. Now, each bit corresponds to a distinct length scale of the system. The first bit σ_1 represents the coarsest length scale which splits the system in halves, while the last bit σ_R reflects the finest length scale. Hence, for continuous variables m defined on a specific interval the grid parameter, R determines how exponentially dense the grid gets, while for discrete variables like Matsubara frequencies, R determines how exponentially large the grid gets. In both cases, R specifies the (exponential) number of grid points and a linear increase in R corresponds to an exponential increase in the number of grid points.

This representation can be generalized to functions of $N > 1$ variables by applying the binary representation to each variable separately. For instance, a function $f(x, y, z)$ of three variables is represented as a tensor depending on $3R$ binary

indices

$$F_{x_1,y_1,z_1,x_2,y_2,z_2,\ldots,x_R,y_R,z_R} = f((x_1 \ldots x_R)_2, (y_1 \ldots y_R)_2, (z_1 \ldots z_R)_2), \quad (20)$$

and the $L = 3R$ indices are relabeled $\sigma_1 = x_1, \sigma_2 = y_1, \sigma_3 = z_1, \sigma_4 = x_2, \ldots, \sigma_L = z_R$. Note that indices belonging to different variables are interleaved, which leads to an index ordering where all indices describing large length scales are grouped to the left, and all indices describing small length scales are grouped to the right. This tensor can then be factorized into a TT, also known as a MPS, of the form

$$F_{\sigma_1,\ldots,\sigma_L} \approx \prod_{\ell=1}^{L} M_\ell^{\sigma_\ell} = [M_1]^{\sigma_1}_{1\alpha_1} [M_2]^{\sigma_2}_{\alpha_1\alpha_2} \cdots [M_L]^{\sigma_L}_{\alpha_{L-1}1}, \quad (21)$$

with implied summation over repeated indices. Each M_ℓ is a three-leg tensor with local binary index σ_ℓ and virtual indices $\alpha_{\ell-1}, \alpha_\ell$, and we define the bond dimension D_ℓ as the number of values that index α_ℓ is summed over. Hence, M_ℓ is a $D_{\ell-1} \times 2 \times D_\ell$ tensor. Generally, the bond dimensions D_ℓ are truncated either at a fixed maximum bond dimension $D_{\max}$ or such that the factorization satisfies a specified error tolerance ϵ. This truncated TT factorization can be performed using singular value decomposition (SVD) or using the TCI algorithm (see the next Sec. IV B).

Overcoming the curse of dimensionality now depends on the maximum bond dimension $D_{\max} = \max_\ell(D_\ell)$, as the tensors M have $\mathcal{O}(D_{\max}^2 R)$ elements. The bond dimension required to reach a specified error tolerance ϵ is strongly dependent on the structure of F. If F is not compressible, e.g., a random tensor, bond dimensions will grow exponentially with L as $D_{\max} \approx 2^{L/2}$ and the factorization will thus not result in an efficiency gain. Fortunately, many functions in physics contain low-rank structures when factorized in their length scales. The interleaved representation groups bits corresponding to the same length scale, resulting in a highly compressed representation with small $D_{\max}$ [37,39].

B. Tensor cross interpolation

The TCI-based factorization is performed by sampling a subset of the elements of the full tensor F. To be more specific, the TCI algorithm takes as input a tensor F in the form of a function returning the value $F_{\sigma_1,\ldots,\sigma_L}$ at any given index $(\sigma_1, \ldots, \sigma_L)$ [35,37,39]. The algorithm explores its structure by sampling in a deterministic way and constructs a low-rank approximation $\tilde{F}$ in the form of an MPS. The algorithm increases the number of samples and the bond dimensions of the MPS adaptively until the estimated error ϵ in the maximum norm,

$$\epsilon = \frac{\|F - \tilde{F}\|_\infty}{\|F\|_\infty}, \quad (22)$$

is below a specified tolerance. Here, $\|\cdot\|_\infty$ denotes the maximum norm.

TCI is more efficient than the SVD-based factorization, especially when the full tensor does not fit into the available memory [35,37,39]. SVD-based factorization requires reading all elements of the tensor, leading to an exponential growth of the computation time in R. In contrast, if the target tensor or function is low rank, the computation time of the TCI-based factorization is linear in R, leading to an exponential speedup over the SVD [39]. We refer the reader to Refs. [37,39] for more technical details, e.g., for information on how the sampling points in the TCI algorithm are chosen. In the following computations, TCI will only be used for compressing the initial input vertices and functions. On a more technical note, the `crossinterpolate2` algorithm in the `TensorCrossInterpolation.jl` library was used for compressing the objects. More details on this specific algorithm can be found in Sec. 8.3.1 in Ref. [37].

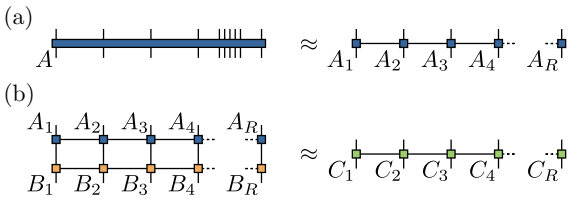


FIG. 3. (a) Decomposition of a tensor in MPO form. (b) Contraction of two MPOs A and B.

C. Matrix product operators

We use MPOs to perform operations on QTTs. As illustrated in Fig. 3(a), an MPO of length L has two physical legs on each tensor. The MPO can be regarded as the factorization of a full tensor of order $2L$ or as a linear operator acting on an MPS of length L.

As we will see in later sections, many operations in QTT can be implemented as the contraction of two MPOs, illustrated in Fig. 3(b). The exact contraction will result in an MPS of large bond dimension $D_A D_B$, where D_A and D_B are the bond dimensions of the two input MPOs, respectively. Thus, the bond dimension of the resulting MPO must be truncated to some $D_{\max}$. The computational cost of a naive SVD-based contraction followed by truncation scales $\mathcal{O}(D_A^3 D_B^3)$.

In the present paper, we will deal with two distinct cases: (a) $D_A = \mathcal{O}(1) \ll D_B$ (channel transformation) and (b) $D_A = D_B = D_{\max}$ (Bethe-Salpeter equation). For the former case (a), the naive approach is efficient enough, with scaling $\mathcal{O}(D_B^3 L)$.

However, a more efficient scheme is necessary for case (b). We use two algorithms: the fit algorithm fits new MPOs to the MPO-MPO contraction [48], and the zip-up algorithm combines contraction of core tensors with truncation of the bond dimensions [49]. We typically combine these using the zip-up algorithm to generate an initial guess for the fit algorithm. If the resulting MPO is truncated to bond dimension $D_{\max} = D_A = D_B$, the computational cost of both algorithms scales as $\mathcal{O}(D_{\max}^4 L)$. We want to emphasize that in the current implementation the combination of the two algorithms is still SVD- and not TCI-based. Thus, TCI is only used in the compression of the input functions and not in the MPO-MPO contractions. However, in future work we plan on using TCI for MPO-MPO contractions as well, since this is expected to be computationally more efficient.

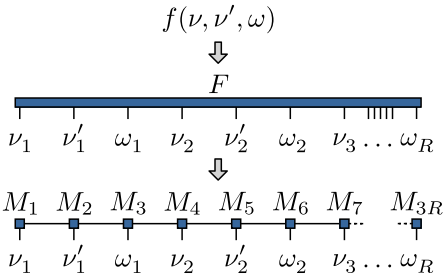


FIG. 4. QTT representation for full vertex function.

V. PARQUET EQUATIONS IN QTT FORMAT

To evaluate the full set of parquet equations completely within the QTT representation, we need to (i) represent the vertex functions in the QTT form and (ii) decompose Eqs. (12)–(17) into operations on QTTs. The latter can be implemented in a straightforward way by employing fundamental operations described in the previous section, namely, MPO-MPO contractions. In the following subsections, we describe the quantics representation of vertex functions, check their compressibility to QTTs, and discuss the implementation of each step in solving the parquet equations. Two operations are particularly important:

(1) Affine transformations that are represented by an MPO with maximum bond dimension of $\mathcal{O}(1)$, needed in the parquet equation Eq. (16) for frequency channel transformations of vertex functions (Sec. V B).

(2) Elementwise and matrix multiplications of two QTT vertex objects for solving the BSEs (12) and SDE (17). In this case, auxiliary MPOs are introduced substituting the MPSs and then MPO-MPO contractions are applied (Sec. V D).

A. Quantics representation and compression of two-particle vertex functions

In this paper, all functions represented in QTT format are functions of bosonic and fermionic frequencies, which are parameterized as $\nu = (2m - 2^R + 1)\pi/\beta$ and $\omega = (2m - 2^R)\pi/\beta$, respectively. The discrete index $m \in \{0, \ldots, 2^R - 1\}$ is then decomposed into quantics bits as in Eq. (19), and bits corresponding to different variables are then interleaved as illustrated in Fig. 4.

The first step in using the QTT framework for solving the parquet equations is to investigate the compressibility of vertex functions in the above representation. We use the full vertex in the density channel (F_d) of the Hubbard atom for numerical demonstration. The fermionic frequency dependence of F_d at $\omega = 0$ is shown for various temperatures in Fig. 5. Note that in the Hubbard atom the results are not separately dependent on temperature and U but only on their ratio βU, since there are no other energy scales [40]. Before we focus on the scaling of bond dimension with temperature, let us first look at the bond dimension along the QTT at $\beta U = 1$ for different grid sizes and tolerances set in TCI as shown in Fig. 6. Moving inward from the first and last bonds, the bond dimension grows exponentially as $D_\ell = \min(2^\ell, 2^{\mathcal{L}-\ell})$, which is the maximum bond dimension of an uncompressed factorization and represents maximum entanglement between

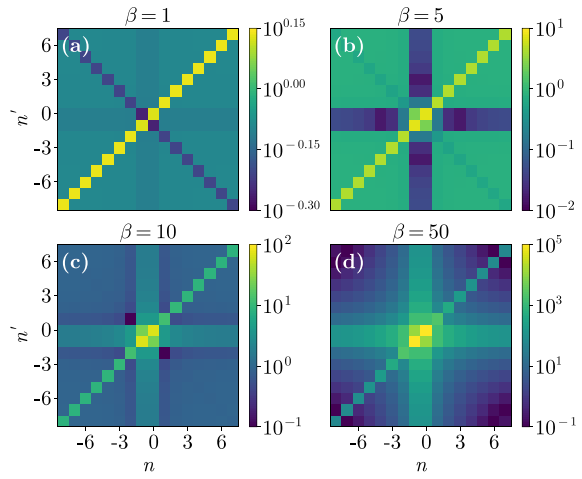


FIG. 5. Absolute value of the full vertex in the density channel F_d at $\omega = 0, U = 1$ for the 16 innermost fermionic Matsubara frequencies $\nu^{(\prime)} = (2n^{(\prime)} + 1)\pi/\beta$ for $\beta = 1, 5, 10, 50$.

these exponentially different length scales. In between, the bond dimension then saturates at a maximum bond dimension $D_{\max}$, therefore indicating that the vertex structures are indeed compressible. The maximum bond dimension $D_{\max}$ is between 80 and 400 and increases with decreasing tolerance ϵ. Importantly, $D_{\max}$ is nearly independent of the grid parameter R, an exponential increase in the number of grid points $[\mathcal{O}(2^R)]$ can be achieved for linear cost $[\mathcal{O}(R)]$ in runtime and memory. These findings are consistent with earlier results on the compressibility of vertices in Ref. [31].

For large temperatures, such as $\beta U = 1$ in the above example, the dominating structures are the diagonal and antidiagonal part of the vertex. For small temperatures, i.e., large βU, the antidiagonal vanishes and an additional cross

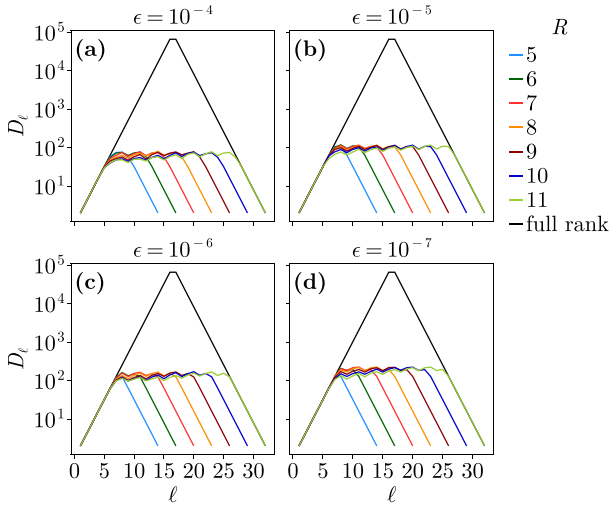


FIG. 6. Bond dimension D_ℓ at bond ℓ for different tolerances set in the TCI construction of the QTT of F_d in the interleaved representation for $\beta = U = 1$. The different values of R correspond to different grid sizes (2^{3R} grid points). The black line indicates the exponentially growing bond dimension of the full rank QTT without any truncation for $R = 11$.

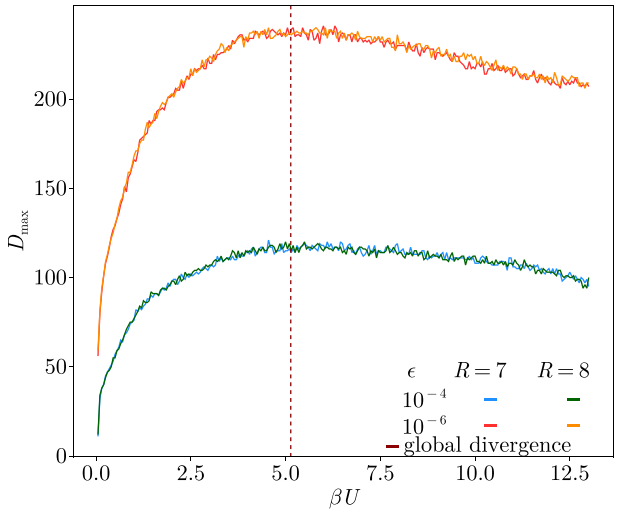


FIG. 7. Maximum bond dimension $D_{\max}$ of the QTT of F_d for different grid sizes and various tolerances set in the TCI construction as a function of βU.

structure appears for one of the Matsubara frequencies equal to $\pm\pi/\beta$. The source and physical meaning of different structures in two-particle vertices have been discussed in Ref. [1]. For large βU, the vertex is large even in some places where both Matsubara frequency arguments are large, as can be seen in Fig. 5(d). All in all, the structure of F_d seems to be much simpler at large temperatures than at small ones, which leads us to expect an increase in bond dimension in QTT representation with increasing βU.

In Fig. 7, we show $D_{\max}$ of the QTT representation of F_d for various values of βU, different tolerances set in TCI, and various grid sizes. As expected from earlier considerations, the maximum bond dimension is quite low at high temperatures and steeply grows with βU until $\beta U \approx 5.1$, where it reaches a maximum. Thereafter, the bond dimension decreases again, which does not conform with our expectations. The maximum can be related to the first global divergence of the irreducible vertex Γ_d [40,50–54], which occurs at $\beta U = 5.13715$ and is indicated by a dashed line in the plot. Although the irreducible vertex Γ_d diverges at this value of βU, there is no phase transition connected with the divergence and F_d remains finite. The presence of a maximum in $D_{\max}$ precisely at the first global divergence of Γ_d is very interesting from two different perspectives. First, we see that $D_{\max}$ stops growing with βU and, thus, calculations for temperatures ranging from very low to very high are manageable within the QTT framework in this case. Second, although the full vertex F_d does not contain singularities, we see fingerprints of this first global divergence of Γ_d in the amount of length scale entanglement in the system represented by the maximum bond dimension. A deeper investigation of this behavior is left for future work.

The vertex functions in other channels show a similar scaling behavior as F_d (not shown here), and hence it can be concluded that vertex functions of the Hubbard atom are nicely compressible with a maximum bond dimension of around 100. Together with the logarithmic scaling in grid size, this indicates that QTCI can indeed overcome memory and computational bottlenecks when dealing with two-particle vertices.

The Hubbard atom is quite an extreme limit of the Hubbard model and one could think that the high compressibility is related to structures present in the vertices that are limited to this atomic limit. However, this is not the case: Prominent structures in the frequency dependence of vertices are well-known to be present in the SIAM (see Sec. VI below for an example), as well as in the local approximation of the Hubbard model (in dynamical mean-field theory) [42,52]. These structures arise specifically from certain types of diagrams [42,52] and also from insertions from one scattering channel to another through the parquet equation (15). For models whose vertices depend on momentum and/or orbital indices, the structures may become more complicated. Nevertheless, their origin is well understood and we expect the vertices to generically have structures for a large range of parameters and models. As long as these structures show some kind of scale separation, they will be QTT compressible, hopefully with still manageable bond dimensions.

B. Channel transformations

Each two-particle reducible vertex Φ is parameterized as a function of two fermionic frequencies ν, ν' and one bosonic frequency ω. Before performing the sum in the parquet equation (16), it is necessary to perform transformations such as $\Phi_d^{\nu\nu'\omega} \to \Phi_d^{\nu(\nu+\omega)(\nu'-\nu)}$ to translate between the frequency parametrizations corresponding to different channels [6], as discussed in Sec. III D and Appendix A. In QTT format, transforming the function arguments is a nontrivial task, since each argument is split into bits across different tensor indices. Affine transformations such as the channel transformations needed here can be expressed as MPOs with small bond dimensions, as described in Appendix B. The QTT implementation of channel transformations is introduced explicitly in Appendix C. As illustrated in Fig. 8(a), these MPOs are then applied to vertex QTTs, followed by truncation of the QTT to the specified bond dimension.

In this process, there are two distinct sources of error: the finite frequency box and the QTT truncation. As an example, Fig. 9(a) shows the absolute normalized error ($||\Delta F_{pp}|| := |F_{pp,\text{trafo}} - F_{pp,\text{exact}}|/||F_{pp,\text{exact}}||_\infty$) of a transformation of the full vertex F in the ph channel parametrization to the pp channel parametrization (denoted as F_{pp}) in the ν, ν'-plane. In two triangular regions, the upper right and lower left corner, errors are large due to the finite size of the frequency box, and are not caused by the QTT compression. These points correspond to grid points outside of the original frequency box that were transformed into the box by the ph to pp transformation. The missing data there can be either replaced by zeros (which corresponds to open boundary conditions of the affine transformation; see Appendix B) or by values extrapolated from another part of the frequency box (e.g., by using periodic boundary conditions for the affine transformation, as in Ref. 55, footnote 14]). We checked that for the examples shown in this paper, the average difference in error between the two options is small. We used periodic boundary conditions for all results presented in the paper. For the half-filled Hubbard atom, this leads to better representation of missing values on

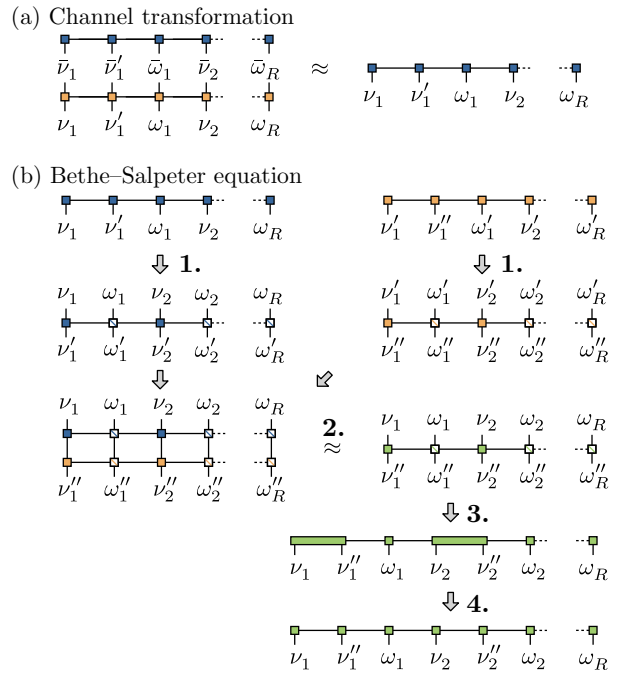


FIG. 8. QTT implementation of the Bethe-Salpeter equations as tensor networks. (a) Channel transformation of a vertex in QTT form (blue) using an affine transform MPO (orange). This is described in more detail in Appendixes B and C. (b) The Bethe-Salpeter equations are evaluated from QTT vertices and are implemented using multiple MPO-MPO contractions (see text). The contraction itself is done in four steps: (1) Both QTTs to be contracted are converted to MPOs. (2) The MPOs are contracted to a single MPO. (3) The duplicate ω''_ℓ legs are removed. (4) The tensors with ν_ℓ and ν'' legs are factorized between their local legs, reaching the original QTT form.

the diagonal due to high symmetry of the full vertex in this case. In general, the choice of boundary conditions can be adapted to the problem at hand. Since the error can easily be reduced and shifted to higher Matsubara frequencies by increasing the size of the frequency box exponentially through increasing R, we do not expect the choice of boundary conditions to have significant effect on the error.

In the remaining diagonal region in between the two yellow corners in Fig. 9(a), the error is entirely due to the tensor train approximation. At a bond dimension of $D_{\max} = 100$, the error is smaller than the error tolerance of $\epsilon = 10^{-5}$ everywhere, meaning that the crossing symmetry is also fulfilled up to this error.

Since this operation consists of a single MPO-MPS contraction, it is expected to scale as $\mathcal{O}(D^3_{\max}L) = \mathcal{O}(D^3_{\max}R)$ provided that the bond dimension is independent of R. We verify this explicitly in Figs. 9(b) and 9(c). Compared to increasing resolution, decreasing error tolerances and thus increasing $D_{\max}$ is more expensive.

C. Parquet equation

The parquet equation (15) with its frequency shifts as in (16) can be solved entirely in QTT by first converting all the vertex functions Λ, Φ to the required channel as described in

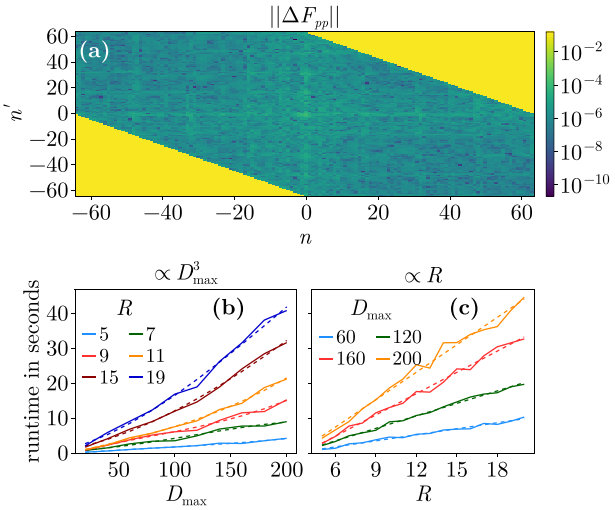


FIG. 9. (a) Absolute normalized error $||\Delta F_{pp}|| := |F_{pp,\text{trafo}} - F_{pp,\text{exact}}|/||F_{pp,\text{exact}}||_\infty$ of the ph to pp channel transformation of F at $\omega = 0$ in the fermionic Matsubara frequency plane for a $D_{\max} = 100$ and $R = 7$, $\beta = U = 1$, $\epsilon = 10^{-8}$. (b), (c) Runtime of the ph to pp channel transformation of F for various maximum bond dimensions and grid size parameters R with $\beta = U = 1$. The dashed lines indicate (b) the cubic runtime increase with $D_{\max}$ and (c) the linear increase by increasing R, which corresponds to an exponential increase in the number of grid points.

the previous section, then performing their summation and subtraction as shown in Ref. [37, Sec. 4.7]. The resultant QTTs for F [and subsequently Γ obtained from (13)] are then compressed to a maximum bond dimension $D_{\max}$ in $\mathcal{O}(D^3_{\max}R)$ computation time. Hence, the parquet equation has the same $\mathcal{O}(D^3_{\max}R)$ computational cost as channel transformations. Further investigation of error and runtime scaling can be found in Appendix D.

D. Bethe-Salpeter equation

The costliest part of the iterative parquet scheme are the Bethe-Salpeter equations (12), where two infinite Matsubara sums have to be performed. These can be implemented as a sequence of matrix multiplications as $(\Gamma\chi_0)F$, where χ_0 is treated as a vertex object with two fermionic frequency axes and one bosonic frequency axis. At each multiplication step, we have to compute the product of two vertex functions A and B as

$$C^{\nu\nu''\omega} = \sum_{\nu'} A^{\nu\nu'\omega} B^{\nu'\nu''\omega}. \tag{23}$$

To express this summation as matrix multiplication, we introduce dummy indices ω' and ω'', such that

$$C^{\nu\nu''\omega} = \sum_{\nu'\omega'} \tilde{A}^{\nu\omega}_{\nu'\omega'} \tilde{B}^{\nu'\omega'}_{\nu''\omega''}\big|_{\omega=\omega''}, \tag{24a}$$

$$\tilde{A}^{\nu\omega}_{\nu'\omega'} := A^{\nu\nu'\omega}\delta_{\omega,\omega'}, \tag{24b}$$

$$\tilde{B}^{\nu'\omega'}_{\nu''\omega''} := B^{\nu'\nu''\omega'}\delta_{\omega',\omega''}, \tag{24c}$$

where $|_{\omega=\omega''}$ denotes the restriction of the result to $\omega = \omega''$. Note that Eq. (24a) has the structure of a matrix multiplication in the combined index (ν', ω'). Thus, this equation can

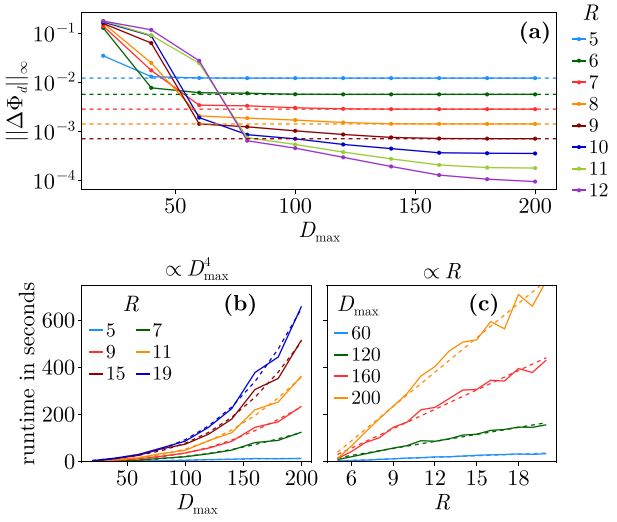


FIG. 10. (a) Dependence on $D_{\max}$ of the maximum absolute normalized error $||\Delta\Phi_d||_\infty := ||\Phi_{d,\text{BSE}} - \Phi_{d,\text{exact}}||_\infty / ||\Phi_{d,\text{exact}}||_\infty$ of the BSE with QTCI for the reducible vertex Φ_d compared to the exact values for various grid sizes, with $U = \beta = 1$ and $\epsilon = 10^{-10}$. The dashed lines indicate the results from the dense grid calculation without QTCI. (b), (c) Runtime of a single evaluation of the BSE for Φ_d for various maximum bond dimensions and grid size parameters R with $\beta = U = 1$. The dashed lines indicate (b) the quartic runtime increase with $D_{\max}$ and (c) the linear increase by increasing R, which corresponds to an exponential increase in the number of grid points.

be evaluated in QTT format through standard MPO-MPO contraction, as illustrated in Fig. 8(b). For numerical computations, and thus also for the QTT approach, the infinite Matsubara sums have to be truncated. The number of Matsubara frequencies taken into account is governed by the chosen Matsubara grid, containing 2^R points in each direction. Introducing dummy indices has a runtime and memory cost of $\mathcal{O}(D^2_{\max}R)$, which is much smaller than the cost of other steps in the algorithm.

After each MPO-MPO contraction, the bond dimensions are truncated to $D_{\max}$. If all MPOs are truncated to $D_{\max}$, the computational cost of each contraction is expected to scale as $\mathcal{O}(D^4_{\max}R)$ as described in Sec. IV C, which is more expensive than the channel transformation for large $D_{\max}$.

We now conduct numerical tests to verify the accuracy of the operation and the scaling of the computational cost. Figure 10(a) shows the dependence on $D_{\max}$ of the maximum absolute normalized error ($||\Delta\Phi_d||_\infty := ||\Phi_{d,\text{BSE}} - \Phi_{d,\text{exact}}||_\infty / ||\Phi_{d,\text{exact}}||_\infty$) [cf. Eq. (22)] of the QTT implementation of the BSE in the density channel for various grid sizes. The dashed lines indicate the results applying these matrix multiplications for the full numerical data and without the compressed QTTs. The error of these dense grid calculations is due to the finite size of the grid and, thus, caused by the truncated Matsubara sum. The results can be improved by increasing the grid size.

For the dense grid calculations, the improved results come at high cost since increasing the grid parameter R leads to an exponential increase in memory and computational cost. By contrast, when using QTTs the memory and computational cost only increase linearly with R, which can be observed in Fig. 10(c), where the linear dependence of the runtime of the BSE on R is shown for different maximum bond dimensions.

Moreover, it can be observed that the QTT BSE errors converge to the box results. Interestingly, for larger grid sizes, slightly larger $D_{\max}$ are needed to reach the same error level as in the case of smaller boxes. However, a maximum normalized error $<10^{-3}$ can be easily reached using QTTs for a still reasonable bond dimension of 200. Without the use of QTTs, this would correspond to calculations with objects of $8 \times 2^{3\times 12} \simeq 5.5 \times 10^{11}$ bytes, for which multiple nodes on a cluster would need to be occupied. With the current SVD-based QTT matrix multiplication implementation the BSE operation only takes about 400 seconds on a single 512 GB node (equipped with two AMD EPYC 7713 processors) without parallelization. The bottleneck of performing the BSE is, as expected, the quartic dependence on $D_{\max}$, which can be observed in Fig. 10(b). Overall we numerically verified $\mathcal{O}(RD^4_{\max})$ computational cost for the BSE using QTTs. Since the error of the result of the BSE is bound by the grid size, QTTs provide an efficient way to overcome this bottleneck.

E. Schwinger-Dyson equation

Similarly to the BSE evaluation, two infinite Matsubara sums have to be performed in the SDE in Eq. (17). Hence, qualitatively similar scaling with $D_{\max}$ and R to the BSE case are expected. This is indeed the case. The results are presented in Fig. 16 in Appendix E.

F. Initialization and update strategy

Let us briefly discuss how the iterative parquet calculations are performed in practice. We start by compressing the initial inputs G_0 and the exact fully irreducible vertices Λ_r with TCI, where in the Hubbard atom case the exact Λ_r is taken as input and in the SIAM case we make use of the parquet approximation ($\Lambda_d = U$, $\Lambda_m = -U$, $\Lambda_s = 2U$, $\Lambda_t = 0$). We then set $\Gamma_r = \Lambda_r$, $F_d = U$, $F_m = -U$, $F_s = 2U$, $F_t = \Lambda_t$, $G = G_0$, $\Phi_r = 0$, where the QTT representation for the input full vertices $F_r = U$ can easily be obtained with or without TCI since this can be represented by a QTT of bond dimension one. Then, the right sides of the four BSEs are computed using (SVD-based) the MPO-MPO contractions that were discussed in Sec. IV C. The output of the BSEs is then linearly mixed with the QTTs of the input Φ_r, resulting in updated QTT approximations of the reducible vertices Φ_r. These updated Φ_r QTTs are then used as input in the parquet equations; as output, the QTT representations of Γ_r and F_r are updated. As a last step, the QTT representation of the self-energy is updated by solving the SDE in QTT format. The updated QTT vertices and self-energy are then used as new input in the next iteration step of the iterative parquet scheme.

VI. RESULTS OF SELF-CONSISTENT CALCULATIONS

With all the components from the previous section in place, the parquet equations can now be iteratively solved within the developed two-particle QTT framework for our two test cases: the Hubbard model in the atomic limit and the single-impurity Anderson model.

