HAL
open science

Learning Feynman Diagrams with Tensor Trains

Yuriel Nunez-Fernandez, Matthieu Jeannin, Philipp T. Dumitrescu, Thomas Kloss, Jason Kaye, Olivier Parcollet, Xavier Waintal

To cite this version:

Yuriel Nunez-Fernandez, Matthieu Jeannin, Philipp T. Dumitrescu, Thomas Kloss, Jason Kaye, et al.. Learning Feynman Diagrams with Tensor Trains. Physical Review X, 2022, 12 (4), pp.041018. 10.1103/PhysRevX.12.041018 . hal-03752036

HAL Id: hal-03752036

https://hal.science/hal-03752036v1

Submitted on 25 May 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Learning Feynman Diagrams with Tensor Trains

Yuriel Núñez Fernández,[1,*] Matthieu Jeannin,[1] Philipp T. Dumitrescu,[2] Thomas Kloss,[1,3] Jason Kaye,[2,4] Olivier Parcollet,[2,5] and Xavier Waintal[1,†]

[1]*Université Grenoble Alpes, CEA, Grenoble INP, IRIG, Pheliqs, F-38000 Grenoble, France*
[2]*Center for Computational Quantum Physics, Flatiron Institute, 162 5th Avenue, New York, New York 10010, USA*
[3]*Université Grenoble Alpes, CNRS, Institut Néel, 38000 Grenoble, France*
[4]*Center for Computational Mathematics, Flatiron Institute, 162 5th Avenue, New York, New York 10010, USA*
[5]*Université Paris-Saclay, CNRS, CEA, Institut de physique théorique, 91191, Gif-sur-Yvette, France*

(Received 13 July 2022; revised 5 September 2022; accepted 8 September 2022; published 16 November 2022)

We use tensor network techniques to obtain high-order perturbative diagrammatic expansions for the quantum many-body problem at very high precision. The approach is based on a tensor train parsimonious representation of the sum of all Feynman diagrams, obtained in a controlled and accurate way with the tensor cross interpolation algorithm. It yields the full time evolution of physical quantities in the presence of any arbitrary time-dependent interaction. Our benchmarks on the Anderson quantum impurity problem, within the real-time nonequilibrium Schwinger-Keldysh formalism, demonstrate that this technique supersedes diagrammatic quantum Monte Carlo by orders of magnitude in precision and speed, with convergence rates $1/N^2$ or faster, where N is the number of function evaluations. The method also works in parameter regimes characterized by strongly oscillatory integrals in high dimension, which suffer from a catastrophic sign problem in quantum Monte Carlo calculations. Finally, we also present two exploratory studies showing that the technique generalizes to more complex situations: a double quantum dot and a single impurity embedded in a two-dimensional lattice.

DOI: 10.1103/PhysRevX.12.041018

Subject Areas: Computational Physics, Condensed Matter Physics, Mesoscopics, Strongly Correlated Materials

I. INTRODUCTION

Many important problems in physics can be formally solved by expressing physical quantities as sums or integrals in high-dimensional spaces, e.g., equilibrium partition functions in condensed matter and statistical physics or high-order perturbative diagrammatic expansions in field theories and in the quantum many-body problem. Calculating integrals in high dimensions is, however, notoriously difficult. Quantum Monte Carlo algorithms have emerged as a class of numerical methods of choice for such problems and have been tremendously successful in many situations [1–5]. They have, nevertheless, well-known major shortcomings. First, as sampling methods, they can become exponentially inefficient due to massive cancellations, a set of related phenomena famously known as the "sign problem," which typically becomes exponentially more severe at low temperatures and for large systems. Second, as stochastic methods, they have an intrinsically slow convergence (as $1/\sqrt{N}$, where N is the number of independent samples), which can severely limit the accuracy of calculations. In fact, overcoming the apparent exponential complexity of the fermionic quantum many-body problem is one of the main motivations for the development of full-scale quantum computers.