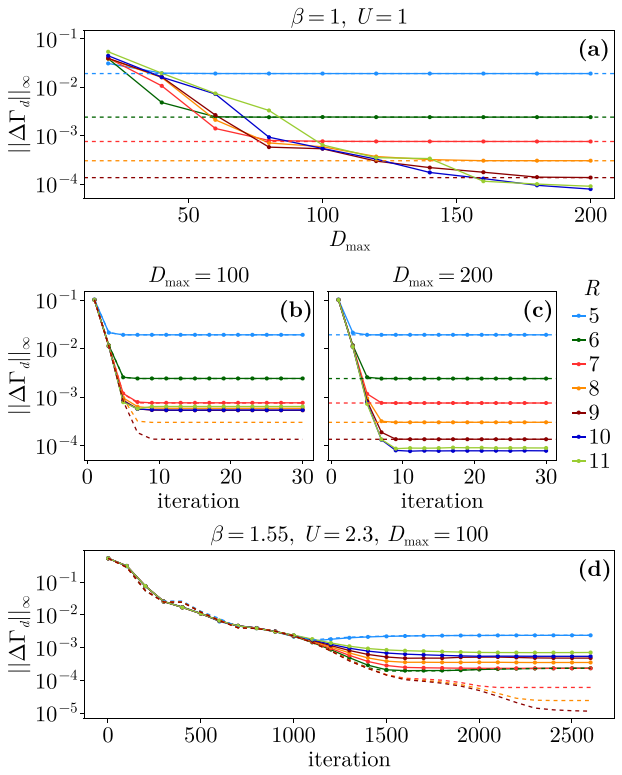


FIG. 11. (a) Maximum absolute normalized error $||\Delta\Gamma_d||_\infty := ||\Gamma_{d,\text{iterative-parquet}} - \Gamma_{d,\text{exact}}||_\infty / ||\Gamma_{d,\text{exact}}||_\infty$ of the iterative parquet scheme with QTCI for the irreducible vertex Γ_d compared to the exact values for various grid sizes (2^{3R} grid points) in the case of $U = \beta = 1$, plotted (a) as a function of $D_{\max}$ after 30 iterations, and as a function of iteration number for (b) $D_{\max} = 100$ and (c) $D_{\max} = 200$. Dashed lines indicate the results from the dense grid calculation without QTCI. (d) The maximum absolute normalized error of Γ_d, shown close to the first divergence, for $\beta = 1.55, U = 2.3$ and $D_{\max} = 100$, up to a very large number of 2600 iterations.

A. Hubbard atom

The atomic limit of the Hubbard model gives rise to rich two-particle correlations giving insight into strong-coupling limit of the Hubbard model [1,40,51]. This atomic limit offers considerable simplification, as the vertex functions are analytically known and become independent of momentum [40]. Therefore, the Hubbard atom serves as an ideal first test case for exploring two-particle properties in strongly correlated electron systems using the QTT framework.

Following the iterative parquet scheme outlined in Fig. 1, we start from the exact fully irreducible vertices Λ_r, set $\Gamma_r = \Lambda_r, F_d = U, F_m = -U, F_s = 2U, F_t = \Lambda_t, G = G_0$, and use TCI to efficiently compress the data into QTTs. We then iterate the four BSEs, the parquet equation, and the SDE in QTT format by means of the discussed MPO operations, which leads to quick convergence of the results for $\beta = U = 1$.

Figure 11(a) shows the maximum absolute normalized error in Γ_d compared to the exact result after 30 iterations of the iterative parquet cycle with a set tolerance of 10^{-10} in the initial TCI. We can now disentangle the two sources of error—the finite size of the discrete frequency grid, corresponding to the error in the respective dense grid calculations indicated by dashed lines, and the QTT approximation. The error due to the QTT approximation for a specified maximum bond dimension can be identified as the difference between the QTT and the dense grid results. Remarkably, the QTT errors quickly converge toward the dense grid results, where larger bond dimensions are needed to reach lower errors for larger grids, e.g., in the case of $R = 9$ already with $D_{\max} = 180$, the error from the QTT approximation de facto vanishes, leading to the same result as the dense grid calculation. However, the difference is that in the dense grid calculations objects containing $2^{3R} \simeq 1.34 \times 10^8$ data points need to be stored, while the QTTs stored for these parameters consist only of $\sim 8.5 \times 10^5$ elements leading to a compression ratio of $\mathcal{O}(10^2)$, remarkably, without any loss of accuracy.

If we allow for a small loss of accuracy, even more impressive compression ratios can be achieved, while at the same time reaching very low errors. For instance, it can be observed that already at a bond dimension of 100 maximum normalized errors, $<10^{-3}$ can be achieved. In the case of $R = 11$, this corresponds to a compression ratio of $\mathcal{O}(10^4)$, leading only to a tiny fraction of the required memory occupation in comparison to the dense grid calculations. Thus, we observe that the outlined framework can be used to efficiently solve the parquet equations in the compressed QTT format.

In Figs. 11(b) and 11(c), we show the maximum normalized errors for various grid sizes with respect to the iteration. At $D_{\max} = 100$, (b) the dense grid results (dashed lines) are only up to $R = 7$ exactly reproduced, while the errors of larger grid calculations level off below 10^{-3} in the vicinity of each other. Furthermore, we see that for larger grids slightly larger maximum bond dimensions are required to obtain the same level of error. In comparison, at $D_{\max} = 200$ (c) the QTT calculations converge toward the resulting dense grid errors also up to $R = 9$.

Next, we show that computations can also be performed for a more challenging case with the parameters $\beta = 1.55, U = 2.3$, which are chosen in the same way as in Ref. [13]. This case is interesting since $\beta U = 3.565$ is very close to the point, where the irreducible vertex in the density channel diverges ($\beta U \approx 3.628$) [45,56–60]. We show the results of these calculations in Fig. 11(d) for a maximum bond dimension $D_{\max} = 100$, where the maximum normalized error of Γ_d is shown with respect to the iteration. Since the parameters are very close to the first divergence line, we use a small mixing parameter $\alpha = 0.01$, leading to convergence only after 2600 iterations. In accordance with (b), we observe that in the case of $R \leqslant 6$ the error plateaus at decreasing levels for increasing R, which is due to the increased box size reflecting the results from the dense grid calculation. However, for $R \geqslant 7$ the situation changes, where the leveled-off errors are closer together. This means that the error is now governed by the QTT approximation (i.e., by $D_{\max}$) and not by the finite box size anymore. Still, it can be seen that already a maximum bond dimension of 100 is sufficient to reach absolute normalized errors $<10^{-3}$ for larger values of R, similar as in (b).

The above analysis demonstrates that during BSE iterations, errors due to the QTT representation do not accumulate—if they did, the solid lines in Figs. 11(b) and 11(c) would slope upward with increasing iteration number.

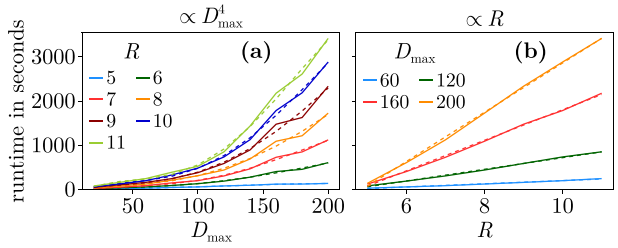


FIG. 12. Runtime of a single iteration of the iterative parquet scheme for various maximum bond dimensions and grid size parameters R in the case of $\beta = U = 1$. The dashed lines indicate (a) the quartic runtime scaling with $D_{\max}$ (governed by the BSEs) and (b) linear increase with R.

Instead, the error saturates at a value governed by the maximum of the error due to the Matsubara sum truncation and the initial QTT approximation. The accuracy of the results can be improved systematically by increasing the maximum bond dimension. In principle, this can also be done during the course of the BSE iterations if these generate structures of increasing complexity, but that was not necessary for the calculations presented here.

Our calculations were performed on a single 512 GB node on a cluster without parallelization, where in Fig. 12 the runtime of a single iteration of the iterative parquet scheme is shown. For the dense grid calculations, performing the same iterative parquet cycle was possible only up to $R = 9$ due to the exponentially increasing memory demand. In contrast, using the QTT approach, calculations for $R = 11$ were easily carried out on this single node without parallelization. This demonstrates the advantage of QTTs, where memory occupation and operations scale logarithmically with increasing resolution [see Fig. 12(b)], in contrast to the rapid growth in memory and computational costs encountered in standard methods. This allows for efficient computation on large grids, providing a significant advantage over dense grid implementations.

B. Single-impurity Anderson model

After solving the parquet equations for the simplified Hubbard atom case, we extend the QTT framework to the more complex SIAM, where a Hubbard atomlike interacting site is coupled to a bath of noninteracting electrons. Using the parquet approximation in which the fully irreducible vertex is approximated by the bare interaction ($\Lambda_d = U$, $\Lambda_m = -U$, $\Lambda_s = 2U$, $\Lambda_t = 0$), we iteratively evaluate the four BSEs (12), the parquet equation (15), and the SDE (17) with QTTs. Starting from $\Gamma_r = \Lambda_r$, $F_r = \Lambda_r$, $G = G_0$, we decompose the relevant functions into QTTs using TCI and then make use of the discussed MPO operations to iteratively solve the parquet equations. To ensure full convergence, we perform 60 iterations with a linear mixing $\alpha = 0.4$. The results presented below were obtained for $\beta = 10$, $U = 1$, $V = 2$, $D = 10$, and half filling, i.e., with $\varepsilon_d = -U/2$. For these parameters, the SIAM is in the weakly correlated regime, where the parquet approximation still holds. We compare our results with reference data obtained for the parquet approximation with the state-of-the-art parquet equations implementation on large

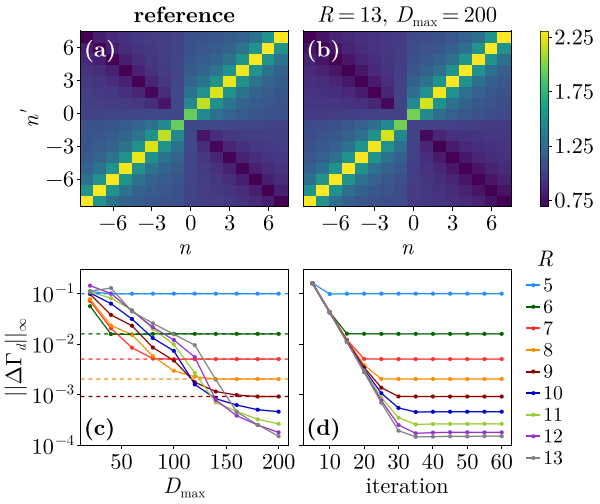


FIG. 13. Irreducible vertex Γ_d of the reference data (a) compared to the iterative parquet approximation calculations for $R = 13, D_{\max} = 200$ (b) for $\beta = 10, U = 1, V = 2, D = 10, \epsilon = 10^{-6}$, and a mixing of 0.4. (c), (d) The maximum normalized error of Γ_d ($||\Delta\Gamma_d||_\infty := ||\Gamma_{d,\text{iterative-parquet}} - \Gamma_{d,\text{ref}}||_\infty / ||\Gamma_{d,\text{ref}}||_\infty$) with respect to the reference data is shown (c) as a function of the maximum bond dimension $D_{\max}$ and (d) dependent on the iteration for $D_{\max} = 200$, where dashed lines indicate the obtained errors from dense grid calculations.

equidistant frequency grids of Ref. [3] using the single- and multiboson exchange formulation [12].

Figure 13(b) shows the irreducible vertex Γ_d at $\omega = 0$ calculated for $R = 13$ and a maximum bond dimension $D_{\max} = 200$, which is in good agreement with reference data in (a). In (c), we show the maximum normalized error of Γ_d obtained from the QTT calculations in comparison to the reference data depending on the maximum bond dimension for various grid sizes. In agreement with the results for the Hubbard atom, it can be observed that the errors of the QTT calculations converge toward the results of the dense grid calculations, which are indicated by dashed lines. Like in the Hubbard atom case, we exactly reproduce the dense grid results at $R = 9$ and $D_{\max} = 180$, leading to a $\mathcal{O}(10^2)$ compression ratio de facto without any loss in accuracy due to the QTT approximation. Moreover, since these calculations were performed up to $R = 13$, very large compression ratios of $\mathcal{O}(10^5)$ can be reached, while obtaining a maximum normalized error $<10^{-3}$. In (d), the maximum normalized error with respect to the iteration for $D_{\max} = 200$ can be observed. We show that for larger grids more iterations are needed to converge due to approaching smaller errors.

Finally, let us mention that the performed calculations for $R = 13$ would correspond to dense grid calculations with multiple objects of the size of $8 \times 2^{3\times 13} \simeq 4.4 \times 10^{12}$ bytes. Instead of the necessity of engaging multiple nodes and making use of parallelization for the dense grid calculations, using the QTT framework it was possible to perform these calculations on a single node without parallelization. Applying this approach allowed us to obtain maximum normalized errors $<10^{-3}$, while using only a tiny fraction of the memory required for the corresponding dense grid computations. This shows the computational advantage of the QTT approach.

C. Technical limitations

In our calculations, there are two main sources of error: (1) the finite size of the discrete Matsubara frequency grid and (2) the QTT approximation, which is governed by the maximum bond dimension. The finite grid size determines how accurately we can evaluate the BSEs and SDE, as it controls the truncation of the infinite Matsubara sums. On the other hand, the QTT approximation dictates how accurately the data can be represented.

The combined QTT and TCI approach exhibits logarithmic scaling in both memory and computational costs relative to the grid size, enabling the potential to handle very large grids (e.g., $R = 20$). While this is theoretically feasible, our explicit calculations for the Hubbard atom show that for such large values of R, the error increases significantly compared to the smaller grid sizes used in this study. This issue is not inherent to the method itself but arises due to the current implementation of MPO-MPO contractions, which relies on bond dimension truncation via SVD. The SVD truncation suffers from a loss of accuracy, such as round-off errors, because the Frobenius norm of the vertex functions diverges at large R due to a constant term in the frequency domain. This limitation can be addressed in future work by switching to a CI-based truncation approach [37]. For more details on this technical aspect, we refer readers to Appendix F. Alternatively, the vertex asymptotics can be explicitly removed from parquet equations as in Ref. [12] or [10].

Finally, it is important to note that the maximum bond dimension directly governs the accuracy of the QTT approximation, as it reflects how compressible the data are. Large bond dimensions can significantly increase computational costs, making them the primary bottleneck for scaling up the calculations.

VII. CONCLUSION AND OUTLOOK

This paper represents a large step forward in solving many-body problems with quantum field theory methods in QTT representations. The chosen example, the self-consistent solution of parquet equations, is a challenging one, requiring both efficiency in constructing the QTT representation of two-particle vertices and in evaluation of matrix multiplications and variable shifts within this representation. At the same time, the parquet equations for the simplest model, the Hubbard atom, can be solved analytically, allowing for careful benchmarking and assessment of the performance at each step of the solution separately. In this paper, we have numerically shown that the QTT representation of the vertex frequency dependence is suitable for solving the parquet equations and that, together with TCI, it leads to only logarithmic scaling in the grid size and with fourth power in the maximum bond dimension. For the two examples of Hubbard atom and SIAM, we observed that the bond dimension of $\sim$100–200 is enough to obtain the solution with high accuracy. For the case of Hubbard atoms, we see a saturation (or even decrease) of the bond dimension with increasing the inverse temperature β. We also expect a saturation or only slow growth of the bond dimension with β in more general cases, as conjectured in Ref. [34].

The naïve iteration of the parquet equations is difficult to converge in some regimes, cf. Sec. III F. While this problem is almost orthogonal to questions of representation and compression of the vertices, QTTs offer potential synergies: In particular, the greatly reduced size of QTT vertices may enable a solver to keep a convergence history and use nonlinear mixing schemes, which have been shown to stabilize convergence in Hartree-Fock [61] or quasi-Newton solvers, which have been able to access previously hidden solutions in self-consistent diagrammatic theories [62].

Although for explicit testing we have chosen models having no other degrees of freedom than frequencies (no momentum or orbital dependence), the dissection of the parquet equations solver into operations on QTTs—TCI compression and MPO-MPO contractions—is general and the extension to lattice models is straightforward. All results presented here were obtained on a single core with 512 GB memory and the grid sizes in each frequency variable were up to 2^{20}. This high compressibility of the frequency dependence of vertex functions can in the future be exploited (i) to solve parquet equations for lattice systems with high momentum resolution and orbital degrees of freedom needed to address material properties (see Appendix G for a brief discussion of how to deal with such additional degrees of freedom) and (ii) to apply QTCI to other vertex-based methods, such as the functional renormalization group (fRG) [63–65]; ladder extensions [42] or fRG extensions [66] of dynamical mean-field theory (possibly using as input results for the local vertex obtained using the numerical renormalization group [2,67,68]), embedded multiboson exchange methods [69], or the Migdal-Eliashberg theory in *ab initio* calculations [70,71].

On a more general note, two-particle objects are central in many more applications involving interacting electrons: In particular, the two-electron integrals $[ij|kl]$ are central to quantum chemistry, whereas the renormalized interaction W_{ijkl} is one of the main ingredients of GW [72]. Neither of these objects have the intricate three-frequency structure of the vertex F, however, they do depend on four orbital (spatial) indices. Handling this dependence is usually the main computational bottleneck in self-consistent field computations, even with sophisticated mitigation techniques [73]. The present paper offers a blueprint for applying QTTs to these methods and is a promising topic for future study.

ACKNOWLEDGMENTS

This work was funded in part by the Austrian Science Fund (FWF) Projects No. P 36332 (Grant No. DOI 10.55776/P36332) and No. V 1018 (Grant No. DOI 10.55776/V1018). For open access purposes, the authors have applied a CC BY public copyright license to any author-accepted manuscript version arising from this submission. This work was funded in part by the Deutsche Forschungsgemeinschaft under Germany's Excellence Strategy EXC-2111 (Project No. 390814868). M.K.R. and J.V.D. acknowledge support from DFG Grant No. LE 3883/2-2. This work is part of the Munich Quantum Valley, supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus. H.S. was supported by JSPS KAKENHI

Grants No. 21H01041, No. 21H01003, No. 22KK0226, and No. 23H03817 as well as JST FOREST Grant No. JPMJFR2232 and JST PRESTO Grant No. JPMJPR2012, Japan. Calculations have been partly performed on the Vienna Scientific Cluster (VSC) using the `Quantics.jl` and `TensorCrossInterpolation.jl` libraries.

DATA AVAILABILITY

The raw data for the figures and the data that support the findings of this paper are openly available [74].

APPENDIX A: PARQUET EQUATION AND FREQUENCY CONVENTIONS

The parquet equation (15) gives the full vertex F as a simple sum of the fully irreducible vertex Λ and vertices reducible in the ph, pp, and $\overline{ph}$ channels. The reducible vertices, however, are represented in their "channel native" frequency parametrization. After applying the parametrization changes and collecting the spin components, the final expressions are linear combinations of different spin components and frequency shifted arguments. For F_d, see Eq. (16), and for the remaining three spin combinations we have

$$F_m^{\nu\nu'\omega} = \Lambda_m^{\nu\nu'\omega} + \Phi_m^{\nu\nu'\omega} - \tfrac{1}{2}\Phi_d^{\nu(\nu+\omega)(\nu'-\nu)} + \tfrac{1}{2}\Phi_m^{\nu(\nu+\omega)(\nu'-\nu)} - \tfrac{1}{2}\Phi_s^{\nu\nu'(-\omega-\nu-\nu')} + \tfrac{1}{2}\Phi_t^{\nu\nu'(-\omega-\nu-\nu')}, \tag{A1a}$$

$$F_s^{\nu\nu'\omega} = \Lambda_s^{\nu\nu'\omega} + \Phi_s^{\nu\nu'\omega} + \tfrac{1}{2}\Phi_d^{\nu\nu'(-\omega-\nu-\nu')} - \tfrac{3}{2}\Phi_m^{\nu\nu'(-\omega-\nu-\nu')} + \tfrac{1}{2}\Phi_d^{\nu(-\nu'-\omega)(\nu'-\nu)} - \tfrac{3}{2}\Phi_m^{\nu(-\nu'-\omega)(\nu'-\nu)}, \tag{A1b}$$

$$F_t^{\nu\nu'\omega} = \Lambda_t^{\nu\nu'\omega} + \Phi_t^{\nu\nu'\omega} + \tfrac{1}{2}\Phi_d^{\nu\nu'(-\omega-\nu-\nu')} + \tfrac{1}{2}\Phi_m^{\nu\nu'(-\omega-\nu-\nu')} - \tfrac{1}{2}\Phi_d^{\nu(-\nu'-\omega)(\nu'-\nu)} - \tfrac{1}{2}\Phi_m^{\nu(-\nu'-\omega)(\nu'-\nu)}. \tag{A1c}$$

The origin of the need for frequency shifts lies in the inherent incompatibility of the parquet equation viewpoint and the BSE viewpoint. In the BSE, we choose the frequency and spin parametrizations so we can eliminate at least one frequency and spin sum. This optimal parametrization is, however, different for generating ph- and pp-reducible diagrams. In the parquet equation, on the other hand, we need all vertices in the same frequency parametrization, hence the need for frequency channel transformations.

Additionally, we also need a transformation between ph and $\overline{ph}$ representations to obtain $\Phi_{\overline{ph}}$. This transformation exploits the so-called crossing symmetry relation between the ph and $\overline{ph}$ frequency channels.

1. Parquet picture

To have a closer look at where the frequency shifts originate, let us first extend the frequency dependence of each vertex by including a fourth fermionic Matsubara frequency ν_4, i.e., a frequency related to the fourth time variable in Eq. (8). It would multiply the time 0 in the exponent, so it is obviously redundant and given by the energy conservation $\nu_1 + \nu_2 + \nu_3 + \nu_4 = 0$. Let us reintroduce the four index notations for the spin variable and use the following combined notation (as in e.g. Ref. [6]):

$$F(1,2,3,4) = F_{\sigma_1\sigma_2\sigma_3\sigma_4}(\nu_1,\nu_2,\nu_3,\nu_4). \tag{A2}$$

Then the parquet equation (15) is simply

$$F(1,2,3,4) = \Lambda(1,2,3,4) + \Phi^{ph}(1,2,3,4) + \Phi^{\overline{ph}}(1,2,3,4) + \Phi^{pp}(1,2,3,4). \tag{A3}$$

In this notation, the crossing symmetry of the full vertex is simply

$$F(1,2,3,4) = -F(1,4,3,2) = -F(3,2,1,4) \tag{A4}$$

and corresponds to exchanging variables of the two creation (annihilation) operators in the expectation value in Eq. (8) (in the language of diagrams one calls it exchanging two incoming or two outgoing lines). One can show that the reducible vertex in the pp channel is also crossing symmetric (and hence also the irreducible since $\Gamma = F - \Phi$). The crossing symmetry transformation applied to the ph channel, however, gives only the following relation:

$$\Phi^{\overline{ph}}(1,2,3,4) = -\Phi^{ph}(1,4,3,2). \tag{A5}$$

In the four frequency notations, the BSEs have the following form:

$$F(1,2,3,4) = \Gamma^{ph}(1,2,3,4) + \Phi^{ph}(1,2,3,4), \tag{A6a}$$

$$\Phi^{ph}(1,2,3,4) = \Gamma^{ph}(1,2,5,6)G(6,7)G(8,5)F(7,8,3,4),$$

$$F(1,2,3,4) = \Gamma^{pp}(1,2,3,4) + \Phi^{pp}(1,2,3,4),$$

$$\Phi^{pp}(1,2,3,4) = \tfrac{1}{2}\Gamma^{pp}(1,5,3,6)G(6,7)G(5,8)F(7,2,4,8), \tag{A6b}$$

where the summation over repeated arguments (5,6,7, and 8) is implied and we also used a two-frequency two-spin notation for the one-particle Green's function $G(1,2) = G_{\sigma_1\sigma_2}(\nu_1,\nu_2)$. This representation reveals the true difference between the ph and pp BSEs—the Green's functions connect the vertices differently, i.e., different frequency arguments are summed over. Practical evaluation of these equations, however, requires the introduction of two different three-frequency (two fermionic, one bosonic) parametrizations. These parametrizations are sometimes called *channel native*.

2. Bethe–Salpeter picture and channel native description

Following Ref. [40], in this paper we use the following frequency convention for the ph and pp channels:

$$\begin{aligned} \textbf{ph}:&\ \nu_1 = -\nu, & \textbf{pp}:&\ \nu_1 = -\nu,\\ &\ \nu_2 = \nu+\omega, & &\ \nu_2 = -(\nu'+\omega),\\ &\ \nu_3 = -(\nu'+\omega), & &\ \nu_3 = \nu+\omega,\\ &\ \nu_4 = \nu'. & &\ \nu_4 = \nu'. \end{aligned} \tag{A7}$$

Applying the above parametrizations to Eqs. (A6) and additionally introducing the $d/m/s/t$ spin combinations leads to Eqs. (12). Now, however, we need channel transformations

(frequency shifts) to evaluate the parquet equation. We need F both in the ph and pp notations for the d/m and s/t channels, respectively. The channel transformations can be derived by going back and forth from three frequency to four frequency representations, e.g.,

$$\begin{aligned} F_{pp}(\nu, \nu', \omega) &= F_{pp}(-\nu_1, \nu_4, \nu_1 + \nu_2) \\ &= F_{ph}(-\nu_1, \nu_4, -\nu_2 - \nu_4) \\ &= F_{ph}(\nu, \nu', -\omega - \nu - \nu'), \end{aligned} \tag{A8}$$

from which we deduce

$$\begin{aligned} ph &\longrightarrow pp, \\ (\nu, \nu', \omega) &\longrightarrow (\nu, \nu', -\omega - \nu - \nu'). \end{aligned} \tag{A9}$$

To use the crossing symmetry relation (A5), we also need the ph to $\overline{ph}$ channel transformation. The crossing transformation means exchanging either first and third or second and fourth frequencies, so we can write

$$\begin{aligned} F_{\overline{ph}}(\nu, \nu', \omega) &= F_{\overline{ph}}(-\nu_1, \nu_4, \nu_1 + \nu_2) \\ &= F_{ph}(-\nu_1, \nu_2, \nu_1 + \nu_4) \\ &= F_{ph}(\nu, \nu + \omega, \nu' - \nu), \end{aligned} \tag{A10}$$

from which we deduce

$$\begin{aligned} ph &\longrightarrow \overline{ph}, \\ (\nu, \nu', \omega) &\longrightarrow (\nu, \nu + \omega, \nu' - \nu). \end{aligned} \tag{A11}$$

All channel transformations needed in Eqs. (16) and (A1) are outlined in Appendix C, together with their numerical implementation.

APPENDIX B: AFFINE TRANSFORMATIONS

An important subset of transformations on a QTT are coordinate transformations, in particular, affine transformations. In this Appendix, we show how to efficiently construct an MPO (B6) for such a transformation.

Rather than striving for full generality, we limit our discussion to the type of affine transformations needed in this paper: transformations between the native frequency representations for the ph, pp, and $\overline{ph}$ channels. In practice, we limit the range of Matsubara frequencies to a finite box. The frequencies within that box can be enumerated by positive integers. Upon transforming to another channel, some frequencies will be mapped to lie outside the frequency box of the new channel, causing missing information in the mapping. Those frequency points then have to be dropped (open boundary conditions) or periodically continued (periodic boundary conditions). With open boundary conditions, the mapping will generically become noninvertible. For the remaining frequencies, the channel transformation maps one constrained set of positive integers to another.

We formalize the above scenario as follows. Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be vectors with N components. An affine transformation is a map $\boldsymbol{x} \mapsto \boldsymbol{y}$ that can be represented as

$$\boldsymbol{y} = A\boldsymbol{x} + \boldsymbol{b}, \tag{B1}$$

where A is an invertible $N \times N$ matrix. In the following, we limit our description to the case relevant for channel transforms, where all components of $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{b}, A$ are integers. We further constrain ourselves to the case where A^{-1} has integer components and the components of $\boldsymbol{b}$ are nonnegative. Given a function $g(\boldsymbol{y})$, we construct a new function $f(\boldsymbol{x})$ by a coordinate transformation:

$$f(\boldsymbol{x}) := g(\boldsymbol{y}(\boldsymbol{x})). \tag{B2}$$

We call this type of transformation a *passive* affine transformation, where for a given $\boldsymbol{x}$, we define the value of the new function $f(\boldsymbol{x})$ by picking the value of the old function $g(\boldsymbol{x})$ at the transformed point $\boldsymbol{y}$. In practice, we limit $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{b}$ to a finite box $S = \{0, \ldots, 2^R - 1\}^N$. Then, some $\boldsymbol{x}$ may be transformed to a $\boldsymbol{y}$ outside the box, in which case the choice of periodic or open boundary conditions becomes relevant. With periodic boundary conditions, we interpret Eq. (B1) as $\boldsymbol{y} \equiv A\boldsymbol{x} + \boldsymbol{b} \pmod{2^R}$, where $\pmod{2^R}$ is to be understood componentwise. With open boundary conditions, we set $f(\boldsymbol{x}) = 0$ if $\boldsymbol{y} \notin S$. The transformation (B1) is not necessarily invertible on S, even if it is invertible on $\mathbb{Z}^N$.

We can write Eq. (B2) as a tensor product

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in S} T(\boldsymbol{x}, \boldsymbol{y}) g(\boldsymbol{y}), \tag{B3}$$

with

$$T(\boldsymbol{x}, \boldsymbol{y}) := \begin{cases} 1 & \boldsymbol{y} = A\boldsymbol{x} + \boldsymbol{b} \\ 0 & \text{else.} \end{cases} \tag{B4}$$

In quantics representation, the tensor T can be factorized to an MPO with small bond dimension, which allows cheap transformation of functions given in QTT format through a single MPO-MPS contraction. To construct this MPO, it is useful to start with *fused* rather than interleaved indices, i.e., we only separate out the length scales of the vector, but not its components:

$$\boldsymbol{x} = \sum_{r=1}^{R} 2^{R-r} \boldsymbol{x}_r, \tag{B5}$$

where $\boldsymbol{x}_r$ is a vector of N bits corresponding to the current scale, i.e., $\boldsymbol{x}_r \in \{0, 1\}^N$. Thus, the legs $\boldsymbol{x}_r$ of the corresponding MPS are of dimension 2^N rather than 2. We perform similar decompositions for $\boldsymbol{y}$ and $\boldsymbol{b}$. Consequentially, T is decomposed as

$$T(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\alpha_1=1}^{D_1} \cdots \sum_{\alpha_{R-1}=1}^{D_{R-1}} [T_1]^{\boldsymbol{x}_1 \boldsymbol{y}_1}_{1\alpha_1} [T_2]^{\boldsymbol{x}_2 \boldsymbol{y}_2}_{\alpha_1 \alpha_2} \cdots [T_R]^{\boldsymbol{x}_R \boldsymbol{y}_R}_{\alpha_{R-1} 1}, \tag{B6}$$

where $[T_r]^{\boldsymbol{x}_r \boldsymbol{y}_r}_{\alpha_{r-1}\alpha_r}$ is the rth core tensor with virtual indices α_{r-1} and α_r as well as local indices $\boldsymbol{x}_r$ and $\boldsymbol{y}_r$. The corresponding tensor network diagram is shown in Fig. 14. The indices are bound by $\alpha_r \in \{1, \ldots, D_r\}, \boldsymbol{x}_r, \boldsymbol{y}_r \in \{0, 1\}^N$. Once the MPO is constructed in this way, we can transform it to the interleaved

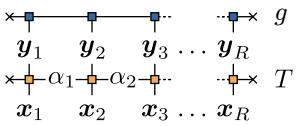


FIG. 14. Affine transform $T(\boldsymbol{y}, \boldsymbol{x})$ applied to function $g(\boldsymbol{x})$ in MPO form.

representation by splitting the core tensors using a QR decomposition.

For the explicit construction of the MPO, we first decompose Eq. (B1) for the finest scale $r = R$:

$$2\boldsymbol{c}_{R-1} + \boldsymbol{y}_R = A\boldsymbol{x}_R + \boldsymbol{b}_R, \tag{B7}$$

where $\boldsymbol{c}_{R-1}$ is the carry, a vector of integers not confined to 0 and 1. Since all components of $2\boldsymbol{c}_{R-1}$ are even, we find that legal values of $\boldsymbol{y}_R$ must satisfy

$$\boldsymbol{y}_R \equiv A\boldsymbol{x}_R + \boldsymbol{b}_R \pmod 2, \tag{B8a}$$

where (mod 2) is to be understood componentwise. Consequently, the carry is obtained as

$$\boldsymbol{c}_{R-1} = \tfrac{1}{2}(A\boldsymbol{x}_R + \boldsymbol{b}_R - \boldsymbol{y}_R). \tag{B8b}$$

The carry $\boldsymbol{c}_{R-1}$ enters the calculation for the next scale $R-1$, so we have to communicate it to the previous core tensor T_{R-1} via the bond. To do so, we first observe that Eqs. (B8) uniquely determine $\boldsymbol{y}_R$ and $\boldsymbol{c}_R$ for each $\boldsymbol{x}_R$. We collect all distinct values of the carry for all possible inputs $\boldsymbol{x}_R$ into a tuple $(\boldsymbol{c}_{R-1,1}, \ldots, \boldsymbol{c}_{R-1,D_{R-1}})$. The core tensor is then given by

$$[T_R]^{\boldsymbol{x}_R \boldsymbol{y}_R}_{\alpha 1} = \begin{cases} 1 & 2\boldsymbol{c}_{R-1,\alpha} + \boldsymbol{y}_R = A\boldsymbol{x}_R + \boldsymbol{b}_R \\ 0 & \text{else.} \end{cases} \tag{B9}$$

For all the other scales r, we must add the incoming carry $\boldsymbol{c}_r$ and must thus amend Eq. (B7) to

$$2\boldsymbol{c}_{r-1} + \boldsymbol{y}_r = A\boldsymbol{x}_r + \boldsymbol{b}_r + \boldsymbol{c}_r, \tag{B10}$$

and Eqs. (B8) to

$$\boldsymbol{y}_r \equiv A\boldsymbol{x}_r + \boldsymbol{b}_r + \boldsymbol{c}_r \pmod 2, \tag{B11a}$$

$$\boldsymbol{c}_{r-1} = \tfrac{1}{2}(A\boldsymbol{x}_r + \boldsymbol{b}_r + \boldsymbol{c}_r - \boldsymbol{y}_r). \tag{B11b}$$

We again collect all distinct outgoing carry values for all possible $\boldsymbol{x}_r$ and $\boldsymbol{c}_r$ into $(\boldsymbol{c}_{r-1,\alpha})_{\alpha=1,\ldots,D_{r-1}}$, and obtain the core tensor

$$[T_r]^{\boldsymbol{x}_r \boldsymbol{y}_r}_{\alpha\alpha'} = \begin{cases} 1 & 2\boldsymbol{c}_{r-1,\alpha} + \boldsymbol{y}_r = A\boldsymbol{x}_r + \boldsymbol{b}_r + \boldsymbol{c}_{r,\alpha'} \\ 0 & \text{else.} \end{cases} \tag{B12}$$

We iterate this procedure from $r = R$ to 1, constructing all MPO core tensors in a single backward sweep. Having reached the first tensor, $r = 1$, we implement open boundary conditions by demanding that $\boldsymbol{c}_0 = \boldsymbol{0}$ in Eq. (B12), such that

$$[T_1]^{\boldsymbol{x}_1 \boldsymbol{y}_1}_{1\alpha'} = \begin{cases} 1 & \boldsymbol{y}_1 = A\boldsymbol{x}_1 + \boldsymbol{b}_1 + \boldsymbol{c}_{1,\alpha'} \\ 0 & \text{else.} \end{cases} \tag{B13}$$

Periodic boundary conditions are implemented by modifying Eq. (B12) such that the leftmost carry $\boldsymbol{c}_0$ is discarded, such that

$$[T_1]^{\boldsymbol{x}_1 \boldsymbol{y}_1}_{1\alpha'} = \begin{cases} 1 & \boldsymbol{y}_1 \equiv A\boldsymbol{x}_1 + \boldsymbol{b}_1 + \boldsymbol{c}_{1,\alpha'} \pmod 2 \\ 0 & \text{else.} \end{cases} \tag{B14}$$

The bond dimension $D_{\max} = \max_r D_r$ of the tensors constructed in this way is likely optimal. This algorithm has $\mathcal{O}(RD^2_{\max} 2^{2N})$ runtime, which is optimal in the sense that at least this amount of runtime and memory is necessary to construct the tensors T_r. The algorithm can be generalized to cases where A and $\boldsymbol{b}$ have entries in $\mathbb{Q}$.

TABLE I. Nonzero elements of $[T_r]^{\boldsymbol{x}_r \boldsymbol{y}_r}_{\alpha_{r-1}\alpha_r}$ (B6) constructed from Eqs. (B7)–(B13) for the affine transform (B15) with open boundary conditions.

r	α_{r-1}	$\boldsymbol{c}_{r-1}$	$\boldsymbol{y}_r$	$\boldsymbol{x}_r$	α_r	$\boldsymbol{c}_r$
R	1	(0, 0)	(0, 0)	(0, 0)	1	(0, 0)
	1	(0, 0)	(1, 1)	(1, 0)	1	(0, 0)
	2	(0, −1)	(0, 1)	(0, 1)	1	(0, 0)
	1	(0, 0)	(1, 0)	(1, 1)	1	(0, 0)
$R-1$	1	(0, 0)	(0, 0)	(0, 0)	1	(0, 0)
	1	(0, 0)	(1, 1)	(1, 0)	1	(0, 0)
	2	(0, −1)	(0, 1)	(0, 1)	1	(0, 0)
	1	(0, 0)	(1, 0)	(1, 1)	1	(0, 0)
	2	(0, −1)	(0, 1)	(0, 0)	2	(0, −1)
	1	(0, 0)	(1, 0)	(1, 0)	2	(0, −1)
	2	(0, −1)	(0, 0)	(0, 1)	2	(0, −1)
	2	(0, −1)	(1, 1)	(1, 1)	2	(0, −1)
⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	1	(0, 0)	(0, 0)	(0, 0)	1	(0, 0)
	1	(0, 0)	(1, 1)	(1, 0)	1	(0, 0)
	1	(0, 0)	(1, 0)	(1, 1)	1	(0, 0)
	1	(0, 0)	(1, 0)	(1, 0)	2	(0, −1)

Let us walk through the algorithm for the example transformation with $N = 2$ and open boundary conditions:

$$\boldsymbol{y} = \begin{pmatrix} 1 & 0 \\ 1 & -1 \end{pmatrix} \boldsymbol{x} + \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{B15}$$

The nonzero elements of the corresponding core tensors (B6) are listed in Table I. For the case $r = R$, we simply apply the transformation (B8a) to each bit combination $\boldsymbol{x}_R \in \{(0,0), (1,0), (0,1), (1,1)\}$. In the case $\boldsymbol{x}_R = (0,1)$, Eq. (B8b) yields a carry of $\boldsymbol{c}_{R-1} = (0,-1)$, to which we assign the bond index $\alpha_{R-1} = 2$, otherwise it is (0,0), to which we assign the index $\alpha_{R-1} = 1$. The dimension of the corresponding bond is thus $D_{R-1} = 2$ and the core tensor has the four nonzero elements listed in rows 1–4 of Table I.

For $r = R-1$, Eq. (B11) directs us to add the incoming carry $\boldsymbol{c}_r$. Hence, we double the number of nonzero entries, as we have to repeat the calculation for each of the two outgoing carries of T_R. We observe that the set of incoming and outgoing carries is identical and assign the same bond indices to them. The corresponding nonzero elements of T_{R-1} are then listed in rows 5–12 of Table I. Since the values of the outgoing carries form the same set as those of the incoming carries, all other core tensors $T_{r'}$ with $1 < r' < R$ are equal to T_{R-1}. For $r = 1$, we impose open boundary conditions, thereby restricting the outgoing carry of T_1 to zero, as shown in Eq. (B13). This cuts half of the elements and yields the entries listed in rows 13–16 of Table I.

APPENDIX C: CHANNEL TRANSFORMATIONS

We describe how to implement channel transformations using affine transformations in QTT, which is defined in Eq. (B2).

1. *ph* to *pp* transformation

We first describe a $ph \to pp$ channel transformation via the $\overline{ph}$ channel:

$$ph \longrightarrow \overline{ph} \longrightarrow pp,$$
$$(\nu, \nu', \omega) \longrightarrow (\nu, \nu+\omega, \nu'-\nu) \longrightarrow (\nu, \nu', -\omega-\nu-\nu'), \tag{C1}$$

where $\nu^{(\prime)} = (2n^{(\prime)}+1)\pi/\beta$ and $\omega = 2m\pi/\beta$. The picture corresponds to the following, e.g., for the full vertex:

$$F_{pp}(\nu, \nu', \omega) = F_{ph}(\nu, \nu', -\omega-\nu-\nu'), \tag{C2a}$$
$$F_{\overline{ph}}(\nu, \nu', \omega) = F_{ph}(\nu, \nu+\omega, \nu'-\nu), \tag{C2b}$$
$$F_{pp}(\nu, \nu', \omega) = F_{\overline{ph}}(\nu, -\nu'-\omega, \nu'-\nu). \tag{C2c}$$

In the following, we will denote the old variables in every transformation step with a tilde. For the ph to $\overline{ph}$ transformation

$$F_{\overline{ph}}(\nu, \nu', \omega) = F_{ph}(\nu, \nu+\omega, \nu'-\nu) = F_{ph}(\tilde{\nu}, \tilde{\nu}', \tilde{\omega}), \tag{C3}$$

we need the transformation matrix [expressing the old $(\tilde{\nu}, \tilde{\nu}', \tilde{\omega})$ by the new variables (ν, ν', ω)]

$$\begin{pmatrix} \tilde{\nu} \\ \tilde{\nu}' \\ \tilde{\omega} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ -1 & 1 & 0 \end{pmatrix} \begin{pmatrix} \nu \\ \nu' \\ \omega \end{pmatrix}, \tag{C4}$$

with $(\tilde{\nu}, \tilde{\nu}', \tilde{\omega}) = (\nu, \nu+\omega, \nu'-\nu)$ and $\tilde{\nu}^{(\prime)} = (2\tilde{n}^{(\prime)}+1)\pi/\beta$ and $\tilde{\omega} = 2\tilde{m}\pi/\beta$. We also need to shift the indices such that, for example, for $n = 0$ and $m = 0$ ($\nu + \omega = \frac{\pi}{\beta} = \tilde{\nu}'$), we are at $\tilde{n}' = 0$ again:

$$0 \leqslant a, b, c, \tilde{a}, \tilde{b}, \tilde{c} \leqslant N-1,$$
$$n = a - \frac{N}{2}, \quad n' = b - \frac{N}{2}, \quad m = c - \frac{N}{2}, \tag{C5}$$
$$\tilde{n} = \tilde{a} - \frac{N}{2}, \quad \tilde{n}' = \tilde{b} - \frac{N}{2}, \quad \tilde{m} = \tilde{c} - \frac{N}{2}, \tag{C6}$$

with $N = 2^R$. This leads to

$$\begin{aligned} \tilde{a} &= a, && \text{no shift,} \\ \tilde{b} &= a + b - \frac{N}{2} = n + m + \frac{N}{2}, && \text{shift by } \frac{N}{2} \\ \tilde{c} &= -a + b + \frac{N}{2} = -n + n' + \frac{N}{2}, && \text{shift by } \frac{N}{2}. \end{aligned} \tag{C7}$$

Hence, we get the shift vector $\boldsymbol{b} = (0, \frac{N}{2}, \frac{N}{2})^{\mathrm{T}}$. For example, at $n = n'$ we need $\omega = 0$ and, thus, $\tilde{c} = \frac{N}{2}$, which is ensured by the shift. This first transformation can be represented by an MPO with $D_{\max} = 9$.

For the $\overline{ph} \to pp$ transformation,

$$F_{pp}(\nu, \nu', \omega) = F_{\overline{ph}}(\nu, -\nu'-\omega, \nu'-\nu) = F_{\overline{ph}}(\tilde{\nu}, \tilde{\nu}', \tilde{\omega}), \tag{C8}$$

we need the transformation matrix

$$\begin{pmatrix} \tilde{\nu} \\ \tilde{\nu}' \\ \tilde{\omega} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & -1 \\ -1 & 1 & 0 \end{pmatrix} \begin{pmatrix} \nu \\ \nu' \\ \omega \end{pmatrix}, \tag{C9}$$

with $(\tilde{\nu}, \tilde{\nu}', \tilde{\omega}) = (\nu, -\nu'-\omega, \nu'-\nu)$. Using the same procedure as above, we get the shift vector $\boldsymbol{b} = (0, \frac{N}{2}-1, \frac{N}{2})^{\mathrm{T}}$.

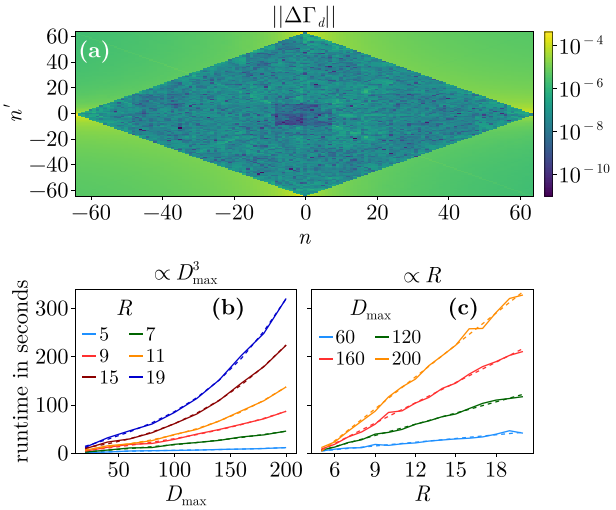


FIG. 15. (a) Absolute normalized error $||\Delta\Gamma_d|| := |\Gamma_{d,\text{parquet}} - \Gamma_{d,\text{exact}}|/||\Gamma_{d,\text{exact}}||_\infty$ of the parquet equation with QTTs for Γ_d compared to the exact values at $\omega = 0$ in the fermionic Matsubara frequency plane for $D_{\max} = 200, R = 7, \beta = U = 1, \epsilon = 10^{-8}$. Runtime of the parquet equation for Γ_d for various maximum bond dimensions and grid size parameters R with $\beta = U = 1$. The dashed lines indicate (b) the cubic runtime increase with $D_{\max}$ and (c) linear increase by increasing R, which corresponds to an exponential increase in the number of grid points.

This affine transformation can be represented by an MPO with $D_{\max} = 15$.

2. *pp* to *ph* transformation

Basically, we use the same procedure as above since the ph to pp transformation is its own inverse.

$$pp \longrightarrow \overline{pp} \longrightarrow ph,$$
$$(\nu, \nu', \omega) \longrightarrow (\nu, \nu+\omega, \nu'-\nu) \longrightarrow (\nu, \nu', -\omega-\nu-\nu'), \tag{C10}$$

The picture corresponds to the following, e.g., for the full vertex:

$$F_{ph}(\nu, \nu', \omega) = F_{pp}(\nu, \nu', -\omega-\nu-\nu'), \tag{C11a}$$
$$F_{\overline{pp}}(\nu, \nu', \omega) = F_{pp}(\nu, \nu+\omega, \nu'-\nu), \tag{C11b}$$
$$F_{ph}(\nu, \nu', \omega) = F_{\overline{pp}}(\nu, -\nu'-\omega, \nu'-\nu). \tag{C11c}$$

The transformation matrices and shift factors and, hence, the MPO representations are identical to the ph to pp transformation.

APPENDIX D: PARQUET EQUATION IN QTT FORMAT

In the parquet equation (16) and (A1), the triangle-shaped frequency box errors from frequency transformations add up to a diamond-shaped error with larger errors in the corners. This can be observed in Fig. 15, where a plot of the absolute normalized error of the irreducible vertex in the density channel Γ_d computed via Eq. (16) and (13) with QTCI compared to the exact Γ_d is shown at $\omega = 0$.

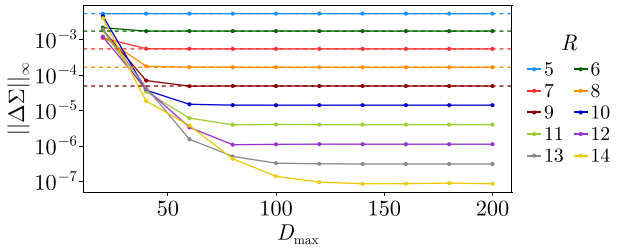


FIG. 16. Plot of the maximum absolute normalized error of the SDE $||\Delta\Sigma||_\infty := ||\Sigma_{d,\mathrm{SDE}} - \Sigma_{d,\mathrm{exact}}||_\infty / ||\Sigma_{d,\mathrm{exact}}||_\infty$ with QTTs for the self-energy Σ compared to the exact values for various grid sizes depending on the maximum bond dimension with $U = \beta = 1$. The dashed lines indicate the results from the dense grid calculation without QTTs.

The cubic dependence on $D_{\max}$ is shown in Fig. 15(b), which corresponds to the cubic scaling of the channel transformations inside the parquet equations and, thus, constitutes the bottleneck of the parquet equation. Linear scaling of the runtime of the parquet equation with QTTs in R is shown in Fig. 15(c). Again, we want to emphasize that exponentially increasing the number of grid points comes only at linearly increasing computational cost in the parquet equation.

APPENDIX E: SDE IN QTT FORMAT

In Fig. 16, we show the maximum absolute normalized error of the self-energy Σ obtained by using QTTs in the SDE. The dashed lines represent the errors of the dense grid calculations, which are due to the finite size of the grid. A qualitatively similar behavior to the BSE can be observed, with the difference that already quite low bond dimensions are sufficient for obtaining very low errors. This is caused by the frequency dependence of the functions in the SDE, where only the full vertex depends on three Matsubara frequencies. At this point, we should emphasize the fact that exponentially increasing the number of grid points by increasing the grid parameter R exponentially reduces the error (exponential convergence to the true solution), but only comes with linearly increasing computational cost. In the case of $R = 14$, the computation with a maximum bond dimension of 100 took only around 100 seconds using QTTs on a single 512 GB node on a cluster, while without the use of QTTs the calculation would include computations with the numerical data of the full vertex, which is the size of $8 \times 2^{3\times 14} \simeq 3.5 \times 10^{13}$ bytes. This would only be possible by engaging a larger number of nodes on a cluster, which emphasizes the strength of the QTCI approach.

APPENDIX F: TECHNICAL LIMITATIONS

Theoretically, in the iterative parquet calculations with QTCI, it should easily be possible to run calculations for much larger grids, e.g., $R = 20$ ($2^{3\times 20}$ grid points), on a single 512 GB node on a cluster, without running into any memory or computational time bottlenecks, because the computational costs only depend linearly on R. This is still true, but there is another limiting technical difficulty at the moment.

In the calculations, a specified maximum bond dimension is set not only in the initial TCI but also in every QTT operation to avoid a blowing up of the bond dimensions since this is the computational bottleneck. As was shown in Sec. V A, for the initial TCI already bond dimensions slightly above 100 are sufficient to reach maximum normalized errors of 10^{-6} of the QTTs with respect to the exact data. However, a problem emerges in the QTT operations, which are at the moment SVD based and make use of the `truncate` function in `ITensors.jl` to compress the resulting QTTs back to a certain maximum bond dimension. Furthermore, the fit algorithm used for MPO-MPO contractions relies on the SVD truncation internally [48]. The SVD truncation minimizes the difference between an original MPS and an approximated one in terms of the Frobenius norm. Because the Frobenius norm of the vertex functions grows exponentially with R due to a constant term, the SVD truncation is expected to fail at large R; the Frobenius norm reaches $c(2^R)^3 \approx c \times 3 \times 10^{13}$ at $R = 15$ (c is the constant term).

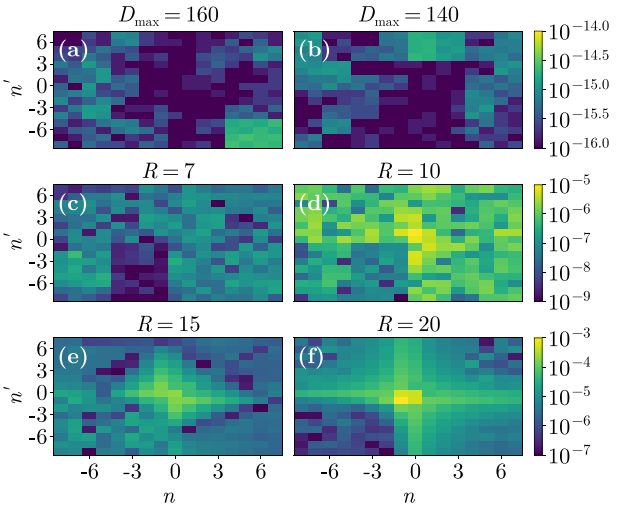


FIG. 17. Absolute normalized error of the reconstructed F_d compared to the exact data for the innermost 16×16 fermionic Matsubara frequency grid at $\omega = 0$, $\beta = U = 1$. (a), (b) The errors after using TCI with a set tolerance of 10^{-10} and the maximum bond dimensions set to 160 and 140, respectively. (c)–(f) The errors after applying the SVD based truncation to the QTT with maximum bond dimension 160 (a) with a set maximum bond dimension of 140. The truncation error increases with larger values of the grid parameter R.

Here, we have observed the truncation error to become more significant the larger the value of the grid parameter R is and even leads to wrong results around $R = 15$. In Fig. 17, we show this behavior in case of the full vertex in the density channel F_d. In Figs. 17(a) and 17(b), the absolute normalized error is shown after applying TCI to evaluate a QTT for F_d for different maximum bond dimensions. It can be seen that the error in this center (16×16) fermionic Matsubara frequency box is of $\mathcal{O}(10^{-15})$. Figures 17(c)–17(f) show the errors after applying the SVD-based truncation to the QTT with maximum bond dimension 160 down to a maximum bond dimension of 140 for various grid sizes determined by R. Although using TCI with a maximum bond dimension of 140 [Fig. 17(b)] the QTT was able to reconstruct the exact data with an absolute normalized error of $\mathcal{O}(10^{-15})$, applying the SVD-based truncation significantly worsens the results, leading to normalized errors between 10^{-9} and 10^{-5}.

Moreover, it can be seen that the error increases significantly with increasing R. This is why in the case of the iterative parquet solutions for the Hubbard atom, only results up to $R = 11$ are shown, since the resulting maximum normalized error does not improve anymore for larger grids due to the truncation errors. This can also already seen in Fig. 11 for $R = 11$, where the maximum normalized error at a maximum bond dimension of 200 is only slightly lower than in the case of $R = 10$. However, this should only be a problem of the current implementation and first numerical tests indicate that it is possible to overcome this limitation in the future, e.g., by using CI-based truncation [37]. This is because the CI-based truncation relies on the maximum norm and thus does not suffer from the divergence of the Frobenius norm.

An alternative way to deal with the infinite Frobenius norm is, as mentioned at the end of Sec. VI, to change into a formalism with vertices with removed asymtotics and thus finite Frobenius norm. The recent reformulation of parquet equations into single- and multiboson exchange vertices provides such a solution [3,12]. Earlier approaches to parquet equations have used the so-called kernel asymptotics [10,44].

APPENDIX G: MODEL EXTENSIONS

In future work, our goal will be to apply the QTT representation to two-particle calculations with orbital and momentum degrees of freedom. Suppose that these are labeled by an additional (composite) index, say $i = 1, \ldots, N$, then the vertex carries four such indices, F_{ijkl}, and has N^4 components. The memory costs for such computations depend on how QTTs are used to parametrize the vertex.

For example, a naive approach would be to use a separate QTT to parametrize the frequency dependence of $F_{ijkl}(\nu, \nu', \omega)$ for each index combination (i, j, k, l). This would require N^4 different QTTs and N^4 BSEs connecting them all, etc. We estimate that with this approach, computations for $N = 6$, $R = 10$ and $D_{\max} = 200$ should be feasible on a single 512 GB node.

However, such a naive approach would not exploit low-rank structures that may arise if different vertex components have similar frequency dependencies. In such a case, it could be more efficient to use a single QTT to parametrize the dependence of the vertex on its frequencies *and* all i indices. To pursue such a strategy and optimize its efficiency, further methodological developments will be required to address some open questions: What is the best grouping and order of quantics indices for a combined frequency and momentum (orbital) representation? How can MPO-MPO contractions (the current bottleneck) be performed more efficiently? What are the best strategies for parallelizing the computations? These issues are currently being explored in ongoing work.

[1] G. Rohringer, A. Valli, and A. Toschi, Local electronic correlation at the two-particle level, Phys. Rev. B **86**, 125114 (2012).

[2] Seung-Sup B. Lee, F. B. Kugler, and J. von Delft, Computing local multipoint correlators using the numerical renormalization group, Phys. Rev. X **11**, 041007 (2021).

[3] F. Krien and A. Kauch, The plain and simple parquet approximation: Single-and multi-boson exchange in the two-dimensional Hubbard model, Eur. Phys. J. B **95**, 69 (2022).

[4] C. De Dominicis and P. C. Martin, Stationary entropy principle and renormalization in normal and superfluid systems. I. Algebraic formulation, J. Math. Phys. **5**, 14 (1964).

[5] C. De Dominicis and P. C. Martin, Stationary entropy principle and renormalization in normal and superfluid systems. II. Diagrammatic formulation, J. Math. Phys. **5**, 31 (1964).

[6] N. E. Bickers, Self-consistent many-body theory for condensed matter systems, in *Theoretical Methods for Strongly Correlated Electrons*, edited by D. Sénéchal, A.-M. Tremblay, and C. Bourbonnais (Springer, New York, 2004), pp. 237–296.

[7] K.-M. Tam, H. Fotso, S.-X. Yang, T.-W. Lee, J. Moreno, J. Ramanujam, and M. Jarrell, Solving the parquet equations for the Hubbard model beyond weak coupling, Phys. Rev. E **87**, 013311 (2013).

[8] C. J. Eckhardt, C. Honerkamp, K. Held, and A. Kauch, Truncated unity parquet solver, Phys. Rev. B **101**, 155104 (2020).

[9] G. V. Astretsov, G. Rohringer, and A. N. Rubtsov, Dual parquet scheme for the two-dimensional Hubbard model: Modeling low-energy physics of high-T_c cuprates with high momentum resolution, Phys. Rev. B **101**, 075109 (2020).

[10] N. Wentzell, G. Li, A. Tagliavini, C. Taranto, G. Rohringer, K. Held, A. Toschi, and S. Andergassen, High-frequency asymptotics of the vertex function: Diagrammatic parametrization and algorithmic implementation, Phys. Rev. B **102**, 085106 (2020).

[11] F. Krien, A. Valli, P. Chalupa, M. Capone, A. I. Lichtenstein, and A. Toschi, Boson-exchange parquet solver for dual fermions, Phys. Rev. B **102**, 195131 (2020).

[12] F. Krien, A. Kauch, and K. Held, Tiling with triangles: Parquet and $gw\gamma$ methods unified, Phys. Rev. Res. **3**, 013149 (2021).

[13] M. Wallerberger, H. Shinaoka, and A. Kauch, Solving the Bethe-Salpeter equation with exponential convergence, Phys. Rev. Res. **3**, 033168 (2021).

[14] H. Shinaoka, D. Geffroy, M. Wallerberger, J. Otsuki, K. Yoshimi, E. Gull, and J. Kuneš, Sparse sampling and tensor network representation of two-particle Green's functions, SciPost Phys. **8**, 012 (2020).

[15] H. Shinaoka, J. Otsuki, K. Haule, M. Wallerberger, E. Gull, K. Yoshimi, and M. Ohzeki, Overcomplete compact representation of two-particle Green's functions, Phys. Rev. B **97**, 205111 (2018).

[16] D. Kiese, H. U. R. Strand, K. Chen, N. Wentzell, O. Parcollet, and J. Kaye, Discrete Lehmann representation of three-point functions, Phys. Rev. B **111**, 035135 (2025).

[17] S. Mallat, A theory for multiresolution signal decomposition: The wavelet representation, IEEE Trans. Pattern Anal. Mach. Intell. **11**, 674 (1989).

[18] I. Daubechies, The wavelet transform, time-frequency localization and signal analysis, IEEE Trans. Inf. Theory **36**, 961 (1990).

[19] E. Moghadas, N. Dräger, A. Toschi, J. Zang, M. Medvidović, D. Kiese, A. J. Millis, A. M. Sengupta, S. Andergassen, and D.

Di Sante, Compressing the two-particle Green's function using wavelets: Theory and application to the Hubbard atom, Eur. Phys. J. Plus **139**, 700 (2024).

[20] I. V. Oseledets, Approximation of matrices with logarithmic number of parameters, Doklady Math. **80**, 653 (2009).

[21] B. N. Khoromskij, $O(d \log N)$-quantics approximation of N-d tensors in high-dimensional numerical modeling, Constr. Approximation **34**, 257 (2011).

[22] S. Dolgov, B. Khoromskij, and D. Savostyanov, Superfast Fourier transform using QTT approximation, J. Anal. Appl. **18**, 915 (2012).

[23] B. N. Khoromskij, *Tensor Numerical Methods in Scientific Computing*, 1st ed., Radon Series on Computational and Applied Mathematics (De Gruyter, Berlin, Boston, 2018), Vol. 19.

[24] N. Gourianov, M. Lubasch, S. Dolgov, Q. Y. van den Berg, H. Babaee, P. Givi, M. Kiffner, and D. Jaksch, A quantum inspired approach to exploit turbulence structures, Nat. Comput. Sci. **2**, 30 (2022).

[25] R. D. Peddinti, S. Pisoni, A. Marini, P. Lott, H. Argentieri, E. Tiunov, and L. Aolita, Complete quantum-inspired framework for computational fluid dynamics, arXiv:2308.12972.

[26] E. Kornev, S. Dolgov, K. Pinto, M. Pflitsch, M. Perelshtein, and A. Melnikov, Numerical solution of the incompressible Navier-Stokes equations for chemical mixers via quantum-inspired tensor train finite element method, arXiv:2305.10784.

[27] L. Hölscher, P. Rao, L. Müller, J. Klepsch, A. Luckow, T. Stollenwerk, and F. K. Wilhelm, Quantum-inspired fluid simulation of two-dimensional turbulence with GPU acceleration, Phys. Rev. Res. **7**, 013112 (2025).

[28] N. Gourianov, P. Givi, D. Jaksch, and S. B. Pope, Tensor networks enable the calculation of turbulence probability distributions, Sci. Adv. **11**, eads5990 (2025).

[29] E. Ye and N. F. G. Loureiro, Quantum-inspired method for solving the Vlasov-Poisson equations, Phys. Rev. E **106**, 035208 (2022).

[30] N. Jolly, Y. N. Fernández, and X. Waintal, Tensorized orbitals for computational chemistry, arXiv:2308.03508.

[31] H. Shinaoka, M. Wallerberger, Y. Murakami, K. Nogaki, R. Sakurai, P. Werner, and A. Kauch, Multiscale space-time ansatz for correlation functions of quantum systems based on quantics tensor trains, Phys. Rev. X **13**, 021015 (2023).

[32] H. Ishida, N. Okada, S. Hoshino, and H. Shinaoka, Low-rank quantics tensor train representations of Feynman diagrams for multiorbital electron-phonon model, arXiv:2405.06440.

[33] M. Murray, H. Shinaoka, and P. Werner, Nonequilibrium diagrammatic many-body simulations with quantics tensor trains, Phys. Rev. B **109**, 165135 (2024).

[34] H. Takahashi, R. Sakurai, and H. Shinaoka, Compactness of quantics tensor train representations of local imaginary-time propagators, SciPost Phys. **18**, 007 (2025).

[35] Y. Núñez Fernández, M. Jeannin, P. T. Dumitrescu, T. Kloss, J. Kaye, O. Parcollet, and X. Waintal, Learning Feynman diagrams with tensor trains, Phys. Rev. X **12**, 041018 (2022).

[36] A. Erpenbeck, W.-T. Lin, T. Blommel, L. Zhang, S. Iskakov, L. Bernheimer, Y. Núñez-Fernández, G. Cohen, O. Parcollet, X. Waintal, and E. Gull, Tensor train continuous time solver for quantum impurity models, Phys. Rev. B **107**, 245135 (2023).

[37] Y. Núñez Fernández, M. K. Ritter, M. Jeannin, J.-W. Li, T. Kloss, T. Louvet, S. Terasaki, O. Parcollet, J. von Delft, H. Shinaoka, and X. Waintal, Learning tensor networks with tensor cross interpolation: New algorithms and libraries, SciPost Phys. **18**, 104 (2025).

[38] M. Eckstein, Solving quantum impurity models in the nonequilibrium steady state with tensor trains, arXiv:2410.19707.

[39] M. K. Ritter, Y. Núñez Fernández, M. Wallerberger, J. von Delft, H. Shinaoka, and X. Waintal, Quantics tensor cross interpolation for high-resolution parsimonious representations of multivariate functions, Phys. Rev. Lett. **132**, 056501 (2024).

[40] P. Thunström, O. Gunnarsson, S. Ciuchi, and G. Rohringer, Analytical investigation of singularities in two-particle irreducible vertex functions of the Hubbard atom, Phys. Rev. B **98**, 235107 (2018).

[41] A. C. Hewson, *The Kondo Problem to Heavy Fermions*, Cambridge Studies in Magnetism (Cambridge University Press, Cambridge, England, 1993).

[42] G. Rohringer, H. Hafermann, A. Toschi, A. A. Katanin, A. E. Antipov, M. I. Katsnelson, A. I. Lichtenstein, A. N. Rubtsov, and K. Held, Diagrammatic routes to nonlocal correlations beyond dynamical mean field theory, Rev. Mod. Phys. **90**, 025003 (2018).

[43] K. Held, Beyond DMFT: Spin fluctuations, pseudogaps and superconductivity, in *Dynamical Mean-Field Theory of Correlated Electrons*, edited by E. Pavarini, E. Koch, A. Lichtenstein, and D. Vollhardt (Forschungszentrum Jülich GmbH Institute for Advanced Simulation, Jülich, Germany, 2022), pp. 11.1–11.30.

[44] G. Li, A. Kauch, P. Pudleiner, and K. Held, The victory project v1.0: An efficient parquet equations solver, Comput. Phys. Commun. **241**, 146 (2019).

[45] E. Kozik, M. Ferrero, and A. Georges, Nonexistence of the Luttinger-Ward functional and misleading convergence of skeleton diagrammatic series for Hubbard-like models, Phys. Rev. Lett. **114**, 156402 (2015).

[46] G. Li, N. Wentzell, P. Pudleiner, P. Thunström, and K. Held, Efficient implementation of the parquet equations: Role of the reducible vertex function and its kernel approximation, Phys. Rev. B **93**, 165103 (2016).

[47] I. V. Oseledets, Tensor-train decomposition, SIAM J. Sci. Comput. **33**, 2295 (2011).