Parsimonious (or compressed) representations of high-dimensional functions based on tensor trains, and more generally on a low-rank tensor network (TN) [6–11], offer another route to compute such large-dimensional integrals. Indeed, they provide an effective separation of variables that reduces the calculation of high-dimensional integrals to the evaluation of a set of one-dimensional integrals, a much simpler problem [7]. The *tensor cross interpolation* (TCI) formula [12–14] is an algorithmically efficient way to obtain such a representation, in time scaling polynomially with the dimension. It is a generalization to tensors of cross

*yurielnf@gmail.com
†xavier.waintal@cea.fr

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

interpolation for matrices [15–17] and is closely related to the interpolative decomposition [18].

The subject of this article is the replacement of Monte Carlo sampling by tensor network-based algorithms such as TCI in some many-body algorithms, in particular, diagrammatic quantum Monte Carlo. We emphasize that this use of tensor networks is radically different from their original application in the density matrix renormalization group (DMRG) algorithm [19] and its descendants, where it is used as a variational ansatz for the many-body wave function. Here, we use tensor network representations for the many-body correlation functions arising in the context of Feynman diagram expansions. Like for tensor train applications in machine learning, e.g., [20], we use tensor trains (also known as matrix product states) to *learn*, in a controlled manner, the function representing the sum of Feynman diagrams.

Diagrammatic quantum Monte Carlo methods, i.e., high-order diagrammatic perturbation expansions in powers of the interaction strength, are natural candidates for tensor network techniques. Despite their perturbative nature, when properly combined with resummation techniques and judiciously chosen (field-theory) counterterms [21–34], diagrammatic expansions have been successfully used to explore physics far beyond weak coupling. This includes the Kondo regime of a quantum dot [29,30,33,35], the pseudogap regime of the Hubbard model [36], and low-density electron gas [28,34]. They are particularly useful in nonequilibrium settings [25,30,33], for which there are very few accurate methods available. Computing the expansion coefficient at order n involves, at minimum, computing n-dimensional integrals over time, as well integrals or sums over other dimensions, and the different Feynman diagrams themselves. Since the formulation of perturbation theory as a stochastic sampling over $n!$ Feynman diagrams [21], there has been an effort to reformulate the problem and develop new algorithms for the coefficients in the perturbation series [25,27,33]. Despite major advances, the integration techniques used thus far have been variations of sampling from a non-negative probability distribution. These techniques inevitably suffer from a sign problem for very oscillatory integrals. Rapidly oscillating integrals are encountered especially often in the real-time Schwinger-Keldysh formalism [25,30,33]. We note that, among the quantum Monte Carlo algorithms, diagrammatic Monte Carlo typically manipulates the integrals with lowest dimensions, since the complexity of the calculation of the sum of Feynman diagrams grows exponentially with n [typically as $O(2^n)$] [25]. Hence, they are natural first candidates for a tensor network approach to integration.

In this paper, we explore the use of TCI for real-time nonequilibrium Schwinger-Keldysh perturbation expansions up to high order $n \sim 30$ and high precision. We apply the tensor decomposition to the bare Keldysh n-body correlators appearing in Feynman integrals. We demonstrate very fast convergence, as fast as $O(1/N^2)$ in the number N of integration points. The final precision is limited in practice only by machine precision and rounding errors, something usually out of reach in Monte Carlo calculations.

The main observation underlying our results is that the n-body Keldysh correlators we consider are well approximated by a low-rank tensor train when viewed as functions of n time differences. We refer to this property as "ϵ-factorizability." The ϵ-factorizability property yields a separation of variables which reduces the high-dimensional integrals to a sequence of one-dimensional integrals which can be computed rapidly. Crucially, this ϵ-factorizability persists even in parameter regimes in which the integrands are highly oscillatory. This renders the approach largely immune to the sign problem, which is reduced to the problem of integrating oscillatory functions of a single variable. Finally, the tensor train representation of the n-body correlator directly provides the full time dependence of the observable for an arbitrary time-dependent interaction coupling strength with a costless postprocessing step.