[48] F. Verstraete and J. I. Cirac, Renormalization algorithms for quantum-many body systems in two and higher dimensions, arXiv:cond-mat/0407066.

[49] E. M. Stoudenmire and S. R. White, Minimally entangled typical thermal state algorithms, New J. Phys. **12**, 055026 2010.

[50] T. Schäfer, G. Rohringer, O. Gunnarsson, S. Ciuchi, G. Sangiovanni, and A. Toschi, Divergent precursors of the Mott-Hubbard transition at the two-particle level, Phys. Rev. Lett. **110**, 246405 (2013).

[51] T. Schäfer, S. Ciuchi, M. Wallerberger, P. Thunström, O. Gunnarsson, G. Sangiovanni, G. Rohringer, and A. Toschi, Nonperturbative landscape of the Mott-Hubbard transition: Multiple divergence lines around the critical endpoint, Phys. Rev. B **94**, 235108 (2016).

[52] P. Chalupa, T. Schäfer, M. Reitner, D. Springer, S. Andergassen, and A. Toschi, Fingerprints of the local moment formation and its Kondo screening in the generalized susceptibilities of many-electron problems, Phys. Rev. Lett. **126**, 056403 (2021).

[53] S. Rohshap, Analytical calculation of two-particle vertices for multiorbital Hubbard atom, Master's thesis, TU Wien, 2023.

[54] M. Pelz, S. Adler, M. Reitner, and A. Toschi, Highly non-perturbative nature of the Mott metal-insulator transition: Two-particle vertex divergences in the coexistence region, Phys. Rev. B **108**, 155101 (2023).

[55] S. X. Yang, H. Fotso, J. Liu, T. A. Maier, K. Tomko, E. F. D'Azevedo, R. T. Scalettar, T. Pruschke, and M. Jarrell, Parquet approximation for the 4×4 Hubbard cluster, Phys. Rev. E **80**, 046706 (2009).

[56] O. Gunnarsson, G. Rohringer, T. Schäfer, G. Sangiovanni, and A. Toschi, Breakdown of traditional many-body theories for correlated electrons, Phys. Rev. Lett. **119**, 056402 (2017).

[57] H. Eßl, M. Reitner, G. Sangiovanni, and A. Toschi, General Shiba mapping for on-site four-point correlation functions, Phys. Rev. Res. **6**, 033061 (2024).

[58] M. Reitner, L. Crippa, D. R. Fus, J. C. Budich, A. Toschi, and G. Sangiovanni, Protection of correlation-induced phase instabilities by exceptional susceptibilities, Phys. Rev. Res. **6**, L022031 (2024).

[59] M. Reitner, L. D. Re, M. Capone, and A. Toschi, Non-perturbative feats in the physics of correlated antiferromagnets, arXiv:2411.13417.

[60] H. Eßl, M. Reitner, E. Kozik, and A. Toschi, How to stay on the physical branch in self-consistent many-electron approaches, arXiv:2502.01420.

[61] P. Pulay, Convergence acceleration of iterative sequences. The case of scf iteration, Chem. Phys. Lett. **73**, 393 (1980).

[62] H. U. R. Strand, A. Sabashvili, M. Granath, B. Hellsing, and S. Östlund, Dynamical mean field theory phase-space extension and critical properties of the finite temperature Mott transition, Phys. Rev. B **83**, 205136 (2011).

[63] C. Hille, F. B. Kugler, C. J. Eckhardt, Y.-Y. He, A. Kauch, C. Honerkamp, A. Toschi, and S. Andergassen, Quantitative functional renormalization group description of the two-dimensional Hubbard model, Phys. Rev. Res. **2**, 033372 (2020).

[64] N. Niggemann, B. Sbierski, and J. Reuther, Frustrated quantum spins at finite temperature: Pseudo-Majorana functional renormalization group approach, Phys. Rev. B **103**, 104431 (2021).

[65] F. Bippus, B. Schneider, and B. Sbierski, Pseudo-Majorana functional renormalization for frustrated XXZ spin-$\frac{1}{2}$ models with field or magnetization along the spin-Z direction at finite temperature, Phys. Rev. B **111**, 054420 (2025).

[66] D. Vilardi, C. Taranto, and W. Metzner, Antiferromagnetic and d-wave pairing correlations in the strongly interacting two-dimensional Hubbard model from the functional renormalization group, Phys. Rev. B **99**, 104501 (2019).

[67] F. B. Kugler, Seung-Sup B. Lee, and J. von Delft, Multipoint correlation functions: Spectral representation and numerical evaluation, Phys. Rev. X **11**, 041006 (2021).

[68] J.-M. Lihm, J. Halbinger, J. Shim, J. von Delft, F. B. Kugler, and Seung-Sup B. Lee, Symmetric improved estimators for multipoint vertex functions, Phys. Rev. B **109**, 125138 (2024).

[69] D. Kiese, N. Wentzell, I. Krivenko, O. Parcollet, K. Held, and F. Krien, Embedded multi-boson exchange: A step beyond quantum cluster theories, Phys. Rev. Res. **6**, 043159 (2024).

[70] F. Schrodi, P. M. Oppeneer, and A. Aperis, Full-bandwidth Eliashberg theory of superconductivity beyond Migdal's approximation, Phys. Rev. B **102**, 024503 (2020).

[71] W. Sano, T. Koretsune, T. Tadano, R. Akashi, and R. Arita, Effect of van Hove singularities on high-T_c superconductivity in H_3S, Phys. Rev. B **93**, 094525 (2016).

[72] A. Szabo and N. Ostlund, *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory*, Dover Books on Chemistry (Dover Publications, Mineola, New York, 1996).

[73] B. I. Dunlap, Robust and variational fitting, Phys. Chem. Chem. Phys. **2**, 2113 (2000).

[74] S. Rohshap, M. K. Ritter, H. Shinaoka, J. von Delft, M. Wallerberger, and A. K. Kauch, Data for "Two-particle calculations with quantics tensor trains—solving the parquet equations" (2025), https://doi.org/10.48436/jn6hd-40f98.

4.3 Publication 5: Computing and compressing local vertex functions in imaginary and real frequencies from the multipoint numerical renormalization group using quantics tensor cross interpolation

In this section, the following publication is reprinted:

P5 *Computing and compressing local vertex functions in imaginary and real frequencies from the multipoint numerical renormalization group using quantics tensor cross interpolation*,
Markus Frankenbach, Marc K. Ritter, Mathias Pelz, Nepomuk Ritz, Jan von Delft, and Anxiang Ge,
to be submitted to Physical Review Research,
doi:10.48550/arXiv.2506.13359.
Reprinted on pages 142–161.

Computing and compressing local vertex functions in imaginary and real frequencies from the multipoint numerical renormalization group using quantics tensor cross interpolation

Markus Frankenbach, Marc K. Ritter, Mathias Pelz, Nepomuk Ritz, Jan von Delft, and Anxiang Ge
Arnold Sommerfeld Center for Theoretical Physics, Center for NanoScience, and Munich Center for Quantum Science and Technology, Ludwig-Maximilians-Universität München, 80333 Munich, Germany
(Dated: June 17, 2025)

The multipoint numerical renormalization group (mpNRG) is a powerful impurity solver that provides accurate spectral data useful for computing local, dynamic correlation functions in imaginary or real frequencies non-perturbatively across a wide range of interactions and temperatures. It gives access to a local, non-perturbative four-point vertex in imaginary and real frequencies, which can be used as input for subsequent computations such as diagrammatic extensions of dynamical mean–field theory. However, computing and manipulating the real-frequency four-point vertex on large, dense grids quickly becomes numerically challenging when the density and/or the extent of the frequency grid is increased. In this paper, we compute four-point vertices in a strongly compressed quantics tensor train format using quantics tensor cross interpolation, starting from discrete partial spectral functions obtained from mpNRG. This enables evaluations of the vertex on frequency grids with resolutions far beyond the reach of previous implementations. We benchmark this approach on the four-point vertex of the single-impurity Anderson model across a wide range of physical parameters, both in its full form and its asymptotic decomposition. For imaginary frequencies, the full vertex can be represented to an accuracy on the order of $2 \cdot 10^{-3}$ with maximum bond dimensions not exceeding 120. The more complex full real-frequency vertex requires maximum bond dimensions not exceeding 170 for an accuracy of $\lesssim 2\%$. Our work marks another step toward tensor-train-based diagrammatic calculations for correlated electronic lattice models starting from a local, non-perturbative mpNRG vertex.

I. INTRODUCTION

In the study of strongly correlated systems, correlations at the two-particle level play a key role. A powerful framework for computing two-particle correlation functions is given by quantum field theory approaches such as the functional renormalization group (fRG) [1] or (closely related [2–4]) the parquet equations [5]. While these methods formally provide exact and unbiased equations at the four-point level, solving them in practice requires some approximations. A common choice is the perturbative parquet approximation, which limits the applicability of these methods to weak interactions. In order to apply these diagrammatic methods to correlated electronic lattice systems in the physically relevant strong interaction regime, it has been proposed to combine them with dynamical mean–field theory (DMFT) [6]. DMFT approximates the self-energy to be local, i.e., momentum-independent, thereby neglecting spatial correlations but capturing local correlations non-perturbatively [7]. In the form of DMF^{2}RG [8] or the dynamical vertex approximation (DΓA) [9, 10], the fRG or the parquet equations can, in principle, be used to self-consistently add non-local correlations on the two-particle level on top of the local DMFT result.

However, such calculations entail two numerical challenges: the solution of the impurity model arising in the self-consistent DMFT loop and, subsequently, solving the fRG or parquet equations for frequency- *and* momentum-dependent vertices. The present work is concerned with the *interface* between these two steps, i.e., the conversion of local four-point spectral functions obtained from an impurity solver to a four-point (4p) vertex. An impurity solver that yields such 4p spectral functions is the multipoint numerical renormalization group (mpNRG)[11, 12]. This extension of the numerical renormalization group (NRG) [13, 14] is capable of computing both imaginary and real-frequency local correlation functions up to the four-point level in the form required for a subsequent diagrammatic extension of DMFT [15, 16]. Just as NRG, which has been the gold standard for solving impurity problems on the two-point level for decades [17, 18], mpNRG can be applied to a wide range of parameters, including large interactions and low temperatures. A central ingredient to mpNRG are spectral representations of time-ordered correlation functions in the frequency domain [11]. These represent correlators as convolutions of formalism-dependent but system-independent kernels with formalism-independent but system-dependent partial spectral functions (PSFs). While the former are known analytically, the latter are obtained from their respective Lehmann representations, using the eigenenergies and (discarded) eigenstates obtained from mpNRG. The local 4p vertex can be computed using the symmetric improved estimator (sIE) technique [15], which avoids the numerically unstable amputation of two-point (2p) Green's functions.

An appealing feature of spectral representations is that the same set of PSFs can be used to obtain imaginary- and real-frequency correlation functions. However, even

when energy conservation is exploited, the 4p vertex is a huge, three-dimensional object. Hence its computation from PSFs on a large, dense grid quickly becomes challenging or even unfeasible due to its huge memory footprint. Furthermore, performing calculations with such vertices as required in fRG or parquet calculations poses a major challenge [19–21].

It is thus highly desirable to represent 4p vertices in a *compressed format* that reduces the computational cost of the operations occurring in diagrammatic calculations. A promising candidate for compression is the quantics tensor train (QTT) representation [22, 23] of multivariate functions, which has recently proven useful in various areas of physics [24–29]. Its first application in the context of many-body theory was in Ref. 25, a study which demonstrated the compressibility of correlation functions and used QTT-based algorithms to solve the Schwinger-Dyson and Bethe–Salpeter equations. Furthermore, the QTT representation has been employed successfully in imaginary-frequency parquet calculations for the Hubbard atom and the single-impurity Anderson model (SIAM), using the parquet approximation [28].

These recent developments and the need for efficient representations of 4p vertices motivate this work: We use mpNRG to compute the local vertex of the SIAM as a function of real and imaginary frequencies in QTT format and investigate its compressibility across a broad range of physical parameters. The reason for studying the SIAM is its natural appearance in a DMFT treatment of the Hubbard model and the fact that it can be solved accurately using (mp)NRG. To compute the vertex of the SIAM, we employ the quantics tensor cross interpolation (QTCI) algorithm [30–33], which iteratively constructs a QTT by sparse sampling of the target function. This sampling-based interpolation enables evaluation of the mpNRG vertex on grids much larger and much denser than those accessible with the previous state-of-the-art [15]. For appropriate error tolerances, the maximum bond dimensions (ranks) of the resulting QTTs are within a range where diagrammatic calculations, such as those presented in Ref. 28, should be feasible, even for real frequencies.

This paper is organized as follows: In Sec. II, we recapitulate how the 4p vertex of the SIAM can be obtained in imaginary or real frequencies from PSFs. Additionally, we briefly explain key features of the QTCI algorithm and how it is employed in this work. In Sec. III, we show that imaginary- and, in particular, real-frequency vertices are representable by low-rank QTTs within a reasonable error margin. In the final Sec. IV, we provide an outlook on how the results of this work may be used to perform diagrammatic calculations for lattice models in QTT format.

II. METHODS

This section explains how to compute 4p vertices in QTT format. This is achieved in two steps: First, we convolve PSFs with formalism-dependent frequency kernels to obtain correlation functions (cf. Ref. 11). In a second step, the sIE scheme is employed [15] to extract the 4p vertex from various correlators and self-energies in a numerically stable fashion. The vertex is computed both in its asymptotic decomposition [34], which the sIE naturally yields, and in its 'full' form.

A. Partial spectral functions

The input to our calculations is given by the PSFs

$$\mathcal{S}[\boldsymbol{\mathcal{O}}](\boldsymbol{\omega}) = \int \frac{\mathrm{d}^\ell t}{(2\pi)^\ell}\, \mathrm{e}^{\mathrm{i}\boldsymbol{\omega}\cdot\boldsymbol{t}} \Big\langle \prod_{i=1}^{\ell} \mathcal{O}^i(t_i) \Big\rangle, \tag{1}$$

depending on a tuple $\boldsymbol{\mathcal{O}} = (\mathcal{O}^1, \ldots, \mathcal{O}^\ell)$ of operators in the Heisenberg picture and ℓ frequency arguments $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_\ell)$. By $\langle \mathcal{O} \rangle = \mathrm{Tr}\big[\mathrm{e}^{-\beta H}\mathcal{O}\big]/Z$, we denote the thermal expectation value, with the partition function $Z = \mathrm{Tr}[e^{-\beta H}]$ at inverse temperature $\beta = 1/T$. Time translation invariance implies:

$$\mathcal{S}[\boldsymbol{\mathcal{O}}](\boldsymbol{\omega}) = \delta(\omega_{1\ldots\ell})\, S[\boldsymbol{\mathcal{O}}](\boldsymbol{\omega}), \tag{2}$$

with the shorthand $\omega_{1\ldots\ell} = \sum_{i=1}^{\ell} \omega_i$, thus making $S[\boldsymbol{\mathcal{O}}]$ a function of $\ell - 1$ independent frequencies. In this work, we are primarily interested in the case $\ell = 4$, i.e., three-dimensional PSFs.

The PSFs carry the formalism-independent information that is specific to the model itself. The *same* set of PSFs can thus be used to compute Matsubara and Keldysh correlators by convolution with formalism-specific kernels. In this work, PSFs were computed using mpNRG as described in Ref. 12. This yields PSFs on a discrete, $(\ell-1)$-dimensional logarithmic energy grid for all relevant operator tuples $\boldsymbol{\mathcal{O}}$. This yields the representation

$$S[\boldsymbol{\mathcal{O}}](\boldsymbol{\omega}) = \sum_{\boldsymbol{\epsilon}} S[\boldsymbol{\mathcal{O}}](\boldsymbol{\epsilon})\, \delta(\boldsymbol{\omega} - \boldsymbol{\epsilon}), \tag{3}$$

where the peak weights $S[\boldsymbol{\mathcal{O}}](\boldsymbol{\epsilon})$ and energies $\boldsymbol{\epsilon}$ are obtained as output of the mpNRG computation. The energies $\boldsymbol{\epsilon}$ are binned into a Cartesian product of logarithmic grids.

B. Matsubara formalism

A Matsubara correlator $G(\mathrm{i}\boldsymbol{\omega})$ depending on ℓ operators $(\mathcal{O}^1, \ldots, \mathcal{O}^\ell)$ can be expressed via $\ell!$ PSFs $S[\boldsymbol{\mathcal{O}}_p]$, as was shown in Sec. II.C of Ref. 11:

$$G(\mathrm{i}\boldsymbol{\omega}) = \sum_p G_p(\mathrm{i}\boldsymbol{\omega}_p) = \sum_{p,\boldsymbol{\epsilon}} \boldsymbol{\zeta}^p K(\mathrm{i}\boldsymbol{\omega}_p - \boldsymbol{\epsilon}_p) S[\boldsymbol{\mathcal{O}}_p](\boldsymbol{\epsilon}_p). \tag{4}$$

The frequencies $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_\ell)$ are restricted to discrete fermionic or bosonic grids, depending on the type of the respective operator $\mathcal{O}^i$. Similarly to Eq. (2), frequency conservation $\omega_{1\ldots\ell} = 0$ is understood. The sum $\sum_p$ is over all permutations of ℓ elements, permuting frequency arguments and operators accordingly. Using the shorthand $\bar{i} = p(i)$, we can then write

$$\boldsymbol{\omega}_p = \left(\omega_{p(1)}, \ldots, \omega_{p(\ell)}\right) = \left(\omega_{\bar{1}}, \ldots, \omega_{\bar{\ell}}\right), \tag{5}$$

$$\boldsymbol{\mathcal{O}}_p = \left(\mathcal{O}_{p(1)}, \ldots, \mathcal{O}_{p(\ell)}\right) = \left(\mathcal{O}_{\bar{1}}, \ldots, \mathcal{O}_{\bar{\ell}}\right). \tag{6}$$

Depending on whether p transposes an even or odd number of fermionic operators, a sign factor $\boldsymbol{\zeta}^p = \pm 1$ is required in Eq. (4). The summands $G_p(\mathrm{i}\boldsymbol{\omega}_p)$ are termed *partial correlators*. Most importantly, Eq. (4) also introduces the Matsubara frequency kernel K, which reads

$$K(\boldsymbol{\Omega}_p) = \begin{cases} \prod_{i=1}^{\ell-1} \Omega_{\bar{1}\ldots\bar{i}}^{-1} & \text{if } \prod_{i=1}^{\ell-1} \Omega_{\bar{1}\ldots\bar{i}} \neq 0, \\ -\frac{1}{2}\left[\beta + \sum_{\substack{i=1 \\ i\neq j}}^{\ell-1} \Omega_{\bar{1}\ldots\bar{i}}^{-1}\right] \prod_{\substack{i=1 \\ i\neq j}}^{\ell-1} \Omega_{\bar{1}\ldots\bar{i}}^{-1} & \text{if } \exists j : \Omega_{\bar{1}\ldots\bar{j}} = 0, \end{cases} \tag{7}$$

where $\Omega_j = \mathrm{i}\omega_j - \epsilon_j$ (cf. Eq. (4)) and $\Omega_{\bar{1}\ldots\bar{i}} = \mathrm{i}\omega_{\bar{1}\ldots\bar{i}} - \epsilon_{\bar{1}\ldots\bar{i}}$ with $\mathrm{i}\omega_{\bar{1}\ldots\bar{i}} = \sum_{j=1}^{i} \mathrm{i}\omega_{\bar{j}}$ and $\epsilon_{\bar{1}\ldots\bar{i}} = \sum_{j=1}^{i} \epsilon_{\bar{j}}$. The definition (7) assumes that at most one of the partial sums $\Omega_{\bar{1}\ldots\bar{j}}$ vanishes, which is the case if there is at most one bosonic Matsubara frequency (this is always true in the present work). The first case in Eq. (7) is called regular kernel, the second case anomalous kernel. In the most important situation, $\ell = 4$ and $\prod_{i=1}^{\ell-1} \Omega_{\bar{1}\ldots\bar{i}} \neq 0$, the spectral representation of a partial correlator for permutation p is simply given by

$$G_p(\mathrm{i}\boldsymbol{\omega}_p) = \sum_{\epsilon_{\bar{1}}, \epsilon_{\bar{1}\bar{2}}, \epsilon_{\bar{1}\bar{2}\bar{3}}} S[\boldsymbol{\mathcal{O}}_p](\epsilon_{\bar{1}}, \epsilon_{\bar{1}\bar{2}}, \epsilon_{\bar{1}\bar{2}\bar{3}}) \prod_{i=1}^{3} (\mathrm{i}\omega_{\bar{1}\ldots\bar{i}} - \epsilon_{\bar{1}\ldots\bar{i}})^{-1}. \tag{8}$$

Eq. (8) assumes that the PSFs $S[\boldsymbol{\mathcal{O}}_p]$ are parametrized in partially summed energies $(\epsilon_{\bar{1}}, \epsilon_{\bar{1}\bar{2}}, \epsilon_{\bar{1}\bar{2}\bar{3}})$, which is always the case for our data (see Ref. 12).

C. Keldysh formalism

We now turn to the relation between PSFs and correlators in the Keldysh formalism [35–37]. For details and derivations, see Sec. II.D of Ref. 11. Keldysh ℓ-point correlators $G^{\boldsymbol{k}}(\boldsymbol{\omega})$ carry a Keldysh index $\boldsymbol{k} = (k_1, k_2, \ldots k_\ell)$ with $k_i \in \{1, 2\}$. Their spectral representation is analogous to Eq. (4):

$$G^{\boldsymbol{k}}(\boldsymbol{\omega}) = \frac{2}{2^{\ell/2}} \sum_p G_p^{\boldsymbol{k}}(\boldsymbol{\omega}_p), \tag{9a}$$

$$G_p^{\boldsymbol{k}}(\boldsymbol{\omega}_p) = \sum_{\boldsymbol{\epsilon}} \boldsymbol{\zeta}^p K_b^{\boldsymbol{k}_p}(\boldsymbol{\omega}_p, \boldsymbol{\epsilon}_p) S[\boldsymbol{\mathcal{O}}_p](\boldsymbol{\epsilon}_p). \tag{9b}$$

It involves ℓ real frequencies $\boldsymbol{\omega}$ that satisfy $\omega_{1\ldots\ell} = 0$. The broadened Keldysh frequency kernel $K_b^{\boldsymbol{k}_p}$ is a linear combination of the broadened, fully retarded kernels $K_b^{[\lambda]}$:

$$K_b^{\boldsymbol{k}_p}(\boldsymbol{\omega}_p, \boldsymbol{\epsilon}_p) = \sum_{\substack{\lambda=1 \\ k_{\bar{\lambda}} \text{ even}}}^{\ell} (-1)^{\lambda - 1 + k_{\bar{1}\ldots\overline{\lambda-1}}} \cdot K_b^{[\lambda]}(\boldsymbol{\omega}_p, \boldsymbol{\epsilon}_p), \tag{10a}$$

$$K_b^{[\lambda]}(\boldsymbol{\omega}_p, \boldsymbol{\epsilon}_p) = \prod_{j=1}^{\ell-1} \lim_{\gamma_0 \to 0^+} \int_{\mathbb{R}} \mathrm{d}\omega'_{\bar{1}\ldots\bar{j}} \frac{\delta_b(\omega'_{\bar{1}\ldots\bar{j}}, \epsilon_{\bar{1}\ldots\bar{j}})}{\omega_{\bar{1}\ldots\bar{j}} - \omega'_{\bar{1}\ldots\bar{j}} + \mathrm{i}\gamma^{\lambda}_{0,j}}, \tag{10b}$$

where $\delta_b(\omega'_{\bar{1}\ldots\bar{j}}, \epsilon_{\bar{1}\ldots\bar{j}})$ is a broadened version of the Dirac-δ function appearing in Eq. (3). This broadening ensures a smooth structure of the kernel, free from unphysical poles or δ-peaks. The imaginary shifts $\mathrm{i}\gamma^{\lambda}_{0,j}$ in Eq. (10b) are defined as

$$\mathrm{i}\gamma^{\lambda}_{0,j} = \begin{cases} \mathrm{i}\gamma_0 \cdot (\ell - j) & j \geq \lambda, \\ -\mathrm{i}\gamma_0 \cdot j & j < \lambda. \end{cases} \tag{10c}$$

While the factors $\ell - j$ and j in Eq. (10c) can be disregarded in the limit $\gamma_0 \to 0^+$, they remain relevant for the linear broadening. Details on the broadening procedure can be found in App. A and Ref. 12, Sec. VI.

D. Symmetric improved estimators: from correlators to the vertex

In principle, the one-particle irreducible 4p vertex can be obtained simply by amputating the four external 2-point (2p) propagators ("legs") of the connected impurity Green's function $G_{\mathrm{con}}[d_{\sigma_1} d^\dagger_{\sigma_2} d_{\sigma_3} d^\dagger_{\sigma_4}]$ (cf. Sec. III A). In practice, however, this leads to pronounced numerical artifacts, especially at asymptotically large frequencies, where both functions decay to zero. A numerically stable scheme that avoids direct amputation is the symmetric improved estimator (sIE) technique introduced in Ref. 15. In addition, this method yields the vertex in its asymptotic decomposition [34]. This decomposition separates the contributions that decay only in one or two frequency directions from the genuinely three-dimensional core vertex Γ_{core}, which asymptotically decays in all three frequencies,

$$\begin{aligned} \Gamma(\omega, \nu, \nu') = {} & \Gamma_{\mathrm{core}}(\omega, \nu, \nu') \\ & + \sum_{r=a,p,t} \left[\mathcal{K}_2^r(\omega_r, \nu_r) + \mathcal{K}_{2'}^r(\omega_r, \nu'_r) + \mathcal{K}_1^r(\omega_r)\right] \\ & + \Gamma_0. \end{aligned} \tag{11}$$

The functions $\mathcal{K}_1^r$, $\mathcal{K}_2^r$ and $\mathcal{K}_{2'}^r$ (not to be confused with the kernels K defined above) only depend on one or two frequencies if parametrized in their native channel r. They are one- and two-dimensional contributions to the two-particle reducible vertex in channel r. This channel can

be the antiparallel (a), parallel (p), or the transverse (t) channel. These are also known as the particle-hole, particle-hole crossed and particle-particle channels, respectively [15]. By Γ_0 we denote the frequency-independent bare vertex. Note that the sIE method does not provide a decomposition of Γ_{core} into two-particle reducible contributions $\mathcal{K}_3^r$ and a two-particle irreducible term.

In this work, we consider the single-impurity Anderson model with interaction $H_{\text{int}} = U d_\uparrow^\dagger d_\uparrow d_\downarrow^\dagger d_\downarrow$, where $d_\sigma^\dagger$ creates an electron with spin σ on the impurity, see Sec. III A for details. We denote the self-energy of the impurity Green's function $G[d_\sigma, d_{\sigma'}^\dagger](\nu)$ as $\Sigma^{\sigma\sigma'}(\nu)$. In the absence of a magnetic field, it satisfies $\Sigma^{\sigma\sigma'}(\omega) = \delta^{\sigma\sigma'}\Sigma(\omega)$. Following Ref. 15, Γ_{core} can be obtained as

$$\Gamma_{\text{core}}(\boldsymbol{\omega}) = \sum_{a_i \in \{d,q\}} Y_{a_1}(\omega_1) Y_{a_3}(\omega_3)\, G_{\text{con}}[a_1, a_2^\dagger, a_3, a_4^\dagger](\boldsymbol{\omega}) \cdot Y_{a_2}(\omega_2) Y_{a_4}(\omega_4), \tag{12}$$

where we introduced an auxiliary operator $q = [d, H_{\text{int}}]$, and

$$Y_{a_i}(\omega_i) = \begin{cases} -\Sigma\left((-1)^{i-1}\omega_i\right) & a_i = d, \\ X = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right) & a_i = q, \end{cases} \tag{13}$$

is a 2×2 matrix acting on the ith Keldysh index of G_{con}. In the Matsubara formalism, X is replaced by scalar unity. In practice, the main workload in computing Γ_{core} at a given frequency $\boldsymbol{\omega}$ is the evaluation of all $2^4 = 16$ connected correlators, each comprised of $4! = 24$ partial correlators. The quantities $\mathcal{K}_1^r$ and $\mathcal{K}_{2^{(\prime)}}^r$ can be computed using an analogous approach presented in App. B.

E. Quantics Tensor Cross Interpolation

Having summarized the evaluation of the 4-point vertex in its asymptotic decomposition, we next discuss the quantics tensor cross interpolation (QTCI) method [30–33], which we used to obtain vertex functions in the form of QTTs. Recently, it has been shown in the Matsubara formalism that this representation is well-suited for efficient diagrammatic calculations [28]. We discuss only the basics of QTCI here. For a detailed introduction, we refer to Ref. 38.

Let us begin with the quantics representation [22, 23]. Consider a one-dimensional function $f(\omega)$ defined on a discrete, equidistant grid $\{\omega_0, \ldots, \omega_{2^R-1}\}$ consisting of 2^R points with $\omega_m \in \mathbb{R}$. The grid index m of a point ω_m can be written in binary representation

$$m = \sum_{\ell=1}^{R} 2^{R-\ell}\sigma_\ell,\ \sigma_\ell \in \{0,1\}, \tag{14}$$

so that m can be identified with the R-tuple $(\sigma_1, \ldots, \sigma_R)$. Hence, the mapping $m \mapsto f(\omega_m)$ can be viewed as an R-leg tensor $F_{\sigma_1\cdots\sigma_R} = f(\omega_{m(\{\sigma_\ell\})})$. This so-called quantics encoding can be generalized to higher-dimensional functions, in particular to functions $f(\omega, \nu, \nu')$ of three frequency arguments. The frequencies lie on a Cartesian product of 1D grids, each of size 2^R. We use a binary encoding

$$(\omega_i, \nu_j, \nu'_k) = \big((\sigma_{11}, ..., \sigma_{1R}), (\sigma_{21}, ..., \sigma_{2R}), (\sigma_{31}, ..., \sigma_{3R})\big), \tag{15}$$

with the binary variables $\sigma_{n\ell}$ labelled by $n = 1, 2, 3$ for ω, ν, ν'. The function f can then be represented by a tensor with $3R$ indices:

$$F_{\boldsymbol{\sigma}} = F_{\sigma_{11}\sigma_{21}\sigma_{31}\cdots\sigma_{1R}\sigma_{2R}\sigma_{3R}} = f(\omega_i, \nu_j, \nu'_k). \tag{16}$$

Importantly, note that the tensor indices in Eq. (16) have been *interleaved*, such that the indices corresponding to the same length scale $2^{R-\ell}$ in different variables $\sigma_{1\ell}, \sigma_{2\ell}, \sigma_{3\ell}$, are adjacent. Alternatively, triples $(\sigma_{\ell 1}, \sigma_{\ell 2}, \sigma_{\ell 3})$ of legs can be fused to single legs $\tilde{\sigma}_\ell = \sum_{n=1}^{3} 2^{n-1}\sigma_{\ell n}$, which yields the *fused* representation of f as an R-leg tensor:

$$\widetilde{F}_{\tilde{\boldsymbol{\sigma}}} = \widetilde{F}_{\tilde{\sigma}_1\cdots\tilde{\sigma}_R} = f(\omega_i, \nu_j, \nu'_k). \tag{17}$$

The second ingredient of QTCI is the tensor cross interpolation (TCI) algorithm [30–32, 38], which approximates tensors $F_{\boldsymbol{\sigma}} = F_{\sigma_1\cdots\sigma_L}$ (with $L = R, 2R$ or $3R$ for one-, two- or three-dimensional functions, respectively) using tensor trains constructed from a sampled subset of all tensor elements. If a low-rank factorization of the tensor exists, the number of samples taken is much smaller than the number of elements of the full tensor. This way, the cost of generating all tensor elements, exponentially large in R, can be avoided.

More precisely, the TCI algorithm seeks to find a tensor train (TT)

$$F^{\text{QTCI}}_{\sigma_1\ldots\sigma_L} = \sum_{\alpha_1\ldots\alpha_{L-1}} [M_1^{\sigma_1}]_{1\alpha_1} [M_2^{\sigma_2}]_{\alpha_1\alpha_2} \cdots [M_L^{\sigma_L}]_{\alpha_{L-1}1} \tag{18}$$

that minimizes the elementwise error

$$\varepsilon_{\boldsymbol{\sigma}}[F] = \frac{|F^{\text{QTCI}}_{\boldsymbol{\sigma}} - F_{\boldsymbol{\sigma}}|}{\max_{\boldsymbol{\sigma}'} |F_{\boldsymbol{\sigma}'}|}\,. \tag{19}$$

Here, the α_ℓ are virtual bond indices with ℓ-dependent bond dimensions, $\alpha_\ell = 1, ..., \chi_\ell$. The maximum bond dimension, $\chi = \max \chi_\ell$, is called the *rank* of $F^{\text{QTCI}}_{\boldsymbol{\sigma}}$. The maximum in Eq. (19) is estimated using all sampled entries of $F_{\boldsymbol{\sigma}}$. The TCI algorithm optimizes the tensors $[M_\ell^{\sigma_\ell}]_{\alpha_{\ell-1}\alpha_\ell}$ iteratively, progressively sampling $F_{\boldsymbol{\sigma}}$, until no $\boldsymbol{\sigma}$ is found where the error $\varepsilon_{\boldsymbol{\sigma}}$ exceeds a given tolerance τ. During this process, the bond dimensions χ_ℓ are increased dynamically to improve the accuracy of the tensor train representation. Finding a tensor train representation of $F^{\text{QTCI}}_{\boldsymbol{\sigma}}$ with TCI has a computational cost of $\mathcal{O}\big(R\chi^3\big)$.

In conjunction with the quantics representation, TCI can be employed to approximate not only tensors, but also

functions defined on discrete grids by tensor trains. Once the bond dimensions are saturated, i.e., no longer increase with R, the computational cost of QTCI scales linearly in R. This translates to an exponential resolution of the target function at linear cost. Of course, the function is only approximated within an error margin given by Eq. (19). The next section details how we used QTCI to compress 4p vertex functions on exponentially large grids.

F. Implementation details

To obtain the core and full vertices in the Matsubara and Keldysh formalisms as QTTs, we apply QTCI to functions that evaluate $\Gamma_{\text{core}}(\boldsymbol{\omega})$ and $\Gamma(\boldsymbol{\omega})$ on individual frequency points $\boldsymbol{\omega}$ to be specified on demand by the TCI algorithm. The frequencies $\boldsymbol{\omega}$ reside on an equidistant grid of 2^{3R} points. For Matsubara grids, the grid spacing is set by the temperature, and the extent of the grid can be increased exponentially by increasing R. For Keldysh vertices, which are functions of continuous frequencies, one may exponentially increase either the density of grid points, or the extent of the grid, or both, by increasing R. It is important not to precompute the vertices on a dense grid, as this precomputation step would incur costs scaling as $\mathcal{O}(2^{3R})$. By avoiding precomputation, R can be increased to yield grid sizes and/or grid densities beyond those attainable by conventional means. The TCI algorithm samples a *sparse* set of $\mathcal{O}(\chi^2 R)$ points $\boldsymbol{\omega}$, which is much smaller than 2^{3R}, the total number of grid points for the 4p vertex functions considered in this work. For this application, function evaluation during sampling is the dominant cost as opposed to the $\mathcal{O}(\chi^3 R)$ cost of computing prrLU factorizations (see [38, Sec. 3.3]). More specifically, the computational effort is dominated by the evaluation of partial 4p correlators. These enter Γ_{core} and Γ via the full correlators appearing in Eq. (12). In this section, we discuss how our code evaluates partial correlators in an efficient way. Readers only interested in our results on the compressibility of Matsubara and Keldysh vertices can move on to Sec. III.

1. *Matsubara vertices*

In the Matsubara case, evaluating the regular part of the 4p correlator, Eq. (8), constitutes the majority of computational cost. Using the shorthand $k_{\omega_{\bar{1}\dots\bar{i}}\epsilon_{\bar{1}\dots\bar{i}}} = \mathrm{i}\Omega_{\bar{1}\dots\bar{i}} = (\mathrm{i}\omega_{\bar{1}\dots\bar{i}} - \epsilon_{\bar{1}\dots\bar{i}})^{-1}$, we can rewrite Eq. (8) as

$$G_p(\mathrm{i}\boldsymbol{\omega}_p) = \sum_{\epsilon_{\bar{1}},\epsilon_{\bar{1}\bar{2}},\epsilon_{\bar{1}\bar{2}\bar{3}}} k_{\omega_{\bar{1}}\epsilon_{\bar{1}}} k_{\omega_{\bar{1}\bar{2}}\epsilon_{\bar{1}\bar{2}}} k_{\omega_{\bar{1}\bar{2}\bar{3}}\epsilon_{\bar{1}\bar{2}\bar{3}}} S[\boldsymbol{\mathcal{O}}_p](\epsilon_{\bar{1}},\epsilon_{\bar{1}\bar{2}},\epsilon_{\bar{1}\bar{2}\bar{3}}). \tag{20}$$

Since $(\omega_{\bar{1}},\omega_{\bar{1}\bar{2}},\omega_{\bar{1}\bar{2}\bar{3}})$ and $(\epsilon_{\bar{1}},\epsilon_{\bar{1}\bar{2}},\epsilon_{\bar{1}\bar{2}\bar{3}})$ live on finite frequency grids, the kernels $k_{\omega_{\bar{1}\dots\bar{i}}\epsilon_{\bar{1}\dots\bar{i}}}$ can be viewed as matrices. A typical grid size for the spectral function peaks

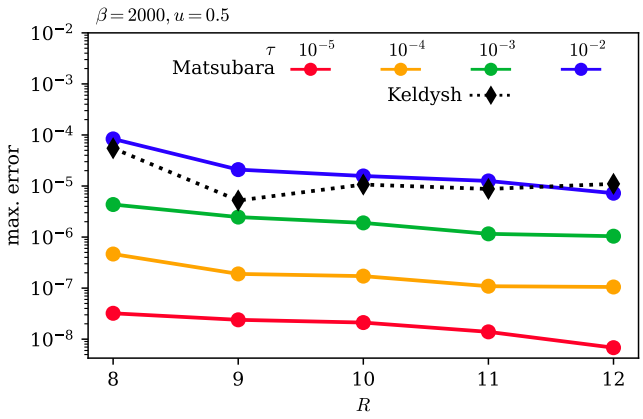


FIG. 1. Maximum error of Matsubara (circles) and Keldysh (diamonds) core vertex evaluations with SVD truncations as described in the main text. The error is measured relative to the maximum of the respective vertex function. By τ we denote the target TCI tolerance, and use a cutoff of $S_{\text{cut}} = 10^{-2}\tau$ for Matsubara. For Keldysh, we choose an SVD cutoff of 10^{-6} times the largest singular value, and find that this yields results that are sufficiently accurate for a tolerance of $\tau = 10^{-3}$. We show maximum errors over 64000 sampling points for the Matsubara core vertex and $2\cdot 10^6$ sampling points for the more complicated Keldysh core vertex. All errors are well below the respective target tolerance τ.

$(\epsilon_{\bar{1}},\epsilon_{\bar{1}\bar{2}},\epsilon_{\bar{1}\bar{2}\bar{3}})$ is $70\times 70\times 70$, while the largest grids for Matsubara frequencies $(\omega_{\bar{1}},\omega_{\bar{1}\bar{2}},\omega_{\bar{1}\bar{2}\bar{3}})$ used in this work have $2^{12} = 4096$ points in each dimension. Thus $k_{\omega_{\bar{1}\dots\bar{i}}\epsilon_{\bar{1}\dots\bar{i}}}$ can be precomputed and stored for all relevant grid sizes.

We implemented two methods to speed up the threefold contractions in Eq. (20): (i) compressing the kernels and (ii) performing one kernel contraction as a preprocessing step.

(i) Compressing the kernels is the more general of the two methods, in that it has a smaller memory footprint ($< 1\,$GB per full correlator $G(\mathrm{i}\boldsymbol{\omega})$ for $R = 12, \tau = 10^{-3}$). The idea is to compress the kernels $k_{\omega\epsilon}$ by exploiting their low-rank structure [39, 40]: We SVD-decompose each $k_{\omega\epsilon}$ and discard singular values below a given cutoff S_{cut}, resulting in the approximation

$$k_{\omega\epsilon} \approx \sum_a U_{\omega a} S_a V^\dagger_{a\epsilon}. \tag{21}$$

We then contract the singular values $S_a \geq S_{\text{cut}}$ and the right-hand isometries $V^\dagger$ with the PSF by performing ϵ sums to obtain a smaller rank-3 tensor A, thus reducing the cost of the threefold summation in Eq. (20):

$$G_p(\mathrm{i}\boldsymbol{\omega}_p) = \sum_{a_1a_2a_3} A_{a_1a_2a_3} \prod_{i=1}^{3} U_{\omega_i a_i}, \tag{22a}$$

$$A_{a_1a_2a_3} = \sum_{\epsilon_{\bar{1}},\epsilon_{\bar{1}\bar{2}},\epsilon_{\bar{1}\bar{2}\bar{3}}} S[\boldsymbol{\mathcal{O}}_p](\epsilon_{\bar{1}},\epsilon_{\bar{1}\bar{2}},\epsilon_{\bar{1}\bar{2}\bar{3}}) \prod_{i=1}^{3} S_{a_i} V^\dagger_{a_i\epsilon_{\bar{1}\dots\bar{i}}}. \tag{22b}$$

The computations (21) and (22b) are performed during preprocessing prior to the QTCI run. Note that this treatment of the low-rank Matsubara kernels is closely related to the so-called intermediate representation (IR) of Matsubara Green's functions, see Refs. [41, 42]. The cutoff S_{cut} should be chosen as to introduce an error significantly below the TCI tolerance τ in all target quantities. While one can bound the error in Eq. (22a), e.g., using the Cauchy–Schwarz inequality, these estimates were found to be very conservative. We observed that setting $S_{\mathrm{cut}} = 10^{-2}\,\tau$ leads to errors more than two orders of magnitude below the TCI tolerance when evaluating correlators and vertices. This is shown in Fig. 1, where we plot the accuracy of Matsubara (and Keldysh) core vertex evaluations for different TCI tolerances τ and numbers of quantics bits R. In Matsubara, the accuracy improves with increasing R. This is because, for a fixed SVD cutoff, fewer singular values are discarded for larger R.

A further speedup can be achieved by realizing that, even though all singular values S_a in Eq. (21) are larger than S_{cut}, their products appearing in Eq. (22b) can become negligibly small. Ordering S_{a_i} by decreasing magnitude, we therefore discard all entries $A_{a_1a_2a_3}$ where $a_1 + a_2 + a_3$ is larger than some integer N:

$$G_p(\mathrm{i}\boldsymbol{\omega}_p) = \sum_{\sum_i a_i \le N} A_{a_1a_2a_3} \prod_{i=1}^{3} U_{\omega_i a_i}\,. \tag{23}$$

A similar truncation is useful when constructing IR 4pt Green's functions, see Ref. 43. To determine N for a prescribed tolerance τ, we estimate the contribution from terms with $\sum_i a_i > N$ via the Cauchy–Schwarz inequality:

$$\begin{aligned}
&\left| \sum_{\sum_i a_i > N} A_{a_1a_2a_3} \prod_{i=1}^{3} U_{\omega_i a_i} \right| \\
&\quad \le \sqrt{\sum_{\sum_i a_i > N} |A_{a_1a_2a_3}|^2} \prod_{i=1}^{3} \max_{\omega_i} \sqrt{\sum_{a_i} |U_{\omega_i a_i}|^2} \\
&\quad = \sqrt{\sum_{\sum_i a_i > N} |A_{a_1a_2a_3}|^2}\,.
\end{aligned} \tag{24}$$

The second factor in the second line of Eq. (24) is equal to one, since the U's are isometries. Hence, Eq. (24) provides a simple bound on the error in $G_p(\mathrm{i}\boldsymbol{\omega})$, which is independent of the frequency at hand. We choose N such that

$$\sqrt{\sum_{\sum_i a_i > N} |A_{a_1a_2a_3}|^2} \le \frac{\tau}{10} \max_{\boldsymbol{\omega}} |G(\mathrm{i}\boldsymbol{\omega})|, \tag{25}$$

which ensures an error one order of magnitude below the TCI tolerance in the full correlator G. While this error in principle occurs *per* partial correlator G_p, the criterion (25) was observed to yield sufficient accuracy. Overall, this first method gives a substantial speedup compared to directly performing the contractions in Eq. 20: It accelerates the evaluation of the full 4p impurity correlator $G[d_\uparrow, d_\uparrow^\dagger, d_\uparrow, d_\uparrow^\dagger]$ at $\beta = 2000$, $u = 0.5$ (cf. Sec. III A) at a single frequency in an $R = 12$ quantics grid using an SVD cutoff of $S_{\mathrm{cut}} = 10^{-5}$ by more than a factor 60. This observation simply reflects the strong compresssibilty of the Matsubara kernels.

(ii) A more straightforward way to speed up pointwise evaluations of partial correlators (20) is to precompute one of the three contractions before running QTCI. This yields an object depending on the variables $(\omega_{\bar{1}}, \epsilon_{\bar{1}\bar{2}}, \epsilon_{\bar{1}\bar{2}\bar{3}})$,

$$B_p(\omega_{\bar{1}}, \epsilon_{\bar{1}\bar{2}}, \epsilon_{\bar{1}\bar{2}\bar{3}}) = \sum_{\epsilon_{\bar{1}}} k_{\omega_{\bar{1}}\epsilon_{\bar{1}}} S[\boldsymbol{\mathcal{O}}_p](\epsilon_{\bar{1}}, \epsilon_{\bar{1}\bar{2}}, \epsilon_{\bar{1}\bar{2}\bar{3}}). \tag{26}$$

B_p gives access to G_p via

$$G_p(\mathrm{i}\boldsymbol{\omega}_p) = \sum_{\epsilon_{\bar{1}\bar{2}}\epsilon_{\bar{1}\bar{2}\bar{3}}} k_{\omega_{\bar{1}\bar{2}}\epsilon_{\bar{1}\bar{2}}} k_{\omega_{\bar{1}\bar{2}\bar{3}}\epsilon_{\bar{1}\bar{2}\bar{3}}} B_p(\omega_{\bar{1}}, \epsilon_{\bar{1}\bar{2}}, \epsilon_{\bar{1}\bar{2}\bar{3}}). \tag{27}$$

In this approach, we only have two kernel contractions in each evaluation G_p, but have to store the intemediates (26) for all partial correlators. Their size grows linearly in the grid frequency grid size, i.e., as 2^R where R is the number of quantics bits. For example, if $R = 12$ and the PSFs live on a $70\times70\times70$ logarithmic grid (which results from 2×6 decades of energy bins with 8 points per decade and discarding zeros in the PSFs), each full correlator consumes 12.9 GB of memory. On an $R = 12$ grid at $\beta = 2000$, $u = 0.5$, the precomputation also yields a speedup of about a factor 60. But in contrast to the compression of kernels (Eq. (21)), this speedup is independent of the TCI tolerance τ. Overall, method (ii) is recommended as long as its memory demands can be met, because it evaluates correlators in a numerically exact fashion.

2. *Keldyh vertices*

In the Keldysh formalism, evaluating partial correlators $G_p^{\boldsymbol{k}}$ (cf. Eq. (9b)) also comes down to threefold contractions of a 3-dimensional PSF with kernel matrices. This can be seen by rewriting the kernel $K_b^{[\lambda]}$ (10b) as a product of one-dimensional kernels evaluated at frequencies $\omega_{\bar{1}\ldots\bar{i}}$:

$$K_b^{[\lambda]}(\boldsymbol{\omega}_p, \boldsymbol{\epsilon}_p) = \prod_{i=1}^{3} k_b^{[\lambda,i]}(\omega_{\bar{1}\ldots\bar{i}}, \epsilon_{\bar{1}\ldots\bar{i}}), \tag{28a}$$

$$k_b^{[\lambda,i]}(\omega_{\bar{1}\ldots\bar{i}}, \epsilon_{\bar{1}\ldots\bar{i}}) = \lim_{\gamma_0\to0^+} \int_{\mathbb{R}} \mathrm{d}\omega'_{\bar{1}\ldots\bar{i}} \frac{\delta_b(\omega'_{\bar{1}\ldots\bar{i}}, \epsilon_{\bar{1}\ldots\bar{i}})}{\omega_{\bar{1}\ldots\bar{i}} - \omega'_{\bar{1}\ldots\bar{i}} + \mathrm{i}\gamma^{\lambda}_{0,i}}. \tag{28b}$$

Equation (9b) can then be written as:

$$G_p^{\boldsymbol{k}}(\boldsymbol{\omega}_p) = \sum_{\substack{\lambda=1 \\ k_{\overline{\lambda}}\text{even}}}^{4} (-1)^{\lambda-1+k_{\bar{1}\ldots\overline{\lambda-1}}} \cdot G_p^{[\lambda]}(\boldsymbol{\omega}_p), \tag{29a}$$

$$G_p^{[\lambda]}(\boldsymbol{\omega}_p) = \sum_{\epsilon_{\bar{1}},\epsilon_{\bar{1}\bar{2}},\epsilon_{\bar{1}\bar{2}\bar{3}}} S[\boldsymbol{\mathcal{O}}_p](\epsilon_{\bar{1}},\epsilon_{\bar{1}\bar{2}},\epsilon_{\bar{1}\bar{2}\bar{3}})\times \prod_{i=1}^{3} k_b^{[\lambda,i]}(\omega_{\bar{1}\ldots\bar{i}},\epsilon_{\bar{1}\ldots\bar{i}}). \quad (29b)$$

The ensuing contractions (29b) to be performed for $\lambda = 1,...,4$ are analogous to Eq. (20). However, interpolating the complex structure of the Keldysh vertex requires more evaluations compared to its Matsubara counterpart. At the same time, the memory cost of a precomputation analogous to Eq. (26) becomes prohibitive for large ($R \gtrsim 12$) frequency grids, since it must be applied to $G_p^{[\lambda]}$ for $\lambda = 1,...,4$ and for each partial correlator. For these reasons, the optimization of Eq. (29b) needs to go beyond the compression scheme for the Matsubara kernels from Eqs. (21) and (22b). To this end, we exploit the fact that the structure of the 1D kernels $k_b^{[\lambda,i]}(\omega_{\bar{1}\ldots\bar{i}},\epsilon_{\bar{1}\ldots\bar{i}})$ becomes simpler at large frequencies $\omega_{\bar{1}\ldots\bar{i}}$: We divide the $\omega_{\bar{1}\ldots\bar{i}}$ grid into n_L equally-sized intervals $I_1^i,\ldots,I_{n_L}^i$, with $n_L = 2^3$ as a default. Then, for each dimension i and each interval I_j^i, we SVD-decompose the restricted kernel

$$k_b^{[\lambda,i]}(\omega_{\bar{1}\ldots\bar{i}},\epsilon_{\bar{1}\ldots\bar{i}})\Big|_{\omega_{\bar{1}\ldots\bar{i}}\in I_j^i} \approx \sum_{a_i} U^{ij}_{\omega_{\bar{1}\ldots\bar{i}}a_i} S^{ij}_{a_i} V^{\dagger ij}_{a_i\epsilon_{\bar{1}\ldots\bar{i}}}, \quad (30)$$

discarding singular values that are at least 6 orders of magnitude smaller than the largest singular value. This strategy of partitioning the $\omega_{\bar{1}\ldots\bar{i}}$ grid prior to the SVD truncation allows us to discard more singular values in outer intervals, where the kernel is more compressible. Next, for each triple of intervals (I_k^1, I_l^2, I_m^3), we contract the corresponding singular values and right hand isometries into the PSF. While this entails precomputing n_L^3 3-leg tensors of the form

$$\left(A^{klm}\right)_{a_1a_2a_3} = \\ = \sum_{\epsilon_{\bar{1}},\epsilon_{\bar{1}\bar{2}},\epsilon_{\bar{1}\bar{2}\bar{3}}} (SV^\dagger)^{1k}_{a_1\epsilon_{\bar{1}}} (SV^\dagger)^{2l}_{a_2\epsilon_{\bar{1}\bar{2}}} (SV^\dagger)^{3m}_{a_3\epsilon_{\bar{1}\bar{2}\bar{3}}} S[\boldsymbol{\mathcal{O}}_p](\epsilon_{\bar{1}},\epsilon_{\bar{1}\bar{2}},\epsilon_{\bar{1}\bar{2}\bar{3}}), \quad (31)$$

it yields a substantial speedup in evaluations of $G^{\boldsymbol{k}}(\boldsymbol{\omega})$: For $\beta = 2000$, $u = 0.5$, $\omega_{\max} = 0.65$ (cf. Sec. III A) and $R = 12$ this scheme is about a factor 150 faster than a naive kernel contraction. This speedup refers to an average over $2 \cdot 10^5$ evaluations on random frequency points, since the compressibility of the kernels depends on the intervals (I_k^1, I_l^2, I_m^3) the frequencies $(\omega_{\bar{1}}, \omega_{\bar{1}\bar{2}}, \omega_{\bar{1}\bar{2}\bar{3}})$ belong to. Indeed, truncated isometries U^{ij} pertaining to the outermost intervals usually have about 5 times fewer rows than those of the inner intervals. That Keldysh core vertex evaluations using the above scheme are sufficiently accurate (i.e. to more than 10^{-3}, see Sec. III C) is verified in Fig. 1.

Having explained the optimization of vertex evaluations, we turn to the settings chosen in the QTCI routine. All of our code is written in Julia (versions 1.9.4 and 1.10.3), using the TCI package `TensorCrossInterpolation.jl`, the quantics utilities `QuanticsGrids.jl` as well as the QTCI package `QuanticsTCI.jl` of the tensor4all collaboration [38, 44]. The latter exposes the `quanticscrossinterpolate` routine, which is the entry point of the QTCI algorithm and offers various settings: We used the default `:backandforth` sweep strategy and the `:fullsearch` pivot search strategy. The increase in computational cost entailed by a full pivot search was accepted to ensure a reliable interpolation. For Matsubara objects, the interleaved representation was chosen to obtain maximum memory compression. In the Keldysh case, the interleaved representation exhibited convergence problems: After 80 sweeps (with ≤ 5 sweeps until convergence being common), a QTT with an error significantly exceeding the tolerance was obtained. Switching to the fused representation solved this problem. This is due to the fact that a 2-site update in a 3D fused representation corresponds to a 6-site update in the interleaved representation, which implies more extensive sampling of the target function. Another choice worth mentioning is that of initial pivots: For Keldysh vertices, it was sufficient to use the grid center as the only initial pivot. In Matsubara, the same choice occasionally lead to premature termination of the TCI algorithm, resulting in a QTT representation that was missing relevant features. We therefore chose 125 initial pivots forming a cube at Matsubara frequencies $(\omega_i, \nu_{j-1}, \nu'_{k-1})$ with $i,j,k \in \{-2,...,2\}$. On fermionic grids, the cube is thus centered around $\nu_{-1} = \nu'_{-1} = -\pi T$. This choice of initial pivots ensures that the sharp Matsubara vertex structure around the origin is properly sampled. Finally, since vertex evaluations are the bottleneck of our QTCI-compressions, a significant speedup can be achieved via multithreading. The samples evaluated during a two-site optimization step (see Ref. [38, Sec. 4.3]) are independent of one another, and can therefore be evaluated in parallel.

We tested our code for evaluating vertex functions with the sIE scheme against the prior Matlab implementation used in Ref. 15. We found numerically exact agreement with a normalized discrepancy $< 10^{-13}$ for the Matsubara quantities. In Keldysh the maximum discrepancy in the core vertex between our Julia code and the Matlab code of Ref. 15 is about $0.002 \cdot ||\Gamma_{\text{core}}||_\infty$ (with the supremum norm $||\cdot||_\infty$). This discrepancy can be attributed to small differences in the broadening implementation, mainly the interpolation of the broadened kernel from a logarithmic to a linear grid (cf. App. A). This discrepancy is one order of magnitude smaller than the error introduced by the arbitrariness inherent in the choice of broadening parameters. As an additional test, our code was used to generate Keldysh vertex data to check the fulfillment of exact diagrammatic relations of mpNRG data [16].

To conclude this section, Fig. 2 compares the singular value spectra of regular Matsubara kernels (Eq. (7)) and broadened, fully retarded Keldysh kernels (Eq. (28a)) at different temperatures. As expected, the singular values of both Matsubara and Keldysh kernels decay significantly faster at higher temperatures. In Keldysh, this is due

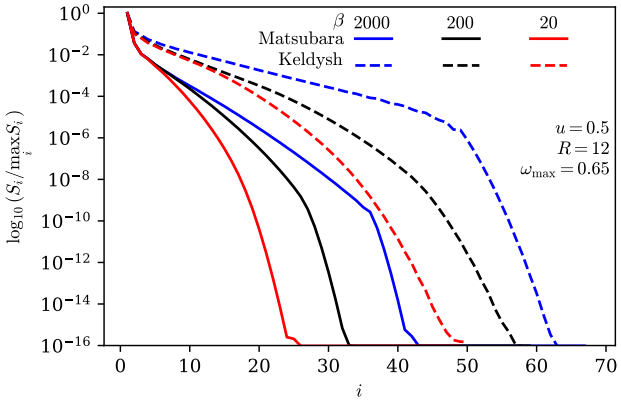


FIG. 2. Singular values S_i of regular Matsubara kernels (Eq. (7), solid line) and broadened, fully retarded Keldysh kernels (Eq. (28a), dashed line). We show kernels at inverse temperatures $\beta \in \{20, 200, 2000\}$ and interaction $u = 0.5$. The frequency grids are bosonic with 2^{12} points, with the Keldysh grid ranging from -0.65 to $\omega_{\max} = 0.65$.

U	Δ	u	T_K	$\beta_K = 1/T_K$
0.05	0.0318	0.5	$4.14 \cdot 10^{-2}$	24.2
0.05	0.0159	1.0	$9.58 \cdot 10^{-3}$	104
0.05	0.0106	1.5	$3.57 \cdot 10^{-3}$	280
0.05	0.00530	3.0	$3.36 \cdot 10^{-4}$	2980
0.05	0.00318	5.0	$2.06 \cdot 10^{-5}$	48400

TABLE I. Kondo temperatures $T_K = T_K(U, \Delta)$ with inverses $\beta_K = 1/T_K$ for different parameter sets. The Kondo temperature was computed via the Bethe ansatz solution of the SIAM, see, e.g., Ref. 45. All quantities have been rounded to three significant digits.

to the temperature-dependent linear broadening γ_L (see App. A). Moreover, the singular values of Keldysh kernels decay much more slowly than their Matsubara counterparts at the same temperature. This reflects the more complex structure, i.e., lower compressibility, of Keldysh vertices.

III. RESULTS

In this section, we show how QTCI performs in compressing the 4p vertex of the single-impurity Anderson model (SIAM) in the Matsubara (Sec. III B) and Keldysh (Sec. III C) formalisms. We discuss the benefits of its QTT representation compared to storing the vertex on dense frequency grids, considering both the core and the full vertex (Γ_{core} and Γ in Eq. (11)). The asymptotic contributions are discussed in App. B. The two most relevant numerical parameters are the number R of quantics bits, corresponding to a grid with 2^R points in each dimension and the maximum bond dimension χ, which serves as a measure for compressibility.

A. Single impurity Anderson model

The Hamiltonian of the single impurity Anderson model (SIAM) [46] reads

$$H = \sum_\sigma \epsilon_d \, n_\sigma + U n_\uparrow n_\downarrow + \sum_{k\sigma} \epsilon_k c^\dagger_{k\sigma} c_{k\sigma} + \sum_{k\sigma} V_k \left(d^\dagger_\sigma c_{k\sigma} + \text{h.c.} \right), \quad n_\sigma = d^\dagger_\sigma d_\sigma \tag{32}$$

where $d^\dagger_\sigma$ with spin $\sigma \in \{\uparrow, \downarrow\}$ creates an electron in an interacting, single-orbital impurity. $c^\dagger_{b\sigma}$ creates an electron in a noninteracting bath, coupled to the impurity via a hybridization term V_k. Electrons on the impurity site interact with the interaction strength U. Since the $c_{k\sigma}$ electrons occur only quadratically, they can formally be integrated out, yielding a frequency-dependent hybridization function $\Delta(\nu)$ as an additional quadratic term for the d electrons. We choose the hybridization function as

$$\Delta(\nu) = \frac{\Delta}{\pi} \ln \left| \frac{\nu + D}{\nu - D} \right| - i\Delta \, \theta(D - |\nu|), \tag{33}$$

with a box-shaped imaginary part, characterized by the bandwidth $2D$ and the hybridization strength $\Delta \in \mathbb{R}$. Moreover, we set $\epsilon_d = -U/2$, which leads to a particle-hole symmetric Hamiltonian.

In the following, energy, temperature and frequencies are measured in units of half the bandwidth $D = 1$. The interaction strength is specified by the dimensionless quantity $u = U/\pi\Delta$. Our analysis covers a wide parameter range from weak ($u = 0.5$) to very strong ($u = 5.0$) interactions and moderate ($\beta = 20$) to low ($\beta = 2000$) temperatures. The corresponding Kondo temperatures are given in Tab. I. (It should be noted that the two datasets for $\beta = 20$ and $\beta = 200$ have $u = 0.5004$ rather than $u = 0.5$. This minor difference changes the Kondo temperature by less than a factor 1.002.) To parametrize the vertex, different frequency conventions and index orderings can be used. Both are listed in App. D. Finally, note that the spin structure of the vertex $\Gamma^{\sigma_1\sigma_2\sigma_3\sigma_4}$ can be simplified by exploiting the SU(2) spin symmetry of the SIAM in the absence of a magnetic field. Only components of the form

$$\Gamma^{\sigma\sigma'} = \Gamma^{\sigma\sigma\sigma'\sigma'} \tag{34}$$

are needed. Moreover, we have $\Gamma^{\downarrow\downarrow} = \Gamma^{\uparrow\uparrow}$ and $\Gamma^{\downarrow\uparrow} = \Gamma^{\uparrow\downarrow}$ by spin flip symmetry, such that only $\Gamma^{\uparrow\uparrow}$ and $\Gamma^{\uparrow\downarrow}$ remain independent. The same applies to $\Gamma^{\sigma\sigma'}_{\text{core}}$.

B. mpNRG vertex functions: Matsubara formalism

Let us first consider the QTCI-compression of the Matsubara core and full vertices. An important input to the QTCI algorithm is the specified error tolerance τ, see Eq. (19). When compressing vertices from mpNRG, the

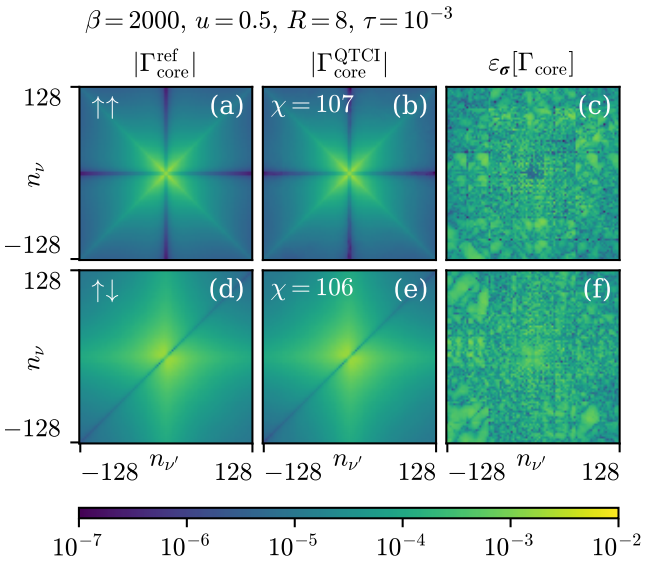


FIG. 3. QTCI-compression of the Matsubara core vertex $\Gamma_{\mathrm{core}}(\omega, \nu, \nu')$ in the p-channel at $\beta = 2000$, with $R = 8$ and tolerance $\tau = 10^{-3}$. Heatmaps show the $\log_{10}$ absolute value of $\Gamma^{\uparrow\uparrow}_{\mathrm{core}}$ in (a,b) and $\Gamma^{\uparrow\downarrow}_{\mathrm{core}}$ in (d,e) on the slice $\omega = 0$. n_ν and $n_{\nu'}$ enumerate the fermionic Matsubara frequencies ν, ν'. Left column: Reference data $\Gamma^{\mathrm{ref}}_{\mathrm{core}}$. Center column: QTCI representation $\Gamma^{\mathrm{QTCI}}_{\mathrm{core}}$. Right column: Normalized error $\varepsilon_{\boldsymbol{\sigma}}[\Gamma_{\mathrm{core}}] \lesssim 1.58 \cdot 10^{-3}$ defined in Eq. (19). We reproduce key features of the vertex on a large frequency box with a comparatively low QTT rank of $\chi = 107$ and $\chi = 106$, respectively.

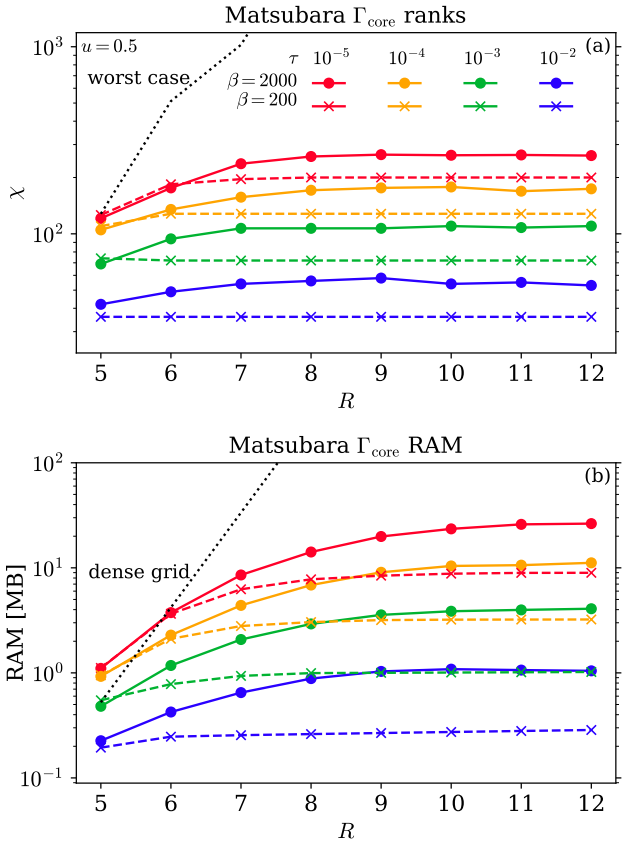


FIG. 4. (a) Rank and (b) RAM usage of the interleaved QTT representation of the Matsubara core vertex $\Gamma^{\uparrow\uparrow}_{\mathrm{core}}$ in the p-channel vs. frequency grid size for different tolerances. The grid has 2^R points in each frequency argument. For the target tolerance of $\tau = 10^{-3}$, ranks saturate at $\chi \approx 100$. Dotted worst-case lines in (a) and (b) indicate the maximum rank of a $3R$-leg QTT (hence the even-odd alternation in the worst case of (a)) and the RAM requirements of dense grids with 2^{3R} points, respectively.

choice of tolerance should be based on the accuracy of the PSFs. Based on benchmark results of Refs. 11, 12, and 15, we expect the mpNRG vertex to be reliable to roughly $10^{-3} \cdot ||\Gamma_{\mathrm{core}}||_\infty$, where the error is partially systematic (as opposed to pure white noise). It should be emphasized that this is only an estimate and inherent errors in mpNRG (due to discretization of the noninteracting bath and discarding high-energy eigenstates during iterative diagonalization) are different from those stemming from TCI. A tolerance significantly below $\tau = 10^{-3}$ may be desirable for two reasons: First, to avoid errors (19) *larger* than our mpNRG accuracy estimate of 10^{-3}: After all, a local error (19) below the tolerance is only expected within the set of pivots that have been sampled by TCI – and even for these, the tolerance is not strictly guaranteed by the TCI routine used here (see [38], Sec. 4.3.1) , which breaks full nesting conditions. Lowering the tolerance increases the confidence that the required accuracy has been reached even outside the sampled set. The second motivation is to assess the potential of our approach for situations where more precise input data is available. We shall therefore investigate tolerances ranging from 10^{-2} to 10^{-5}.

The vertex functions $\Gamma_{\mathrm{core}}(\omega, \nu, \nu')$ and $\Gamma(\omega, \nu, \nu')$ to be represented in QTT format here generally have prominent structures around the origin, along the frequency axes, and along the diagonals [47]. This is exemplified in Fig. 3, which shows slices of Γ_{core} at fixed bosonic frequency $\omega = 0$. The inverse temperature is $\beta = 2000$. We compare reference data with the QTT representation of the vertex for a TCI tolerance of $\tau = 10^{-3}$. The reference was obtained by evaluating the vertex only on the two-dimensional slice shown in Fig. 3. A logarithmic color scale has been chosen to expose imperfections of the TCI approximation. Fig. 3 illustrates how QTCI represents important features of the vertex in a strongly compressed format: For a $256 \times 256 \times 256$ ($R = 8$) frequency grid, we have ranks of $\chi = 107$ for $\Gamma^{\uparrow\uparrow}_{\mathrm{core}}$ and $\chi = 106$ for $\Gamma^{\uparrow\downarrow}_{\mathrm{core}}$. This translates to memory footprints reduced by factors of 92 (268 MB to 2.9 MB) and 89 (268 MB to 3.0 MB), respectively.

A systematic account of the compressibility of Γ_{core} is

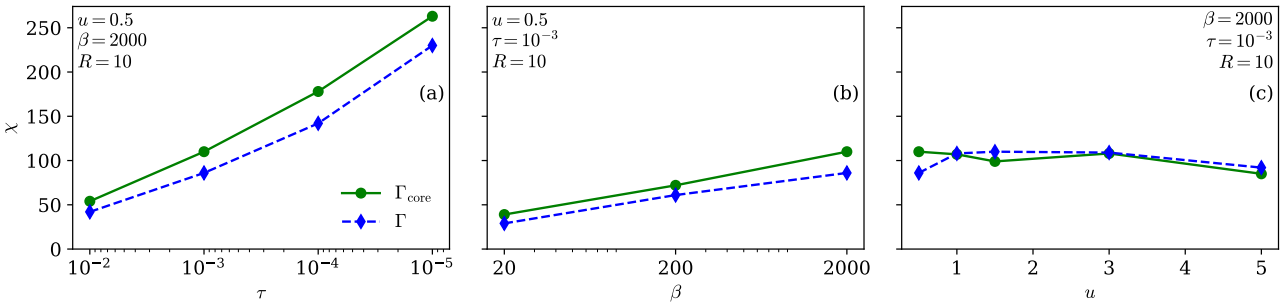


FIG. 5. Rank of the Matsubara core vertex $\Gamma^{\uparrow\uparrow}_{\text{core}}$ and full vertex $\Gamma^{\uparrow\uparrow}$ in the p-channel versus (a) TCI tolerance τ, (b) inverse temperature β and (c) interaction strength u.

provided in Fig. 4. It shows (a) ranks and (b) memory consumption of the resulting QTTs, as a function of tolerance τ and grid size. The grid size is governed by the number R of quantics bits in each dimension. Two different datasets at $\beta = 200$ and $\beta = 2000$ are represented by crosses and circles, respectively. The QTT ranks do not exceed 250 (red circles, $\tau = 10^{-5}$), with compressions for the target accuracy of $\tau = 10^{-3}$ (green) *saturating* w.r.t. R at $\chi = 96$ even for the low temperature data. This rank saturation reflects the simple asymptotic structure of Γ_{core}.

In light of recent work by Rohshap et. al. [28], these results are very promising: There, the authors demonstrate that self-consistent parquet calculations with maximum bond dimensions of 200 are feasible on a single CPU (cf. Ref. 28, Sec. VI B). While the calculations in Ref. 28 were performed in a different parameter regime of the SIAM, their computational cost is determined by the bond dimensions of the QTTs involved. Our results therefore suggest that QTT-based parquet calculations with an NRG Matsubara vertex as input will be feasible. Fig. 4 further shows that TCI ranks of Γ_{core} do not significantly increase beyond a grid size of $R = 8$. In this region of saturated ranks, both memory usage and runtime required for manipulations of the vertex such as convolutions or frequency transformations scale logarithmically in the grid size (linearly in R) [33]. In this regime, the QTT representation yields an exponential reduction in computational cost compared to dense grids. Lowering the tolerance (thus increasing χ) comes at a runtime cost of $O(\chi^4)$ for the most expensive manipulations performed in Ref. 28 (see Sec. V.D there).

The linear scaling in R generically allows for exponentially cheap reduction of discretization errors (for continuous variables) or errors due to finite-size domains (for discrete variables) [33]. For Matsubara vertices, the asymptotic structure contains terms $\Gamma_0, \mathcal{K}_1, \mathcal{K}_2$, and $\mathcal{K}_{2'}$ that are independent of some of the frequencies (see Eq. (11)), implying that the function does not decay to zero at infinity [34]. One might be tempted to conclude that this makes any finite box representation invalid without high-frequency extrapolation. In practice, vertex functions are used in evaluating diagrammatic equations such as expectation values of observables, Bethe–Salpeter equations or the Schwinger–Dyson equation. In all of these cases, the vertex is embedded in a frequency summation or integral with single-particle propagators that *do* approach zero asymptotically for high frequencies. The error generated in such summations and integrals is thus the relevant criterion for frequency box size, as was shown in Ref. 28 for a parquet approach, including Bethe–Salpeter equations. There, the error of a QTCI-based self-consistent calculation of the density channel irreducible vertex was shown to improve from 10^{-1} for $R = 5$ to 10^{-3} for $R = 9$. Frequency grids larger than $R = 8$ are hence relevant. We are considering significantly lower temperatures ($\beta D \in \{200, 2000\}$, $U/\pi\Delta = 0.5$ here vs. $\beta D = 100$, $U/\pi\Delta \approx 0.51$ in Ref. 28). Since Γ_{core} becomes more complicated at these low temperatures (cf. Fig. 5(b) below), we expect large grids to be even more relevant in NRG+parquet calculations.

To assess the range of applicability of our approach, we also examined how the QTT rank for Γ_{core} as well as the full vertex Γ depends on the desired tolerance, inverse temperature β and interaction strength u. The results for the $\uparrow\uparrow$ flavor and an $R = 10$ grid are summarized in Fig. 5. The full and core vertices show a similar increase in rank with the TCI tolerance (Fig. 5(a)), since Γ_{core} contains precisely the complex 3-dimensional structure of the full vertex. Consistent with previous results on random pole based Matsubara correlators (cf. Fig. 8b in Ref. 48), the TCI ranks increase with β, though only logarithmically, see Fig. 5(b). Finally, Fig. 5(c) shows the ranks versus the interaction strength. The key finding is that both vertices remain strongly compressible with ranks ≤ 110 when increasing u from the perturbative regime ($u \ll 1$) to very strong coupling ($u = 5$). Since the y-axis ranges only from 85 to 110, the observed variation in ranks with u does not carry much significance. Over all, Fig. 5 suggests that parquet calculations with an mpNRG vertex as input will be feasible across a wide range of parameters.

According to Eq. (11), the full vertex also contains lower-dimensional contributions K_1^r and $K_{2^{(\prime)}}^r$. In App. B,

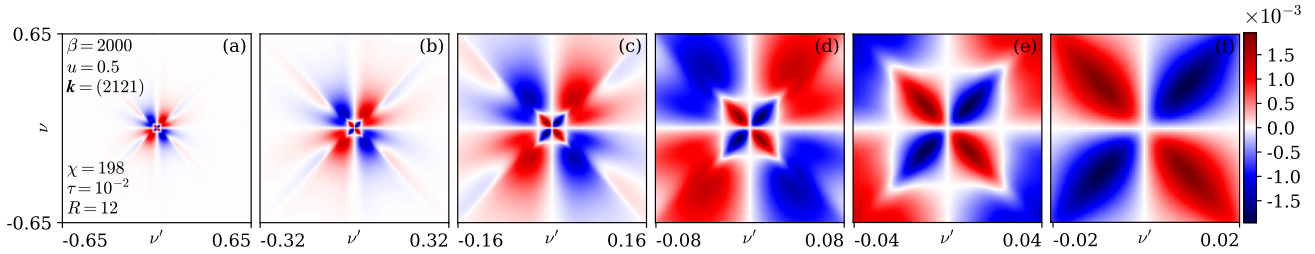


FIG. 6. Imaginary part of the Keldysh core vertex $\Gamma^{2121,\uparrow\uparrow}_{\text{core}}$, at $\omega = 0$, compressed using $R = 12$, $\tau = 10^{-2}$. On this slice, $\text{Re}(\Gamma^{2121,\uparrow\uparrow}_{\text{core}})$ is a factor 25 smaller than $\text{Im}(\Gamma^{2121,\uparrow\uparrow}_{\text{core}})$. The QTT rank is $\chi = 198$. The QTT representation allows us to zoom in by a factor of 2^5 (from left to right) while retaining a sharp resolution throughout.

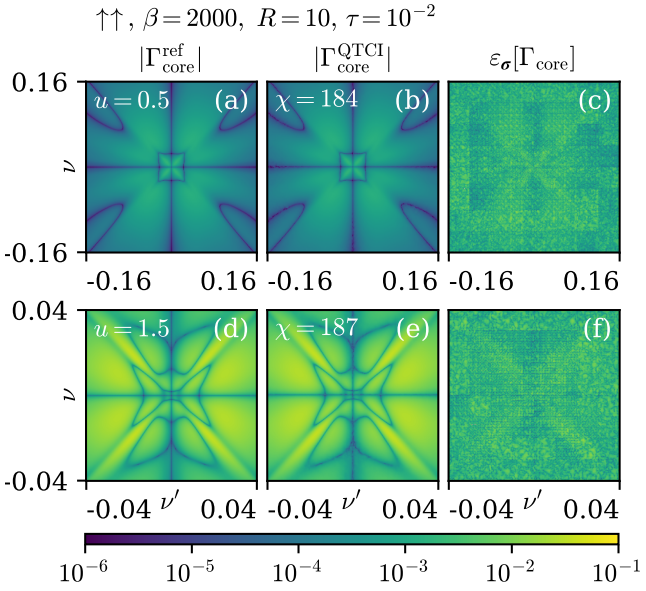


FIG. 7. QTCI-compression of the Keldysh core vertex $\Gamma^{2121,\uparrow\uparrow}_{\text{core}}(\omega, \nu, \nu')$ in the p-channel at $\beta = 2000$, with $R = 10$ and tolerance $\tau = 10^{-2}$. Heatmaps show the $\log_{10}$ absolute value of the vertex on the slice $\omega = 0$. The interaction strengths are $u = 0.5$ in (a,b,c) and $u = 1.5$ in (d,e,f). Left column: Reference data $\Gamma^{\text{ref}}_{\text{core}}$. Center column: QTT representation $\Gamma^{\text{QTCI}}_{\text{core}}$. Right column: Normalized error $\varepsilon_{\boldsymbol{\sigma}}[\Gamma_{\text{core}}] \lesssim 1.73\%$ defined in Eq. (19). The QTT representation captures complex features using moderate bond dimensions of $\chi = 184$ and $\chi = 187$, respectively.

we verify that they have very small TCI ranks ($\chi \lesssim 20$ for $\tau = 10^{-3}$) compared to Γ_{core} and Γ. Moreover, we focused on the $\uparrow\uparrow$ component of the vertex in the p-channel. In App. C, we discuss how the ranks of Γ_{core} and Γ depend on the frequency channel and spin component.

C. mpNRG vertex functions: Keldysh formalism

We now turn to computations of the Keldysh vertex in QTT format. In contrast to the Matsubara vertex, this object gives direct access to real-frequency dynamic response functions, but is a significantly more complicated function on a continuous domain of real frequencies. Faithfully capturing its structure on a finite grid while keeping the computational cost in check is very challenging. This has been achieved in Refs. 20 and 21, but requires tedious manual tuning of nonlinear grids. By contrast, our QTCI-based approach allows us to automatically capture features on different length scales on an extremely fine equidistant grid. Our grid for ω contains 0, while ν and ν' live on a grid that is offset from 0 by half a grid spacing. An alternative choice would be to include 0 in all three grids. In QTCI, we can refine the grid until all features are represented up to a given tolerance, so that shifting the ν and ν' grids by half a grid spacing does not make a difference. The resolution attained with QTCI is exemplified in Fig. 6, showing a QTT representation of $\Gamma^{2121,\uparrow\uparrow}_{\text{core}}$ on a slice at $\omega = 0$ using $R = 13$ quantics bits. The TCI tolerance was set to 10^{-2}. All panels show the same slice, but zoom in by factors of 2 moving from left to right. The rightmost panel still exhibits a sharp resolution after a 32-fold magnification. As a further illustration, Fig. 7 compares the TCI-compressed vertex (center) to the reference (left), showing the normalized error on the right. The slices are taken again at $\omega = 0$ and for $u = 0.5$ (top row) and $u = 1.5$ (bottom row). Overall, we see that TCI resolves the core vertex to 1% precision with ranks of 184 and 187, respectively.

This 1% error is comparable to the uncertainty due to the broadening of spectral functions, which supersedes the NRG error of 10^{-3} in the real-frequency case: While there are well-established schemes [12, 15] to choose broadening parameters, legitimate choices can vary within a range that causes vertex functions to change by a few percent. Our default tolerance for Keldysh objects is therefore chosen as $\tau = 10^{-2}$. In view of ongoing research aiming to develop an impurity solver less susceptible to broadening artifacts [49], which could be extended to the multipoint case in the future, we extend our investigations down to $\tau = 10^{-3}$.

Figure 8 shows (a) the rank and (b) the memory size of the compressed Keldysh core vertex component $\Gamma^{2121,\uparrow\uparrow}_{\text{core}}$ at temperatures $\beta = 200$ and 2000 versus the number of

quantics bits R in each dimension. This Keldysh component of $\Gamma^{\boldsymbol{k}}_{\text{core}}$ was found to have the highest bond dimension (see App. C, Fig. 12). We set a fixed box size of $\omega_{\max} = 0.65$ (cf. Fig. 6(a)) and increase the resolution with R. We verified that the chosen box size is large enough to capture all relevant structures within the target tolerance $\tau = 10^{-2}$. At this tolerance, the rank shown in Fig. 8 saturates at $\chi = 202$. This rank is again of a magnitude where a self-consistent parquet calculation in the Matsubara formalism was shown to be feasible on a single core in Ref. 28. The 16 components of the Keldysh vertex can be inferred from just 5 components using complex conjugation and crossing symmetry [50]. Nevertheless, in follow-up computations such as solving the parquet equations, these multiple Keldysh components in contrast to a single Matsubara vertex may necessitate parallelization already for $\chi \approx 200$. Multithreaded or distributed schemes will certainly be required for the most difficult case considered here ($\tau = 10^{-3}$ and $\beta = 2000$), which results in ranks of $\chi \approx 450$. On a different note, the QTT vertex has a vastly reduced memory footprint, as shown in Fig. 8(b): For $\beta = 2000$, $\tau = 10^{-2}$ and $R = 10$, it requires 11.3 MB of memory, compared to 17.1 GB for a dense grid representation; this corresponds to a compression ratio of 1 : 1513. Although real-frequency diagrammatic calculations for the SIAM are limited by runtime rather than memory [20], this paves the way for investigation of more complicated models with multiple orbitals or momentum dependence, which have prohibitive memory requirements if attempted with dense grids [51–53].

In Fig. 9 we explore the compressibility of the core and full vertices for varying tolerance τ, inverse temperature β and interaction strength u. As seen before in Fig. 8, lowering the tolerance below 10^{-2} results in a steep increase in the rank. As in the Matsubara formalism, the rank increases with β, but only slowly. Panel (c) reveals a much less predictable behavior: The ranks of both the full and core vertices reach a maximum at $u = 1.0$ and decrease significantly for large u. Moreover, the rank of the full vertex Γ approaches that of Γ_{core} from below with increasing interaction u. This reflects the increasing magnitude of Γ_{core} relative to the asymptotic contributions $\mathcal{K}^r_1$ and $\mathcal{K}^r_{2(\prime)}$: At weak interaction, the magnitude of the core vertex is much smaller than that of the full vertex, which is dominated by $\mathcal{K}^r_1$ and $\mathcal{K}^r_{2(\prime)}$. Since TCI measures the error relative to the supremum norm of the target function (cf. Eq. (19), this means that Γ_{core} need not be resolved as accurately at weak interaction. The compression of the asymptotic contributions $\mathcal{K}^r_1$ and $\mathcal{K}^r_{2(\prime)}$ is discussed in App. B, together with their Matsubara counterparts. In App. C, we discuss how the ranks of $\Gamma^{\boldsymbol{k}}_{\text{core}}$ and $\Gamma^{\boldsymbol{k}}$ depend on flavor, frequency channel and Keldysh component $\boldsymbol{k}$.

Finally, we discuss how compressing each Keldysh component of the vertex separately, as was done in this work, compares to running TCI on the entire Keldysh core or full vertex, where the Keldysh components are encoded in an additional leg of a single tensor train. In both cases,

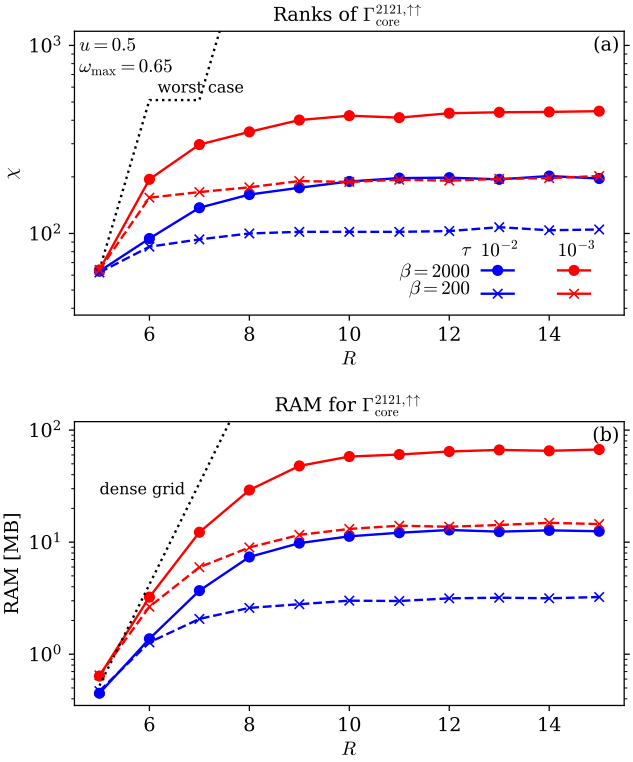


FIG. 8. (a) Rank and (b) RAM usage of Keldysh core vertex $\Gamma^{2121,\uparrow\uparrow}_{\text{core}}(\omega, \nu, \nu')$ in the p-channel vs. frequency grid size for different tolerances and temperatures. The grid has 2^R points in each frequency argument. For a tolerance of $\tau = 10^{-2}$ (blue), the bond dimension saturates at $\chi \approx 200$. Dotted worst-case lines in (a) and (b) indicate the maximum rank of an R-leg QTT (fused representation) and the RAM requirements of dense grids with 2^{3R} points, respectively.

spin components are compressed separately. For a fair comparison of these two approaches, recall the following: (i) TCI measures the interpolation error relative to the supremum norm of the target function (see Eq. (19)). When compressing the entire vertex, any given Keldysh component $\Gamma^{\boldsymbol{k}}$ (or $\Gamma^{\boldsymbol{k}}_{\text{core}}$) should therefore be normalized by $||\Gamma^{\boldsymbol{k}}||_\infty$ (or $||\Gamma^{\boldsymbol{k}}_{\text{core}}||_\infty$), i.e., with the supremum norm of the same Keldysh component $\boldsymbol{k}$. Only then does one achieve the same accuracy as in separate compressions of Keldysh components. (ii) MPO–MPO contractions, the most expensive operations in QTT-based diagrammatic calculations, scale as $\mathcal{O}(R\chi^4)$ in runtime. As a preliminary investigation, we compressed the entire core vertex at $u = 0.5, \beta = 2000$ and $\omega_{\max} = 0.65$ with a tolerance of $\tau = 10^{-2}$ and $R = 8$ quantics bits. The tensor leg for the Keldysh component was placed to the very left and only included the five Keldysh components not related by crossing symmetry or complex conjugation (cf. App. C). The resulting rank χ is compared with the ranks $\chi_{\boldsymbol{k}}$ of individual Keldysh components in Tab. II. We observe the ratio $\chi^4 / \max_{\boldsymbol{k}} \chi^4_{\boldsymbol{k}} \approx 21.77$, thus MPO-MPO contractions

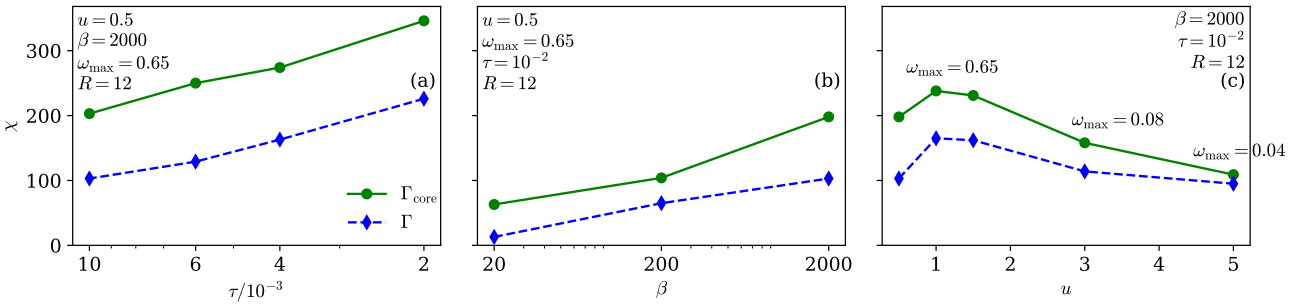


FIG. 9. Rank of Keldysh core vertex $\Gamma^{\uparrow\uparrow}_{\text{core}}$ and full vertex $\Gamma^{\uparrow\uparrow}$ in the p-channel versus (a) TCI tolerance τ, (b) inverse temperature β and (c) interaction strength u. We show data for the $\boldsymbol{k} = (2121)$ Keldysh component. In panel (c), we choose smaller box sizes $\omega_{\max} = 0.08$ and $\omega_{\max} = 0.04$ for $u = 3.0$ and $u = 5.0$, respectively. This is because the extent of the core vertex decreases at these strong interactions.

component $\boldsymbol{k}$	1111	2111	2121	2112	1222	**all**
rank χ	130	136	159	126	95	**344**

TABLE II. QTT ranks of the five Keldysh components not related by crossing symmetry or complex conjugation (center columns) compared to the QTT rank of a single tensor that contains all five components (rightmost column). Parameters are $u = 0.5, \beta = 2000, \omega_{\max} = 0.65, \tau = 10^{-2}$ and $R = 8$.

take about 22 times longer for two entire vertices than for two individual components. On the other hand, the latter type of contraction would have to be performed $5^2 = 25$ times. However, Tab. II shows that some Keldysh components have a significantly lower bond dimension than $\max_{\boldsymbol{k}} \chi_{\boldsymbol{k}}$. In summary, both approaches are worth investigating, and a conclusive comparison is only possible in the context of a specific QTT-based diagrammatic code.

IV. SUMMARY AND OUTLOOK

We have presented a QTCI-based method for representing imaginary- and real-frequency mpNRG vertex functions on large grids that are far beyond the reach of previous implementations. The QTCI algorithm allows us to automatically capture all relevant features of the vertex up to a prescribed accuracy and represents the result in a highly compressed format. Repeated sampling during TCI sweeps necessitates optimizations of the vertex evaluation, which we described in detail. We studied the compressibility of the vertex in a systematic fashion: Imaginary- and even real-frequency vertices are representable as QTTs with maximum bond dimensions sufficiently small ($\chi \approx$ a few hundred) to allow for diagrammatic computations with these objects. This holds true across a broad range of temperatures and interaction strengths, both for the full vertex as well as its asymptotic decomposition. Our work thus constitutes an important step toward QTCI-based diagrammatic calculations which use a nonperturbative DMFT vertex as input, and suggests that these will be feasible. The next step will be to implement a QTCI-based diagrammatic extension of DMFT that augments the local vertex with momentum dependence. An analogous program can be envisioned in the real-frequency setting. Though this is a challenging, computationally demanding endeavor, it would achieve a long-sought goal: a method to obtain nonlocal, real-frequency dynamical response functions of strongly correlated systems.

DATA AND CODE AVAILABILITY

The mpNRG computations were performed with the MuNRG package [12, 54, 55], which is based on the QSpace tensor network library [56–59]. The latest version of QSpace is available [60] and a public release of MuNRG is intended. The code used in this work to compute and compress vertices is available on GitHub, see Ref. 61. The partial spectral functions required as input for that code can be found in Ref. 62.

ACKNOWLEDGMENTS

We thank Seung-Sup Lee and Jae-Mo Lihm for helpful discussions. We thank Seung-Sup Lee for providing the mpNRG code used for computing the PSFs.

This work was funded in part by the Deutsche Forschungsgemeinschaft under Germany's Excellence Strategy EXC-2111 (Project No. 390814868). It is part of the Munich Quantum Valley, supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus. We further acknowledge support from the DFG grant LE 3883/2-2. We gratefully acknowledge computational resources from grant INST 86/1885-1 FUGG of the German Research Foundation (DFG). MF, AG, MR, and NR acknowledge support from the International Max-Planck Research School for Quantum Science and Technology (IMPRS-QST). NR acknowledges funding

from the Studienstiftung des deutschen Volkes and the Marianne-Plehn-Programm of the state of Bavaria.

Appendix A: Computing broadened Keldysh kernels

In this section, we provide details on the numerical computation of the broadened Keldysh kernels $K_b^{[\lambda]}$ appearing in Eq. (10b). It consists of two steps: broadening the Dirac-δ functions in Eq. (3) to $\delta_b(\omega', \epsilon)$, and convolution of $\delta_b(\omega', \epsilon)$ with the Keldysh kernel $(\omega' + \mathrm{i}\gamma_{0,i}^{\lambda})^{-1}$. The numerical details of this procedure are taken from the MuNRG code of Ref. 12.

The broadening combines symmetric log-Gaussian and linear broadening (cf. Ref. 15, App. E.2):

$$\delta_b(\omega', \epsilon) = \int_{\mathbb{R}} \mathrm{d}\epsilon' \delta_{\mathrm{F}}(\omega', \epsilon') \delta_{\mathrm{sLG}}(\epsilon', \epsilon), \tag{A1a}$$

$$\delta_{\mathrm{sLG}}(\epsilon', \epsilon) = \frac{\Theta(\epsilon'\epsilon)}{\sqrt{\pi}\sigma_{\mathrm{sLG}}|\epsilon|} \exp\left[-\left(\frac{\ln|\epsilon/\epsilon'|}{2\sigma_{\mathrm{sLG}}} - \frac{\sigma_{\mathrm{sLG}}}{4} \right)^2 \right], \tag{A1b}$$

$$\delta_{\mathrm{F}}(\omega', \epsilon') = \frac{1}{2\gamma_{\mathrm{L}}} \left(1 + \cosh \frac{\omega' - \epsilon'}{\gamma_{\mathrm{L}}} \right)^{-1}. \tag{A1c}$$

The broadening parameters γ_{L} and σ_{sLG} along with other numerical settings for all physical parameter sets are listed in Tab. III. However, the linear broadening γ_L is multiplied with a prefactor that depends on the current permutation p, the fully retarded index λ and the dimension i. This scheme will be explained further at the end of this section (see Eq. (A4)). The integral (A1a) is performed by trapezoidal quadrature, where ω' and ϵ' are discretized on logarithmic grids. These grids contain 0, are symmetric around the origin and range from $e_{\min}$ to $e_{\max}$ with e_{step} points per decade (see Tab. III). For the ϵ' grid, $e_{\min}$ is automatically replaced by a lower boundary $x_{\min}$ with $0 < x_{\min} < e_{\min}$ if the low-frequency tail of the log-Gaussian broadening kernel extends below $e_{\min}$. This ensures an accurate integration of $\delta_{\mathrm{sLG}}(\epsilon', \epsilon)$. The energies ϵ specifying the location of the spectral function peaks also reside on a logarithmic grid, which arises from the mpNRG computation.

The numerical integration described above yields $\delta_b(\omega', \epsilon)$ with ω' and ϵ on logarithmic grids. To convolve δ_b with the Keldysh kernel $(\omega' + \mathrm{i}\gamma_{0,i}^{\lambda})^{-1}$, we use the identity

$$\begin{aligned} &\lim_{\gamma_0 \to 0^+} \int_{\mathbb{R}} \mathrm{d}\omega' \frac{\delta_b(\omega', \epsilon)}{\omega - \omega' + \mathrm{i}\gamma_{0,i}^{\lambda}} = \\ &= \mathcal{P} \int_{\mathbb{R}} \mathrm{d}\omega' \frac{\delta_b(\omega', \epsilon)}{\omega - \omega'} - \mathrm{i}\pi \mathrm{sgn}(\gamma_{0,i}^{\lambda}) \delta_b(\omega, \epsilon), \end{aligned} \tag{A2}$$

where $\mathcal{P}$ denotes the Cauchy principal value (PV) integral. Also, recall the definition (10c) of $\gamma_{0,i}^{\lambda}$. Importantly, $\delta_b(\omega', \epsilon)$ has been computed on a logarithmic grid, while ω in the broadened Keldysh kernel $K_b^{[\lambda]}(\omega, \epsilon)$ defined in Eq. (10b) resides on a linear grid. This is because the external frequency grids on which we compute vertices are also linear. To obtain the imaginary part of Eq. (A2) on the linear grid, we use linear interpolation of $\delta_b(\omega, \epsilon)$ in the argument ω. Computing the real part, i.e., the PV integral is slightly more involved: By the linear interpolation performed for the imaginary part, $\delta_b(\omega, \epsilon)$ can be viewed as a piecewise linear function. We split the PV integral over $\delta_b(\omega', \epsilon)$ into PV integrals over linear functions $(a_{i,\epsilon}\omega' + b_{i,\epsilon})$ on intervals $[\omega_i', \omega_{i+1}']$. These are evaluated using the formula

$$\begin{aligned} &\mathcal{P} \int_{\omega_i'}^{\omega_{i+1}'} \mathrm{d}\omega' \frac{a_{i,\epsilon}(\omega' - \omega_i) + b_{i,\epsilon}}{\omega - \omega'} = \\ &- a_{i,\epsilon}(\omega_{i+1}' - \omega_i') - (a_{i,\epsilon}(\omega - \omega_i') + b_{i,\epsilon}) \ln \left| \frac{\omega - \omega_{i+1}'}{\omega - \omega_i'} \right|. \end{aligned} \tag{A3}$$

The sum over all PV integrals of the form (A3) then yields the real part of the broadened Keldysh kernel $K_b(\omega, \epsilon)$. For this scheme to be accurate, the extent $[-e_{\max}, e_{\max}]$ of the logarithmic ω' grid should be significantly larger than the frequency box delimited by $\omega_{\max}$. Comparing the values of $e_{\max}$ given in Tab. III with our default frequency box size $\omega_{\max} = 0.65$, one verifies that this is the case.

We now turn to the prefactors of the linear broadening γ_{L} mentioned above. In the linear broadening kernel $\delta_{\mathrm{F}}(\omega'_{\bar{1}\ldots\bar{i}}, \epsilon_i')$ to be convolved with the Keldysh kernel $(\omega'_{\bar{1}\ldots\bar{i}} + \gamma_{0,i}^{\lambda})^{-1}$, the broadening with γ_{L} is replaced by

$$\gamma_{\mathrm{L},i}^{\lambda} = \begin{cases} \gamma_{\mathrm{L}} \cdot (\ell - i) & \text{for} \quad i \geq \lambda, \\ \gamma_{\mathrm{L}} \cdot i & \text{for} \quad i < \lambda. \end{cases} \tag{A4}$$

This choice was found to reduce broadening artifacts in an mpNRG treatment of the Hubbard atom in Ref. 63. Moreover, composite operators q_{ij} in 3p correlators (cf. Ref. 15, Eq. (96)) receive a doubled linear broadening. This was found to cancel discretization and broadening artifacts when computing $\mathcal{K}_{2^{(\prime)}}^r$ by multiplication with self-energies (see Eq. (B3)), see Ref. 63. The broadening of 3p correlators is exemplified in Tab. IV. Finally, the 2p correlators required for self-energies in the symmetric estimators for Γ_{core} and $K_{2^{(\prime)}}^r$ (cf. Sec. II D and App. B.) are broadened according to Tab. V.

u	β	σ_{sLG}	γ_{L}
0.5	20/200	0.693	T
0.5/1.0/1.5	2000	0.4	T
3.0/5.0	2000	0.4	T

TABLE III. Broadening settings for different NRG datasets. $T = 1/\beta$ denotes the temperature. See main text for definitions of the parameters. We set $e_{\min} = 10^{-6}, e_{\max} = 10^4$ and $e_{\mathrm{step}} = 50$. For TCI tolerances $\tau \leq 3.4 \cdot 10^{-3}$, the integration grid was refined to $e_{\mathrm{step}} = 200$ to avoid fitting of numerical noise by the TCI algorithm. We also set $e_{\mathrm{step}} = 200$ to broaden 2p functions.

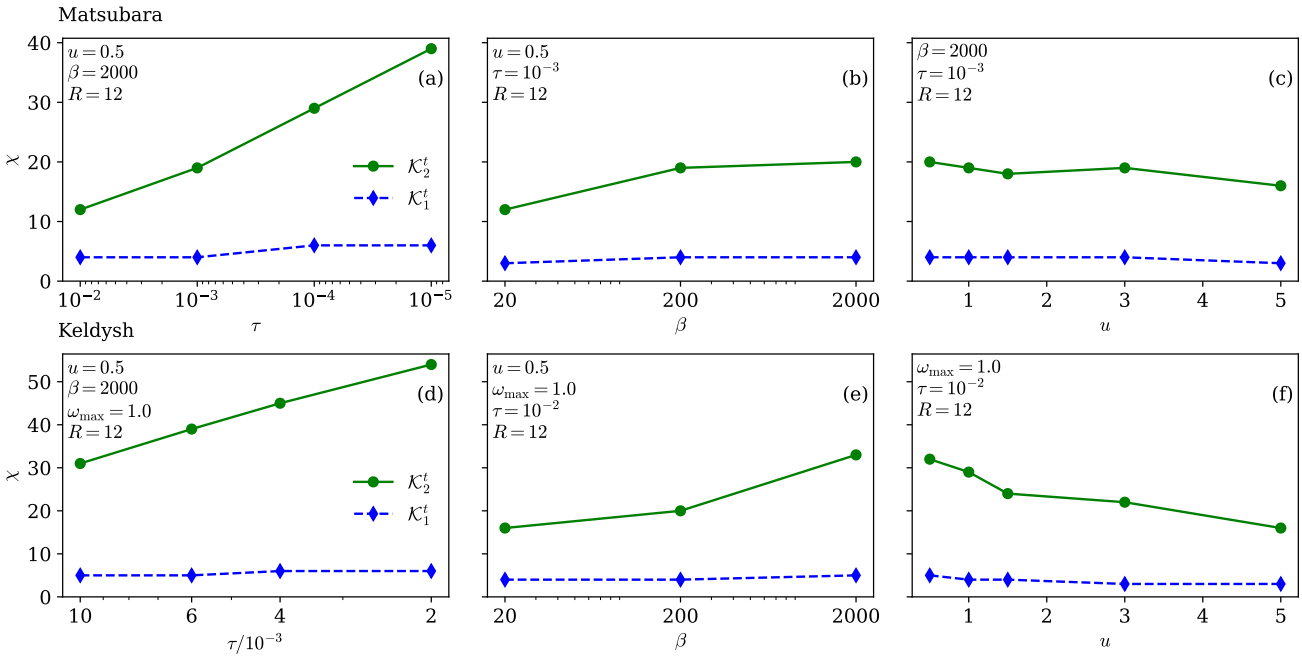


FIG. 10. QTT ranks of Matsubara (top row, (a–c)) and Keldysh (bottom row, (d–f)) $\mathcal{K}_1^{t,\uparrow\uparrow}$ and $\mathcal{K}_2^{t,\uparrow\uparrow}$ contributions to the full vertex versus: (a,d) tolerance τ, (b,e) inverse temperature β and (c,f) interaction strength u. In Keldysh we chose, of all components, the component $\boldsymbol{k}=(22)$ for $\mathcal{K}_1^{t,\uparrow\uparrow}$ and $\boldsymbol{k}=(112)$ for $\mathcal{K}_2^{t,\uparrow\uparrow}$. These components were found to have the highest rank, respectively.

	$i=1$	$i=2$
$\lambda=1$	$2\gamma_L$	γ_L
$\lambda=2$	$2\gamma_L$	γ_L
$\lambda=3$	$2\gamma_L$	$3\gamma_L$

	$i=1$	$i=2$
$\lambda=1$	$3\gamma_L$	γ_L
$\lambda=2$	γ_L	γ_L
$\lambda=3$	γ_L	$3\gamma_L$

TABLE IV. Linear broadening of a 3p correlator with doubled broadening on the composite operator q_{ij}. The operator q_{ij} is in the first slot of the operator tuple for the identity permuation $p=[123]$. Left: Permutation $p=[123]$. Right: Permutation $p=[213]$.

	$i=1,\ p=[12]$	$i=1,\ p=[21]$
$\lambda=1$	$3\gamma_L$	γ_L
$\lambda=2$	γ_L	$3\gamma_L$

TABLE V. Linear broadening of a 2p function used for the aIE self-energy. The two rightmost columns correspond to the two possible permutations.

Appendix B: Compression of 1D and 2D vertex contributions

A QTCI-based parquet calculation exploiting the asymptotic decomposition (11) requires not only Γ_{core}, but also $\mathcal{K}_1^r$ and $\mathcal{K}_{2(\prime)}^r$ represented as QTTs. Recall that $r=a,p,t$ labels the three frequency channels. In this section, we verify that these asymptotic contributions indeed have a significantly lower rank than Γ_{core}, as expected from their simpler structure.

The $\mathcal{K}_1^r$ contributions are simply given by 2p correlators of composite operators, see Ref. 15, Sec. IV.F. We illustrate the evaluation of $\mathcal{K}_{2(\prime)}^r$ using $\mathcal{K}_2^t$ as an example. For derivations and the remaining $\mathcal{K}_{2(\prime)}^r$ components we refer to Ref. 15, Secs. IV.C and IV.F. Since the self-energy is spin-diagonal, we again omit spin indices. First the operator $q=[d,H_{\text{int}}]$ introduced in Sec. II D is used to define the operator $q_{34}=\{q,d^\dagger\}=qd^\dagger+d^\dagger q$. One further introduces 3p correlators $G[q_{34},a_1,a_2^\dagger]$, where $a_1,a_2\in\{d,q\}$. They are defined in terms of connected correlators as

$$G^{\boldsymbol{k}}[q_{34},a_1,a_2^\dagger]=P^{k_1k_2(k_3+k_4)}G^{\boldsymbol{k}}_{\text{con}}[q_{34},a_1,a_2^\dagger]. \tag{B1}$$

In the Keldysh formalism, the tensor P reads

$$P^{k_1k_2(k_3+k_4)}=\begin{cases}\frac{1}{\sqrt{2}} & \text{if } \sum_i k_i \text{ is odd,}\\ 0 & \text{else,}\end{cases} \tag{B2}$$

while it is set to unity in Matsubara. Using the symbol Y_{x_i} introduced in Eq. (13), K_2^t can then be expressed as:

$$\mathcal{K}_2^t(\omega_t,\nu_t)=\sum_{a_1,a_2\in\{d,q\}}Y_{a_1}G[q_{34},a_1,a_2^\dagger](-\omega_{12},\omega_1,\omega_2)Y_{a_2}. \tag{B3}$$

To evaluate $\mathcal{K}_2^t$, the external frequencies ω_1,ω_2 appearing on the RHS of Eq. (B3) are expressed in the t-channel parametrization, i.e., in terms of ω_t and ν_t.

The ranks of Matsubara and Keldysh asymptotic contributions $\mathcal{K}_1^{t,\uparrow\uparrow}$, $\mathcal{K}_2^{t,\uparrow\uparrow}$ for different parameters are shown

in Fig. 10(a–c) and 10(d-f), respectively. We use the t-channel frequency parametrization, thus viewing $\mathcal{K}_1^t(\omega_t)$ as a 1D and $\mathcal{K}_2^t(\omega_t, \nu_t)$ as a 2D function. A comparison of Fig. 10 with Figs. 5 and 9 confirms that the three-dimensional vertex functions will dominate the cost of a diagrammatic calculation: For a tolerance of $\tau = 10^{-3}$, the ranks of the Matsubara $\mathcal{K}_2^t$ component are no larger than 20. The variation of the rank with β and u does therefore not bare much significance. For smaller tolerances, the rank of $\mathcal{K}_2^t$ remains much lower than that of the core vertex. Analogous observations hold for the Keldysh $\mathcal{K}_2^t$ component with a target tolerance of $\tau = 10^{-2}$. Like the Keldysh core vertex, its rank increases slowly with β and decreases for strong coupling u, but the changes are small compared to the core vertex.

Appendix C: Compression for different channels, flavors, Keldysh components

In this section we investigate the compressibility of core and full vertices for different flavors ($\uparrow\uparrow, \uparrow\downarrow$), frequency channels ($r = a, p, t$) and Keldysh components $\boldsymbol{k} = (k_1 k_2 k_3 k_4)$.

Fig. 11 shows the QTT ranks of the Matsubara and Keldysh full and core vertices for different channels and flavors, at $\beta = 2000$ and $u = 0.5$. The tolerances are $\tau = 10^{-3}$ and $\tau = 10^{-2}$ for Matsubara and Keldysh vertices, respectively. We observe that the p-channel exhibits the highest ranks throughout, which is why we used this frequency parametrization in the main text. The QTT ranks of the Matsubara vertices shown in Fig. 11(c) barely differ between the two flavors. By contrast, $\Gamma_{\text{core}}^{2121,\uparrow\uparrow}$ has a significantly higher rank than $\Gamma_{\text{core}}^{2121,\uparrow\downarrow}$ ($\chi = 198$ vs. $\chi = 154$).

The rank of $\Gamma_{\text{core}}^{\uparrow\uparrow}$ and $\Gamma^{\uparrow\uparrow}$ depending on the Keldysh component is shown in Fig. 12. Only components that are not related by crossing or complex conjugation symmetry [50] are considered. We show data for $\beta = 2000$, $u = 0.5$ and $\tau = 10^{-3}$. The (2121) component of Γ_{core} is found to have the highest rank. We therefore selected $\Gamma_{\text{core}}^{2121,\uparrow\uparrow}$ for our analysis of rank saturation and parameter dependence in Figs. 8 and 9.

Appendix D: Frequency conventions

We use the following parametrizations for the t (particle-hole), p (particle-particle) and a (transverse particle-hole) channels:

$$\boldsymbol{\omega} = \begin{cases} (-\nu_r, \omega_r + \nu_r, -\omega_r - \nu_r', \nu_r') & \text{for } r = t \text{ (ph)}, \\ (-\nu_r, \omega_r - \nu_r', -\omega_r + \nu_r, \nu_r') & \text{for } r = p \text{ (pp)}, \\ (-\nu_r, \nu_r', -\omega_r - \nu_r', \omega_r + \nu_r) & \text{for } r = a \text{ ($\overline{\text{ph}}$)}. \end{cases} \tag{D1}$$

These are the same as in Ref. 15, up to a global minus sign. Spin and, if present, Keldysh indices of the

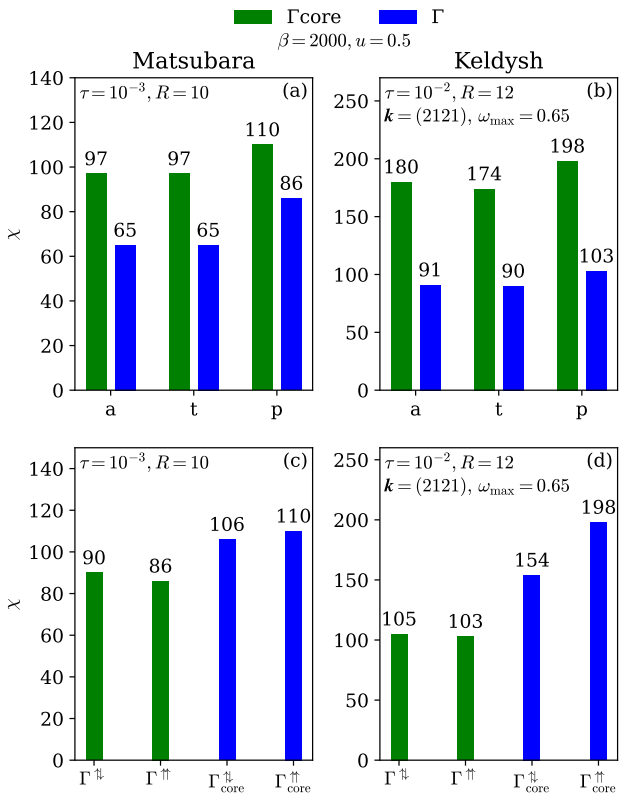


FIG. 11. Top row (a,b): TCI ranks of $\Gamma^{\uparrow\uparrow}$ and $\Gamma_{\text{core}}^{\uparrow\uparrow}$ in the three channels a, p, t. Bottom row (c,d): TCI ranks of Γ and Γ_{core} in the p-channel for the two flavors $\uparrow\uparrow$ and $\uparrow\downarrow$. Matsubara vertices (a,c) were compressed with $\tau = 10^{-3}$ and $R = 10$, Keldysh vertices (b,d) with $\tau = 10^{-2}$ and $R = 12$.

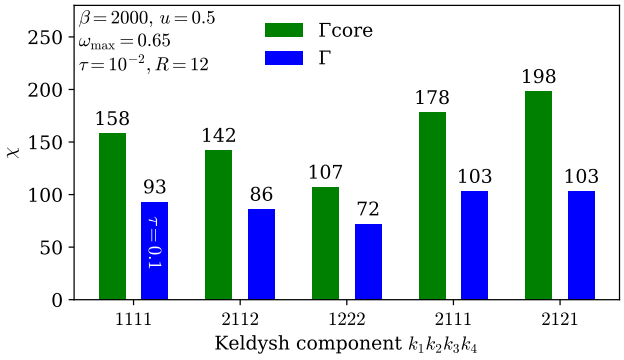


FIG. 12. QTT rank of Keldysh core vertex $\Gamma_{\text{core}}^{\boldsymbol{k},\uparrow\uparrow}$ in the p-channel vs. Keldysh component. From the 16 Keldysh components, only those are shown which are not related by crossing symmetry or complex conjugation. The $\Gamma^{1111,\uparrow\uparrow}$ component of the full vertex was compressed with tolerance $\tau = 0.1$, since it is about a factor 10 smaller than the other Keldysh components of the full vertex. This is because the $\mathcal{K}_1^r$ contributions, which dominate other components $\Gamma^{\boldsymbol{k}\neq 1111}$, vanish in the $\boldsymbol{k} = (1111)$ component.

vertex $\Gamma^{\boldsymbol{k},\sigma_1\sigma_2\sigma_3\sigma_4}(\boldsymbol{\omega})$ are ordered according to the underlying impurity Green's function $G^{\boldsymbol{k}}_{\mathrm{con}}[d_{\sigma_1}d^\dagger_{\sigma_2}d_{\sigma_3}d^\dagger_{\sigma_4}](\boldsymbol{\omega})$. Finally, the evaluation of a 2p correlator G at a frequency ν is defined as $G(\nu,-\nu)$ in our convention. This is relevant for evaluating the self-energy Σ in Eq. (13), because computing the self-energy comes down to evaluating 2p correlators according to the asymmetric estimators (see Eq. (27) in Ref. [15]) we employed.

[1] W. Metzner, M. Salmhofer, C. Honerkamp, V. Meden, and K. Schönhammer, Functional renormalization group approach to correlated fermion systems, Rev. Mod. Phys. **84**, 299 (2012).

[2] F. B. Kugler and J. von Delft, Multiloop functional renormalization group that sums up all parquet diagrams, Phys. Rev. Lett. **120**, 057403 (2018).

[3] F. B. Kugler and J. von Delft, Multiloop functional renormalization group for general models, Phys. Rev. B **97**, 035162 (2018).

[4] F. B. Kugler and J. von Delft, Derivation of exact flow equations from the self-consistent parquet relations, New J. Phys. **20**, 123029 (2018).

[5] N. E. Bickers, Self-Consistent Many-Body Theory for Condensed Matter Systems, in *Sénéchal D., Tremblay A.-M., Bourbonnais C. (eds), Theoretical Methods for Strongly Correlated Electrons, CRM Series in Mathematical Physics* (Springer, New York, 2004).

[6] G. Rohringer, H. Hafermann, A. Toschi, A. A. Katanin, A. E. Antipov, M. I. Katsnelson, A. I. Lichtenstein, A. N. Rubtsov, and K. Held, Diagrammatic routes to nonlocal correlations beyond dynamical mean field theory, Rev. Mod. Phys. **90**, 025003 (2018).

[7] A. Georges, G. Kotliar, W. Krauth, and M. J. Rozenberg, Dynamical mean-field theory of strongly correlated fermion systems and the limit of infinite dimensions, Reviews of Modern Physics **68**, 13 (1996).

[8] C. Taranto, S. Andergassen, J. Bauer, K. Held, A. Katanin, W. Metzner, G. Rohringer, and A. Toschi, From infinite to two dimensions through the functional renormalization group, Phys. Rev. Lett. **112**, (2014).

[9] A. Toschi, A. A. Katanin, and K. Held, Dynamical vertex approximation: A step beyond dynamical mean-field theory, Physical Review B **75**, 045118 (2007).

[10] K. Held, A. A. Katanin, and A. Toschi, Dynamical vertex approximation: An introduction, Prog. Theor. Phys. Supp. **176**, 117 (2008).

[11] F. B. Kugler, S.-S. B. Lee, and J. von Delft, Multipoint correlation functions: Spectral representation and numerical evaluation, Phys. Rev. X **11**, 041006 (2021).

[12] S.-S. B. Lee, F. B. Kugler, and J. von Delft, Computing local multipoint correlators using the numerical renormalization group, Phys. Rev. X **11**, 041007 (2021).

[13] K. G. Wilson, The renormalization group: Critical phenomena and the Kondo problem, Reviews of Modern Physics **47**, 773 (1975).

[14] R. Bulla, T. A. Costi, and T. Pruschke, Numerical renormalization group method for quantum impurity systems, Reviews of Modern Physics **80**, 395 (2008).

[15] J.-M. Lihm, J. Halbinger, J. Shim, J. von Delft, F. B. Kugler, and S.-S. B. Lee, Symmetric improved estimators for multipoint vertex functions, Phys. Rev. B **109**, 125138 (2024).

[16] N. Ritz, A. Ge, M. Frankenbach, M. Pelz, J. von Delft, and F. B. Kugler, Testing the parquet equations and the U(1) Ward identity for real-frequency correlation functions from the multipoint numerical renormalization group (2025), arXiv:2504.05910 [cond-mat.str-el].

[17] H. R. Krishna-murthy, J. W. Wilkins, and K. G. Wilson, Renormalization-group approach to the Anderson model of dilute magnetic alloys. I. Static properties for the symmetric case, Phys. Rev. B **21**, 1003 (1980).

[18] R. Bulla, A. C. Hewson, and T. Pruschke, Numerical renormalization group calculations for the self-energy of the impurity Anderson model, Journal of Physics: Condensed Matter **10**, 8365 (1998).

[19] K.-M. Tam, H. Fotso, S.-X. Yang, T.-W. Lee, J. Moreno, J. Ramanujam, and M. Jarrell, Solving the parquet equations for the Hubbard model beyond weak coupling, Physical Review E **87**, 013311 (2013).

[20] A. Ge, N. Ritz, E. Walter, S. Aguirre, J. von Delft, and F. B. Kugler, Real-frequency quantum field theory applied to the single-impurity Anderson model, Phys. Rev. B **109**, 115128 (2024).

[21] N. Ritz, A. Ge, E. Walter, S. Aguirre, J. von Delft, and F. B. Kugler, KeldyshQFT: A C++ codebase for real-frequency multiloop functional renormalization group and parquet computations of the single-impurity Anderson model, The Journal of Chemical Physics **161**, 054118 (2024).

[22] I. V. Oseledets, Approximation of matrices with logarithmic number of parameters, Doklady Mathematics **80**, 653 (2009).

[23] B. N. Khoromskij, $O(d\log N)$ quantics approximation of N-d tensors in high-dimensional numerical modeling, Constructive Approximation **34**, 257 (2011).

[24] E. Ye and N. F. G. Loureiro, Quantum-inspired method for solving the Vlasov-Poisson equations, Physical Review E **106**, 035208 (2022).

[25] H. Shinaoka, M. Wallerberger, Y. Murakami, K. Nogaki, R. Sakurai, P. Werner, and A. Kauch, Multiscale space-time ansatz for correlation functions of quantum systems based on quantics tensor trains, Phys. Rev. X **13**, 021015 (2023).

[26] N. Jolly, Y. N. Fernández, and X. Waintal, Tensorized orbitals for computational chemistry, (2023), arXiv:2308.03508 [cond-mat.str-el].

[27] M. Środa, K. Inayoshi, H. Shinaoka, and P. Werner, High-resolution nonequilibrium GW calculations based on quantics tensor trains, (2024), arXiv:2412.14032 [cond-mat.str-el].

[28] S. Rohshap, M. K. Ritter, H. Shinaoka, J. von Delft, M. Wallerberger, and A. Kauch, Two-particle calculations with quantics tensor trains: Solving the parquet equations, Phys. Rev. Res. **7**, 023087 (2025).

[29] L. Hölscher, P. Rao, L. Müller, J. Klepsch, A. Luckow, T. Stollenwerk, and F. K. Wilhelm, Quantum-inspired fluid simulation of two-dimensional turbulence with GPU acceleration, Physical Review Research **7**, 013112 (2025).

[30] I. V. Oseledets, Tensor-train decomposition, SIAM Journal on Scientific Computing **33**, 2295 (2011).

[31] D. Savostyanov and I. Oseledets, Fast adaptive interpolation of multi-dimensional arrays in tensor train format, in *The 2011 International Workshop on Multidimensional (nD) Systems* (IEEE, 2011) pp. 1–8.

[32] I. Oseledets and E. Tyrtyshnikov, TT-cross approximation for multidimensional arrays, Linear Algebra and its Applications **432**, 70 (2010).

[33] M. K. Ritter, Y. Núñez Fernández, M. Wallerberger, J. von Delft, H. Shinaoka, and X. Waintal, Quantics tensor cross interpolation for high-resolution parsimonious representations of multivariate functions, Phys. Rev. Lett. **132**, 056501 (2024).

[34] N. Wentzell, G. Li, A. Tagliavini, C. Taranto, G. Rohringer, K. Held, A. Toschi, and S. Andergassen, High-frequency asymptotics of the vertex function: Diagrammatic parametrization and algorithmic implementation, Phys. Rev. B **102**, 085106 (2020).

[35] L. V. Keldysh, Diagram technique for nonequilibrium processes, in *Selected Papers of Leonid V Keldysh*, pp. 47–55.

[36] J. Schwinger, Brownian Motion of a Quantum Oscillator, Journal of Mathematical Physics **2**, 407 (1961).

[37] L. P. Kadanoff and G. A. Baym, *Quantum statistical mechanics* (Benjamin, New York, 1962).

[38] Y. Núñez Fernández, M. K. Ritter, M. Jeannin, J.-W. Li, T. Kloss, T. Louvet, S. Terasaki, O. Parcollet, J. von Delft, H. Shinaoka, and X. Waintal, Learning tensor networks with tensor cross interpolation: new algorithms and libraries, SciPost Phys. **18**, 104 (2025).

[39] C.-W. Ha, Eigenvalues of differentiable positive definite kernels, SIAM Journal on Mathematical Analysis **17**, 415 (1986).

[40] N. Chikano, J. Otsuki, and H. Shinaoka, Performance analysis of a physically constructed orthogonal representation of imaginary-time Green's function, Phys. Rev. B **98**, 035104 (2018).

[41] S. Dirnböck, S.-S. B. Lee, F. B. Kugler, S. Huber, J. von Delft, K. Held, and M. Wallerberger, Overcomplete intermediate representation of two-particle Green's functions and its relation to partial spectral functions, Phys. Rev. Res. **6**, 043228 (2024).

[42] H. Shinaoka, J. Otsuki, M. Ohzeki, and K. Yoshimi, Compressing Green's function using intermediate representation between imaginary-time and real-frequency domains, Phys. Rev. B **96**, 035147 (2017).

[43] M. Wallerberger, H. Shinaoka, and A. Kauch, Solving the Bethe-Salpeter equation with exponential convergence, Phys. Rev. Res. **3**, 033168 (2021).

[44] Tensor4all, `https://tensor4all.org`.

[45] M. Filippone, C. P. Moca, A. Weichselbaum, J. von Delft, and C. Mora, At which magnetic field, exactly, does the Kondo resonance begin to split? A Fermi liquid description of the low-energy properties of the Anderson model, Phys. Rev. B **98**, 075404 (2018).

[46] P. W. Anderson, Localized magnetic states in metals, Phys. Rev. **124**, 41 (1961).

[47] G. Rohringer, A. Valli, and A. Toschi, Local electronic correlation at the two-particle level, Phys. Rev. B **86**, 125114 (2012).

[48] H. Takahashi, R. Sakurai, and H. Shinaoka, Compactness of quantics tensor train representations of local imaginary-time propagators, SciPost Phys. **18**, 007 (2025).

[49] F. D. Picoli, M. Huang, A. Gleis, and J. von Delft, Tangent-Space Krylov Solver for Quantum Impurity Systems, in preparation (2025).

[50] S. G. Jakobs, M. Pletyukhov, and H. Schoeller, Properties of multi-particle Green's and vertex functions within Keldysh formalism, Journal of Physics A: Mathematical and Theoretical **43**, 103001 (2010).

[51] C. J. Eckhardt, C. Honerkamp, K. Held, and A. Kauch, Truncated unity parquet solver, Physical Review B **101**, 155104 (2020).

[52] F. Krien and A. Kauch, The plain and simple parquet approximation: single-and multi-boson exchange in the two-dimensional Hubbard model, The European Physical Journal B **95**, 10.1140/epjb/s10051-022-00329-6 (2022).

[53] G. Li, A. Kauch, P. Pudleiner, and K. Held, The victory project v1.0: An efficient parquet equations solver, Computer Physics Communications **241**, 146 (2019).

[54] S.-S. B. Lee and A. Weichselbaum, Adaptive broadening to improve spectral resolution in the numerical renormalization group, Phys. Rev. B **94**, 235127 (2016).

[55] S.-S. B. Lee, J. von Delft, and A. Weichselbaum, Doublon-holon origin of the subpeaks at the hubbard band edges, Phys. Rev. Lett. **119**, 236402 (2017).

[56] A. Weichselbaum, Non-Abelian symmetries in tensor networks: A quantum symmetry space approach, Annals of Physics **327**, 2972 (2012).

[57] A. Weichselbaum, Tensor networks and the numerical renormalization group, Phys. Rev. B **86**, 245124 (2012).

[58] A. Weichselbaum, X-symbols for non-Abelian symmetries in tensor networks, Phys. Rev. Research **2**, 023385 (2020), arXiv:1910.13736 [cond-mat.str-el].

[59] A. Weichselbaum, QSpace - An open-source tensor library for Abelian and non-Abelian symmetries, SciPost Phys. Codebases , 40 (2024).

[60] A. Weichselbaum, Codebase release 4.0 for QSpace, SciPost Phys. Codebases , 40 (2024).

[61] M. Frankenbach and A. Ge, TCI4Keldysh: A Julia code for computing imaginary- and real- frequency local four-point vertices from mpNRG using QTCI.

[62] Data for: Computing and compressing local vertex functions in real and imaginary frequencies from the multi-point renormalization group using quantics tensor cross interpolation., `https://doi.org/10.5282/ubm/data.613`.

[63] J.-M. Lihm and S.-S. Lee, private communication.

Chapter Five

Conclusion and outlook

5.1 Summary

This thesis has explored two types of approaches for studying quantum many-body systems on a classical computer, those that compress wave functions and those that avoid explicit representation of the wave function. The parquet equations and the closely related fRG belong to the second class, relying on exact relations of quantum many-body correlators. We showed how to apply these approaches to quantum spin systems and how to distinguish both ordered phases and quantum spin liquids. These results are reproducible if the implementation is set up carefully, extensively relying on error-controlled adaptive algorithms.

We then showed how a different type of approach, approximate compression of the wave function, can be generalized to a TT representation of general functions of multiple variables. We showed how a TT for a compressible function is reliably generated by TCI, a active learning algorithm that iteratively optimizes a TT using local updates. Its computational cost scales as $\mathcal{O}(\chi^3\mathcal{L})$. The original TCI algorithm's fragile numerical stability can be improved upon by using a partial rank-revealing LU decomposition to perform the local updates. There are also some variations of TCI tailored for specific purposes. Rook search and block rook search can reduce the number of samples taken during TCI construction. Variants with 1-site or 0-site updates are much cheaper than original TCI, and can be used to restore full nesting or remove spurious pivots that induce numerical instability. Ergodicity issues can be mitigated by global update schemes. Once generated, numerous operations can be performed on a TT in a tensor network fashion with linear cost in the number of variables and polynomial cost in the bond dimension, $\mathcal{O}(\chi^\kappa\mathcal{L})$, where $\kappa \in [2, 4]$. For instance, quadrature, i.e. evaluating an integral, can be performed through factorized sum. This is particularly helpful for oscillating functions, which are difficult to perform using Monte Carlo integration due to the sign problem.

This approach is particularly powerful in combination with the quantics representation, where a function's argument is represented in terms of it binary digits, or bits. Each bit then corresponds to a certain length scale, and factorizing this quantics representation of the function to quantics TT thus factorizes different length scales of the function. Many highly structured functions, for example, sums of exponentials and polynomials are representable at a small bond dimension in this way. Using the well-known Haldane model as an example, we demonstrate how both the propagator and the Berry flux are representable efficiently, easily reaching a resolution of $2^{20} \times 2^{20} \approx 10^6 \times 10^6$ discretization points in the Brillouin zone. For the propagator, the bond dimension scales only weakly with temperature, $\chi \in \mathcal{O}(\sqrt{\beta})$. The berry flux can be summed over efficiently using factorized summation, giving a value for the Chern number that is accurate to 10^{-6}.

Besides integration, other operations that can be performed particularly on a quantics TT using tensor network techniques are the Fourier transform and integrals over the shared variables of two multivariate functions. The quantics Fourier transform is implemented as a partially factorized linear operator, akin to an MPO. Given a specific index ordering, it has a bond dimension of 11 for an accuracy of 10^{-11}. Performing Fourier transforms corresponds to contracting this MPO to the QTT representing the function to be transformed. This operation scales as $\mathcal{O}(\chi^2\mathcal{L}) = \mathcal{O}(\chi^2 \log N)$, where N is the number of data points, which may be faster than the fast Fourier transform for compressible functions. In a similar fashion, coordinate transforms such as rotations can often be expressed as MPO with low bond dimension. Convolutions, or more generally an integral over shared parameters of two multivariate functions, are expressed as MPO-MPO contraction, where each MPO is a reshaped version of a QTT corresponding to each function. If the original functions and the resulting QTT all have bond dimension χ, this operation scales as $\mathcal{O}(\chi^4\mathcal{L})$, which is again linear in $\mathcal{L}$.

This library of QTT operations is sufficiently versatile to allow us to implement entire quantum many-body algorithms within the formalism. In this thesis, we showed an implementation of the parquet equations using this approach. Diagrammatic equations of this kind are a particularly attractive target, since most of the computational effort is expended on integrals over shared frequency and momentum variables in two-particle reducible diagrams. These can be converted to MPO-MPO contractions in the QTT format, which scale only logarithmically with the size of the frequency box. We showed that for frequency boxes that are accessible in a dense grid format, the QTT parquet equations converge with the same speed and accuracy, but consume much less memory. We also showed how to obtain vertices from multipoint NRG, using QTCI to obtain the data directly in QTT format.

5.2 Improving the tensor train toolbox

Currently, the limiting factor in TT-based implementations of algorithms is the computation time required to perform algorithms with scaling $\mathcal{O}(\chi^4)$, most notably MPO-MPO contractions. These are usually performed using a variational optimization algorithm [83], using the result of the so-called zip-up algorithm [84] as initial state. These algorithms were formulated in the context of tensor networks for many-body quantum systems, and existing implementations are optimized for computing wave functions or density matrices. Notably, these algorithms also target a Frobenius norm error, which is not always appropriate in the function representation context [P4]. It is therefore likely that a new implementation optimized for more general applications would improve both computation time and truncation error.

A further speedup could be gained by parallelization of the TCI and MPO-MPO contraction algorithms, for which there are two approaches: One approach is to perform local optimization sweeps on different parts of the TT in parallel, only exchanging information every few sweeps, akin to the parallel DMRG algorithm [85]. An alternative approach is to distribute the information on multiple TT and performing optimization on each TT in parallel. This distribution into multiple TT can be done manually, by constructing separate TT for each discrete value of a subset of the function parameters, and compressing the dependencies on all other parameters. An automatic way of distributing on multiple TT is through an adaptive patching scheme introduced in Ref. [86].

In tensor networks for many-body quantum physics, the site indices are usually uniform, having the same bond dimension and meaning across the whole MPS chain. Established tensor network algorithms therefore assume the same operation is to be applied uniformly, whereas that is often not the intention for TT representing functions. For example, the QTT representing vertices in publication [P4] carries site indices alternating between bosonic frequency bits ω_ℓ and fermionic frequency bits ν_ℓ, ν'_ℓ. Evaluating the Bethe–Salpeter equations requires contraction of ν_ℓ and ν'_ℓ, but elementwise multiplication of ω_ℓ. To avoid a distinct treatment of different tensors in the TT, the tensors carrying ω_ℓ indices were reshaped to a diagonal tensor carrying a duplicated index, such that elementwise multiplication is equivalent to contraction, inducing some numerical overhead. This overhead is the source of some inefficiency, which should be avoided for more challenging applications. To this end, it is necessary to implement more flexible variants of the known tensor network algorithms, with particular emphasis on the MPO-MPO contraction.

Another, more explorative direction is to vary the topology of the tensor network beyond the one-dimensional TT. The TCI algorithm has been generalized to a tree topology by Tindall et al. [87], which resulted in a more efficient representation for the benchmarks tested by them. Efficient contraction algorithms can be formulated on trees as well [71, 88, 89], and it thus seems worth

investigating for which classes of functions an overall gain in efficiency can be expected from a quantics tensor tree implementation of numerical algorithms. More general structures which are not cycle-free, such as a two-dimensional lattice of tensors similar to projected entangled pair states (PEPS) [90], do not permit a canonical form comprised entirely of isometry tensors. These canonical forms are necessary to formulate efficient contraction algorithms [7]. A generalization of the tensor network format beyond trees is therefore unlikely to yield further performance improvements.

In total, these improvements to the implementation on a technical level, combined with a clean and easy-to-use programming interface, would facilitate widespread use of tensor networks for function representations in many different fields of computational physics. Some algorithms have already reached sufficient reliability to be used as black boxes. In the long term, as available libraries mature, tensor network representations may become a standard approach to increasing efficiency, particularly in the fields of hydrodynamics, quantum chemistry, as well as correlator-based methods for quantum many-body systems.

5.3 Future applications of tensor trains

Even when only considering correlator-based methods, the new computational infrastructure of TT representations opens up a wide field of possible applications. Full parameterization of all frequency and momentum dependencies of the vertex was previously unfeasible due to excessive computational cost, as were multi-orbital models. All of this is now in reach of the parquet scheme that uses a QTT parameterization of the vertex, or possibly a combination of multiple QTT. The implementation in this work was entirely based on the imaginary time Matsubara formalism, which requires an inherently ill-defined analytic continuation step to obtain real-frequency quantities that could be observed in experiment. A fundamental improvement would be to implement the parquet equations or a related diagrammatic method, such as the finite-difference parquet scheme [37], in the real-frequency Keldysh formalism. This allows for direct calculation of observables without an ill-defined analtic continuation step, as well as generalizing the scope to non-equilibrium settings [56]. Real-frequency correlators in QTT format were used in independent work to implement Dyson's equation [29], the non-crossing approximation in a strong-coupling expansion [82], and the *GW* formalism [31].

Another method that is currently limited in memory consumption and would benefit from QTT representation of correlators is DMFT. As we showed in Ref. [P5], vertices can be extracted directly in QTT format from an NRG impurity solver in the multipoint NRG scheme [35, 36, 91]. An obvious next step would be to combine this output with the parquet equations in a DΓA scheme.

These developments all contribute to closing a gap between the necessary simplifications made in computational methods in quantum many-body physics and the complexity of real materials in experiment. Often, to make a model treatable with high-precision numerical methods, it must be simplified considerably. The paradigmatic model of high-temperature superconducting cuprates, for instance, is the single-band Hubbard model [92]. Despite this, the model was found to show no superconductivity in its pure form; it only becomes superconducting with the addition of further terms [93]. A more recent study by Jiang et al. [94] found that when downfolding a slightly more realistic three-band model to a single-band Hubbard model, an additional density-assisted hopping term appears. This serves to illustrate that studying single-band models may not always be sufficient to explain phenomena observed in experiment. On the other hand, multi-orbital models are often out of reach for more precise computational methods, and one has to fall back to cheaper approximations. There is thus a tradeoff between completeness of the model and precision of the method, which can be alleviated by new technical developments such as the tensor trains presented in this work.

Appendix A

Conventions

This section gives a brief overview over the notation used in this thesis.

A.1 Acronyms

Acronyms used in this thesis are listed in alphabetical order in Table A.1

Table A.1: Acronyms used in this thesis, in alphabetical order.

BSE	Bethe–Salpeter equation
BZ	Brillouin zone
CI	cross interpolation → matrix cross interpolation
DMFT	dynamical mean-field theory
DMRG	density matrix renormalization group
DΓA	dynamical vertex approximation
FFT	fast Fourier transform
fRG	functional renormalization group
FT	Fourier transform
GK	Gauss–Kronrod quadrature
IR	infrared, i.e. low energy
LU	LU decomposition, i.e. factorization of a matrix into a lower triangular matrix L and an upper triangular matrix U
MCI	matrix cross interpolation
mfRG	multiloop functional renormalization group
MPO	matrix product operator

Continued on next page

Table A.1: Acronyms used in this thesis, in alphabetical order. (Continued)

MPS	matrix product state
ODE	ordinary differential equation
PEPS	projected entangled pair states
pffRG	pseudofermion functional renormalization group
prrLU	partial rank-revealing LU decomposition
QFT	quantum Fourier transform, or quantics Fourier transform
QR	QR decomposition, i.e. factorization of a matrix into a unitary matrix Q and an upper triangular matrix R
QTCI	quantics tensor cross interpolation
QTT	quantics tensor train
SDE	Schwinger–Dyson equation
SIAM	single-impurity Anderson model
SVD	singular value decomposition
TCI	tensor cross interpolation
TT	tensor train
UV	ultraviolet, i.e. high energy

A.2 Table of symbols

In some cases, the publications that are part of this thesis use different symbols for the same quantities, or adhere to different conventions amongst those common in the field. These are listed in Table A.2 below.

Table A.2: Symbols used for various quantitites in each publication that is part of this thesis.

[P1]	[P2]	[P3]	[P4]	[P5]	this thesis
Parquet diagrammatics and fRG					
bosonic Matsubara frequencies					
ω			ω	ω	
fermionic Matsubara frequencies					
ν, ν'	$i\omega$		ν, ν'	ν, ν'	
two-point Green's function					
$G(\nu)$	$G(\mathbf{k}, i\omega)$	G	$G_\sigma(\nu)$	G	$G(1', 1)$

Continued on next page

Table A.2: Symbols used for various quantitites in each publication that is part of this thesis. (Continued)

[P1]	[P2]	[P3]	[P4]	[P5]	this thesis
self energy					
$\Sigma(\nu)$		Σ	$\Sigma_\sigma(\nu)$	Σ	$\Sigma(1', 1)$
pair propagator					
$(G \times G)$			$\chi^{\nu\nu'\omega}_{0,ph/pp}$	Π	
(full) two-particle vertex					
$\Gamma(\omega, \nu, \nu')$			$F^{\nu\nu'\omega}_{\sigma\sigma'}$	$\Gamma(\omega, \nu, \nu')$	$\Gamma(1', 2'; 1, 2)$
r-channel reducible two-particle vertex					
$\gamma_r(\omega_r, \nu_r, \nu'_r)$			$\Phi^{\nu\nu'\omega}_{\sigma\sigma'}$		$\gamma_r(1', 2'; 1, 2)$
r-channel irreducible two-particle vertex					
			$\Gamma^{\nu\nu'\omega}_{\sigma\sigma'}$		I_r
(fully) irreducible two-particle vertex					
			$\Lambda^{\nu\nu'\omega}$		R
fRG cutoff parameter					
Λ					Λ
Tensor trains (TT) and tensor cross interpolation (TCI)					
tensor train representation of F					
	F^{QTCI}	$\tilde{F}$		F^{QTCI}	
bond index					
	ℓ	ℓ	ℓ	ℓ	
length of tensor train					
	$\mathcal{L}$	$\mathcal{L}$	L	L	
general index on bond ℓ					
	α_ℓ	a_ℓ	α_ℓ	α_ℓ	
index on bond ℓ in CI-canonical form					
	α_ℓ, β_ℓ	$i_\ell, j_{\ell+1}$	α_ℓ		
index set on bond ℓ in CI-canonical form					
	$\mathcal{I}_\ell, \mathcal{J}_\ell$	$\mathcal{I}_\ell, \mathcal{J}_{\ell+1}$			

Continued on next page

Table A.2: Symbols used for various quantitites in each publication that is part of this thesis. (Continued)

[P1]	[P2]	[P3]	[P4]	[P5]	this thesis
tensor in tensor train					
	$[M_\ell]^{\sigma_\ell}_{\alpha_{\ell-1}\alpha_\ell}$	$[M_\ell]^{\sigma_\ell}_{a_{\ell-1}a_\ell}$, $[M_\ell^{\sigma_\ell}]_{\alpha_{\ell-1}\alpha_\ell}$	$[M_\ell]^{\sigma_\ell}_{\alpha_{\ell-1}\alpha_\ell}$	$[M_\ell^{\sigma_\ell}]_{\alpha_{\ell-1}\alpha_\ell}$	$[M_\ell^{\sigma_\ell}]_{\alpha_{\ell-1}\alpha_\ell}$
local bond dimension					
	d	d, d_ℓ			d
bond dimension on bond ℓ					
	D_ℓ	χ_ℓ	D_ℓ	χ_ℓ	
maximum bond dimension					
	$D_{\max}$	χ	$D_{\max}$	χ	χ
(estimated) error					
	ε	ϵ	ϵ	ε	
error tolerance					
	ϵ	τ	ϵ	τ	
Quantics representation					
number of original variables					
	n	$\mathcal{N}$	N		
number of quantics bits per variable					
	R	$\mathcal{R}$	R	R	
original variable					
	u_i	x_n		ω	
linear index					
		m	m	m	
quantics index					
	σ_ℓ	σ_ℓ	σ_ℓ, $\nu_\ell, \nu_\ell', \omega_\ell$	σ_ℓ	σ_ℓ
tuple of all quantics indices					
	$\boldsymbol{\sigma}$	$\boldsymbol{\sigma}$		$\boldsymbol{\sigma}$	

Bibliography

[P1] M. K. Ritter, D. Kiese, T. Müller, F. B. Kugler, R. Thomale, S. Trebst, and J. von Delft, "Benchmark calculations of multiloop pseudofermion fRG", The European Physical Journal B **95**, 102 (2022).

[P2] M. K. Ritter, Y. Núñez Fernández, M. Wallerberger, J. von Delft, H. Shinaoka, and X. Waintal, "Quantics tensor cross interpolation for high-resolution parsimonious representations of multivariate functions", Physical Review Letters **132**, 056501 (2024).

[P3] Y. Núñez Fernández, M. K. Ritter, M. Jeannin, J.-W. Li, T. Kloss, T. Louvet, S. Terasaki, O. Parcollet, J. von Delft, H. Shinaoka, and X. Waintal, "Learning tensor networks with tensor cross interpolation: new algorithms and libraries", SciPost Physics **18**, 104 (2025).

[P4] S. Rohshap, M. K. Ritter, H. Shinaoka, J. von Delft, M. Wallerberger, and A. Kauch, "Two-particle calculations with quantics tensor trains: solving the parquet equations", Physical Review Research **7**, 023087 (2025).

[P5] M. Frankenbach, M. K. Ritter, M. Pelz, N. Ritz, J. v. Delft, and A. Ge, "Computing and compressing local vertex functions in imaginary and real frequencies from the multipoint numerical renormalization group using quantics tensor cross interpolation", June 16, 2025, arXiv:2506.13359.

[1] P. W. Anderson, "More is different", Science **177**, 393 (1972).

[2] L. Savary and L. Balents, "Quantum spin liquids: a review", Reports on Progress in Physics **80**, 016502 (2017).

[3] T. Ayral, "Classical and quantum algorithms for many-body problems", Comptes Rendus. Physique **26**, 25 (2025).

[4] M. Kawamura, K. Yoshimi, T. Misawa, Y. Yamaji, S. Todo, and N. Kawashima, "Quantum lattice model solver $H\Phi$", Computer Physics Communications **217**, 180 (2017).

[5] A. Wietek and A. M. Läuchli, "Sublattice coding algorithm and distributed memory parallelization for large-scale exact diagonalizations of quantum many-body systems", Physical Review E **98**, 033309 (2018).

[6] J. Eisert, M. Cramer, and M. B. Plenio, "Colloquium: area laws for the entanglement entropy", Reviews of Modern Physics **82**, 277 (2010).

[7] R. Orús, "A practical introduction to tensor networks: matrix product states and projected entangled pair states", Annals of Physics **349**, 117 (2014).

[8] U. Schollwöck, "The density-matrix renormalization group in the age of matrix product states", Annals of Physics, January 2011 Special Issue **326**, 96 (2011).

[9] P. Henelius and A. W. Sandvik, "Sign problem in Monte Carlo simulations of frustrated quantum spin systems", Physical Review B **62**, 1102 (2000).

[10] M. Troyer and U.-J. Wiese, "Computational complexity and fundamental limitations to fermionic quantum Monte Carlo simulations", Physical Review Letters **94**, 170201 (2005).

[11] N. E. Bickers, "Self-consistent many-body theory for condensed matter systems", *Theoretical methods for strongly correlated electrons*, edited by D. Sénéchal, A.-M. Tremblay, and C. Bourbonnais (Springer-Verlag, New York, 2004), pp. 237–296.

[12] B. Roulet, J. Gavoret, and P. Nozières, "Singularities in the X-ray absorption and emission of metals. I. First-order parquet calculation", Physical Review **178**, 1072 (1969).

[13] W. Metzner, M. Salmhofer, C. Honerkamp, V. Meden, and K. Schönhammer, "Functional renormalization group approach to correlated fermion systems", Reviews of Modern Physics **84**, 299 (2012).

[14] C. Wetterich, "Exact evolution equation for the effective potential", Physics Letters B **301**, 90 (1993).

[15] S. R. White, "Density matrix formulation for quantum renormalization groups", Physical Review Letters **69**, 2863 (1992).

[16] S. R. White, "Density-matrix algorithms for quantum renormalization groups", Physical Review B **48**, 10345 (1993).

[17] I. Oseledets and E. Tyrtyshnikov, "TT-cross approximation for multidimensional arrays", Linear Algebra and its Applications **432**, 70 (2010).

[18] B. N. Khoromskij, "$O(d \log N)$-quantics approximation of N-d tensors in high-dimensional numerical modeling", Constructive Approximation **34**, 257 (2011).

[19] Y. Núñez Fernández, M. Jeannin, P. T. Dumitrescu, T. Kloss, J. Kaye, O. Parcollet, and X. Waintal, "Learning Feynman diagrams with tensor trains", Physical Review X **12**, 041018 (2022).

[20] I. V. Oseledets, "Approximation of matrices with logarithmic number of parameters", Doklady Mathematics **80**, 653 (2009).

[21] D. Savostyanov and I. Oseledets, "Fast adaptive interpolation of multi-dimensional arrays in tensor train format", The 2011 international workshop on multidimensional (nD) systems (Sept. 2011), pp. 1–8.

[22] D. V. Savostyanov, "Quasioptimality of maximum-volume cross interpolation of tensors", Linear Algebra and its Applications **458**, 217 (2014).

[23] N. Gourianov, "Exploiting the structure of turbulence with tensor networks", PhD thesis (University of Oxford, 2022).

[24] N. Gourianov, P. Givi, D. Jaksch, and S. B. Pope, "Tensor networks enable the calculation of turbulence probability distributions", Science Advances **11**, eads5990 (2025).

[25] L. Hölscher, P. Rao, L. Müller, J. Klepsch, A. Luckow, T. Stollenwerk, and F. K. Wilhelm, "Quantum-inspired fluid simulation of two-dimensional turbulence with GPU acceleration", Physical Review Research **7**, 013112 (2025).

[26] R. D. Peddinti, S. Pisoni, A. Marini, P. Lott, H. Argentieri, E. Tiunov, and L. Aolita, "Quantum-inspired framework for computational fluid dynamics", Communications Physics **7**, 1 (2024).

[27] R. Sakurai, H. Takahashi, and K. Miyamoto, "Learning parameter dependence for Fourier-based option pricing with tensor trains", June 24, 2024, arXiv:2405.00701.

[28] N. Jolly, Y. N. Fernández, and X. Waintal, "Tensorized orbitals for computational chemistry", Aug. 7, 2023, arXiv:2308.03508.

[29] M. Murray, H. Shinaoka, and P. Werner, "Nonequilibrium diagrammatic many-body simulations with quantics tensor trains", Physical Review B **109**, 165135 (2024).

[30] H. Shinaoka, M. Wallerberger, Y. Murakami, K. Nogaki, R. Sakurai, P. Werner, and A. Kauch, "Multiscale space-time ansatz for correlation functions of quantum systems based on quantics tensor trains", Physical Review X **13**, 021015 (2023).

[31] M. Środa, K. Inayoshi, H. Shinaoka, and P. Werner, "High-resolution nonequilibrium *GW* calculations based on quantics tensor trains", Dec. 18, 2024, arXiv:2412.14032.

[32] H. Takahashi, R. Sakurai, and H. Shinaoka, "Compactness of quantics tensor train representations of local imaginary-time propagators", SciPost Physics **18**, 007 (2025).

[33] G. Rohringer, H. Hafermann, A. Toschi, A. A. Katanin, A. E. Antipov, M. I. Katsnelson, A. I. Lichtenstein, A. N. Rubtsov, and K. Held, "Diagrammatic routes to nonlocal correlations beyond dynamical mean field theory", Reviews of Modern Physics **90**, 025003 (2018).

[34] A. Toschi, A. A. Katanin, and K. Held, "Dynamical vertex approximation: a step beyond dynamical mean-field theory", Physical Review B **75**, 045118 (2007).

[35] F. B. Kugler, S.-S. B. Lee, and J. von Delft, "Multipoint correlation functions: spectral representation and numerical evaluation", Physical Review X **11**, 041006 (2021).

[36] S.-S. B. Lee, F. B. Kugler, and J. von Delft, "Computing local multipoint correlators using the numerical renormalization group", Physical Review X **11**, 041007 (2021).

[37] J.-M. Lihm, D. Kiese, S.-S. B. Lee, and F. B. Kugler, "The finite-difference parquet method: enhanced electron-paramagnon scattering opens a pseudogap", May 26, 2025, arXiv:2505.20116.

[38] M. Gievers, "Functional approaches to Fermi polarons in cold atomic gases and solid-state systems", PhD thesis (Ludwig-Maximilians-Universität, Munich, Jan. 9, 2025), 280 pp.

[39] G. Rohringer, A. Valli, and A. Toschi, "Local electronic correlation at the two-particle level", Physical Review B **86**, 125114 (2012).

[40] M. Gievers, E. Walter, A. Ge, J. von Delft, and F. B. Kugler, "Multiloop flow equations for single-boson exchange fRG", The European Physical Journal B **95**, 108 (2022).

[41] K. Held, A. A. Katanin, and A. Toschi, "Dynamical vertex approximation: an introduction", Progress of Theoretical Physics Supplement **176**, 117 (2008).

[42] S. Biermann, F. Aryasetiawan, and A. Georges, "First-principles approach to the electronic structure of strongly correlated systems: combining the GW approximation and dynamical mean-field theory", Physical Review Letters **90**, 086402 (2003).

[43] E. Kozik, M. Ferrero, and A. Georges, "Nonexistence of the Luttinger-Ward functional and misleading convergence of skeleton diagrammatic series for Hubbard-like models", Physical Review Letters **114**, 156402 (2015).

[44] H. Eßl, M. Reitner, E. Kozik, and A. Toschi, "How to stay on the physical branch in self-consistent many-electron approaches", Feb. 3, 2025, arXiv:2502.01420.

[45] F. B. Kugler and J. von Delft, "Multiloop functional renormalization group for general models", Physical Review B **97**, 035162 (2018).

[46] F. B. Kugler and J. von Delft, "Multiloop functional renormalization group that sums up all parquet diagrams", Physical Review Letters **120**, 57403 (2018).

[47] F. L. Buessen, "The SpinParser software for pseudofermion functional renormalization group calculations on quantum magnets", SciPost Physics Codebases, 005 (2022).

[48] C. Honerkamp and M. Salmhofer, "Temperature-flow renormalization group and the competition between superconductivity and ferromagnetism", Physical Review B **64**, 184516 (2001).

[49] B. Schneider, J. Reuther, M. G. Gonzalez, B. Sbierski, and N. Niggemann, "Temperature flow in pseudo-Majorana functional renormalization for quantum spins", Physical Review B **109**, 195109 (2024).

[50] C. Taranto, S. Andergassen, J. Bauer, K. Held, A. Katanin, W. Metzner, G. Rohringer, and A. Toschi, "From infinite to two dimensions through the functional renormalization group", Physical Review Letters **112**, 196402 (2014).

[51] F. B. Kugler and J. v. Delft, "Derivation of exact flow equations from the self-consistent parquet relations", New Journal of Physics **20**, 123029 (2018).

[52] O. Benton, L. D. C. Jaubert, R. R. P. Singh, J. Oitmaa, and N. Shannon, "Quantum spin ice with frustrated transverse exchange: from a π-flux phase to a nematic quantum spin liquid", Physical Review Letters **121**, 067201 (2018).

[53] Y. Iqbal, P. Ghosh, R. Narayanan, B. Kumar, J. Reuther, and R. Thomale, "Intertwined nematic orders in a frustrated ferromagnet", **94**, 1 (2016).

[54] M. Salmhofer, C. Honerkamp, W. Metzner, and O. Lauscher, "Renormalization group flows into phases with broken symmetry", Progress of Theoretical Physics **112**, 943 (2004).

[55] P. M. Bonetti, "Accessing the ordered phase of correlated Fermi systems: vertex bosonization and mean-field theory within the functional renormalization group", Physical Review B **102**, 235160 (2020).

[56] A. Ge, N. Ritz, E. Walter, S. Aguirre, J. von Delft, and F. B. Kugler, "Real-frequency quantum field theory applied to the single-impurity Anderson model", Physical Review B **109**, 115128 (2024).

[57] J. Diekmann and S. G. Jakobs, "Parquet approximation and one-loop renormalization group: equivalence on the leading-logarithmic level", Physical Review B **103**, 155156 (2021).

[58] J. Reuther and P. Wölfle, "J_1-J_2 frustrated two-dimensional Heisenberg model: random phase approximation and functional renormalization group", Physical Review B **81**, 144410 (2010).

[59] T. Müller, D. Kiese, N. Niggemann, B. Sbierski, J. Reuther, S. Trebst, R. Thomale, and Y. Iqbal, "Pseudo-fermion functional renormalization group for spin models", Reports on Progress in Physics **87**, 036501 (2024).

[60] J. Krieg and P. Kopietz, "Exact renormalization group for quantum spin systems", Physical Review B **99**, 060403 (2019).

[61] A. A. Abrikosov, "Electron scattering on magnetic impurities in metals and anomalous resistivity effects", Physics Physique Fizika **2**, 5 (1965).

[62] V. N. Popov and S. A. Fedotov, "The functional integration method and diagram technique for spin systems", Journal of Experimental and Theoretical Physics **67**, 535 (1988).

[63] N. V. Prokof'ev and B. V. Svistunov, "From the Popov-Fedotov case to universal fermionization", Physical Review B **84**, 073102 (2011).

[64] B. Schneider, D. Kiese, and B. Sbierski, "Taming pseudofermion functional renormalization for quantum spins: finite temperatures and the Popov-Fedotov trick", Physical Review B **106**, 235113 (2022).

[65] J. Thoenniss, M. K. Ritter, F. B. Kugler, J. von Delft, and M. Punk, "Multiloop pseudofermion functional renormalization for quantum spin systems: application to the spin-$\frac{1}{2}$ kagome Heisenberg model", 2020, arXiv:2011.01268.

[66] N. Niggemann, B. Sbierski, and J. Reuther, "Frustrated quantum spins at finite temperature: pseudo-Majorana functional renormalization group approach", Physical Review B **103**, 104431 (2021).

[67] M. K. Ritter, "Multiloop pseudofermion functional renormalization group study of the pyrochlore XXZ model", master's thesis (Ludwig-Maximilians-Universität München, 2021).

[68] G. Günal, "Multiloop pseudofermion functional renormalization group study of the Heisenberg model with dipolar interactions on the triangular and square lattice", master's thesis (Ludwig-Maximilians-Universität, Munich, May 22, 2023).

[69] D. Kiese, T. Müller, Y. Iqbal, R. Thomale, and S. Trebst, "Multiloop functional renormalization group approach to quantum spin systems", Physical Review Research **4**, 023185 (2022).

[70] B. Schneider and B. Sbierski, "Taming spin susceptibilities in frustrated quantum magnets: mean-field form and approximate nature of the quantum-to-classical correspondence", Physical Review Letters **134**, 176502 (2025).

[71] M. C. Bañuls, "Tensor network algorithms: a route map", Annual Review of Condensed Matter Physics **14**, 173 (2023).

[72] Q. Mortier, L. Devos, L. Burgelman, B. Vanhecke, N. Bultinck, F. Verstraete, J. Haegeman, and L. Vanderstraeten, "Fermionic tensor network methods", SciPost Physics **18**, 012 (2025).

[73] A. Gleis, J.-W. Li, and J. von Delft, "Controlled bond expansion for density matrix renormalization group ground state search at single-site costs", Physical Review Letters **130**, 246402 (2023).

[74] M. Wallerberger, H. Shinaoka, and A. Kauch, "Solving the Bethe-Salpeter equation with exponential convergence", Physical Review Research **3**, 033168 (2021).

[75] H. Shinaoka, D. Geffroy, M. Wallerberger, J. Otsuki, K. Yoshimi, E. Gull, and J. Kuneš, "Sparse sampling and tensor network representation of two-particle Green's functions", SciPost Physics **8**, 012 (2020).

[76] H. Shinaoka, J. Otsuki, K. Haule, M. Wallerberger, E. Gull, K. Yoshimi, and M. Ohzeki, "Overcomplete compact representation of two-particle Green's functions", Physical Review B **97**, 205111 (2018).

[77] J. Kaye, K. Chen, and O. Parcollet, "Discrete Lehmann representation of imaginary time Green's functions", Physical Review B **105**, 235115 (2022).

[78] D. Kiese, H. U. R. Strand, K. Chen, N. Wentzell, O. Parcollet, and J. Kaye, "Discrete Lehmann representation of three-point functions", Physical Review B **111**, 035135 (2025).

[79] C. J. Eckhardt, G. A. H. Schober, J. Ehrlich, and C. Honerkamp, "Truncated-unity parquet equations: application to the repulsive Hubbard model", Physical Review B **98**, 075143 (2018).

[80] J. Lichtenstein, D. S. de la Peña, D. Rohe, E. Di Napoli, C. Honerkamp, and S. A. Maier, "High-performance functional renormalization group calculations for interacting fermions", Computer Physics Communications **213**, 100 (2017).

[81] C. Hille, F. B. Kugler, C. J. Eckhardt, Y.-Y. He, A. Kauch, C. Honerkamp, A. Toschi, and S. Andergassen, "Quantitative functional renormalization group description of the two-dimensional Hubbard model", Physical Review Research **2**, 033372 (2020).

[82] M. Eckstein, "Solving quantum impurity models in the non-equilibrium steady state with tensor trains", Oct. 25, 2024, arXiv:2410.19707.

[83] B.-B. Chen, L. Chen, Z. Chen, W. Li, and A. Weichselbaum, "Exponential thermal tensor network approach for quantum lattice models", Physical Review X **8**, 031082 (2018).

[84] E. M. Stoudenmire and S. R. White, "Minimally entangled typical thermal state algorithms", New Journal of Physics **12**, 055026 (2010).

[85] E. M. Stoudenmire and S. R. White, "Real-space parallel density matrix renormalization group", Physical Review B **87**, 155137 (2013).

[86] G. Grosso, "Efficient tensor compression through adaptive patched quantics tensor cross interpolation", master's thesis (Ludwig-Maximilians-Universität and Technische Universität München, Munich, June 2, 2025).

[87] J. Tindall, M. Stoudenmire, and R. Levy, "Compressing multivariate functions with tree tensor networks", Oct. 4, 2024, arXiv:2410.03572.

[88] N. Chepiga and S. R. White, "Comb tensor networks", Physical Review B **99**, 235426 (2019).

[89] Y.-Y. Shi, L.-M. Duan, and G. Vidal, "Classical simulation of quantum many-body systems with a tree tensor network", Physical Review A **74**, 022320 (2006).

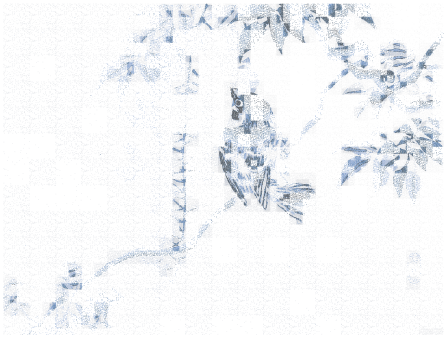
[90] F. Verstraete and J. I. Cirac, "Renormalization algorithms for quantum-many body systems in two and higher dimensions", July 2, 2004, arXiv:cond-mat/0407066.

[91] J.-M. Lihm, J. Halbinger, J. Shim, J. von Delft, F. B. Kugler, and S.-S. B. Lee, "Symmetric improved estimators for multipoint vertex functions", Physical Review B **109**, 125138 (2024).

[92] M. Qin, T. Schäfer, S. Andergassen, P. Corboz, and E. Gull, "The Hubbard model: a computational perspective", Annual Review of Condensed Matter Physics **13**, 275 (2022).

[93] Simons Collaboration on the Many-Electron Problem, M. Qin, C.-M. Chung, H. Shi, E. Vitali, C. Hubig, U. Schollwöck, S. R. White, and S. Zhang, "Absence of superconductivity in the pure two-dimensional Hubbard model", Physical Review X **10**, 031016 (2020).

[94] S. Jiang, D. J. Scalapino, and S. R. White, "Density matrix renormalization group based downfolding of the three-band Hubbard model: importance of density-assisted hopping", Physical Review B **108**, L161111 (2023).



Cover page QTT-compression of a grayscale image, truncated to bond dimension $\chi = 200$ in CI-canonical form, with only the pivot points shown. Original artwork: Kōno Bairei (幸野楳嶺), Sparrow on a wisteria branch. 1859–1895. Ink and color on paper, 18 cm × 23.6 cm. Minneapolis Institute of Art. P.77.27.7. https://collections.artsmia.org/art/42443/sparrow-on-a-wisteria-branch-kono-bairei.