The outline of this paper is as follows. In Sec. II, we summarize our approach and present some illustrative numerical results showcasing its efficiency. Section III reviews the TCI method and can be read independently from the rest of the article. Section IV A gives a concise introduction to the many-body Keldysh formalism and the notations used to compute high-order perturbative expansions. In Sec. V, we adapt the TCI method in Sec. III to calculate the high-order expansion presented in Sec. IV A. We refer to this technique as *tensor train diagrammatics (TTD)*. Section VI presents some numerical results on TTD for calculating properties of the single-impurity Anderson model (SIAM). Section VII shows results beyond SIAM for an impurity embedded in a 2D lattice and a double quantum dot. Section VIII contains concluding remarks.

II. OVERVIEW OF THE MAIN RESULTS

Since TCI and the Wick determinant formalism for high-order expansions might be unfamiliar to some readers, we begin with a brief motivating overview, including a sample of our main results. Most technical details are postponed until later sections.

We consider a Hamiltonian of the form

$$H = H_0 + UH_{\mathrm{int}} \tag{1}$$

with interaction term U and a physical observable $Q(U)$, e.g., the charge in a simple quantum impurity model in the steady state. It has a perturbative expansion

$$Q(U) = \sum_n Q_n U^n \tag{2}$$

with

$$Q_n = \int dv_1 \ldots dv_n \tilde{Q}_n(v_1, \ldots, , v_n), \tag{3}$$

where v are time differences. The Schwinger-Keldysh formalism provides explicit expressions for $\tilde{Q}_n$ in terms of the propagators of H_0. The difficulty lies in the calculation of the n-dimensional integral (3).

Our main result is a compressed approximate representation of $\tilde{Q}_n$ as a matrix product state (MPS):

$$\tilde{Q}_n(v_1, \ldots, v_n) \approx M_1(v_1) \cdots M_n(v_n), \tag{4}$$

where M are matrices of maximal dimension χ, the so-called bond dimension. As the variables are now separated, we have

$$Q_n \approx \left(\int dv_1 M_1(v_1) \right) \cdots \left(\int dv_n M_n(v_n) \right). \tag{5}$$

The central point of this paper is to demonstrate the existence of a highly accurate tensor interpolation of the form (4) for the bare n-body correlators involved in the perturbative expansions at order n, with a moderate bond dimension χ which does not grow significantly with n. This tensor representation can be obtained from $O(n\chi^2)$ evaluations of $\tilde{Q}_n$ using the TCI algorithm, even though the integration volume grows exponentially with n. Furthermore, the approximation is *systematically controlled* by χ. Using this MPS form, the complexity of computing the n-dimensional integral becomes $O(nd\chi^2)$ rather than $O(d^n)$, where d is the number of discretization points (or basis functions) in each dimension. These complexities are expressed in the number of evaluations of the integrand $\tilde{Q}_n(v_i)$. To obtain the total complexity, a factor 2^n must be included to account for the complexity of a single evaluation of $\tilde{Q}_n(v_i)$ in the Keldysh formalism.

The quality of the tensor interpolation is illustrated in Fig. 1(a), for the coefficient Q_n of the perturbative expansion of the charge Q of the Anderson quantum impurity model. We present $\tilde{Q}_n$ on a path in the n-dimensional integration domain (orange line) and its MPS approximation (4) (blue dots) for $\chi = 30$. In Fig. 1(b), we show the convergence of the integral Q_{19} compared to the exact Bethe ansatz solution, as a function of the number N of evaluations of the integrand $\tilde{Q}_{19}$. We obtain an unprecedented $O(1/N^2)$ convergence down to a relative error level of 10^{-7}.

Since it is based on a full interpolation of the correlators, the TTD method allows one to compute, at no extra cost, (i) the *full time dependency of* $Q_n(t)$ after the interaction quench at $t = 0$ and (ii) the same for any *time-dependent coupling constant* $U(t)$ [by multiplying by $U(t)$ before integrating]. This is discussed in detail in Sec. VI B.

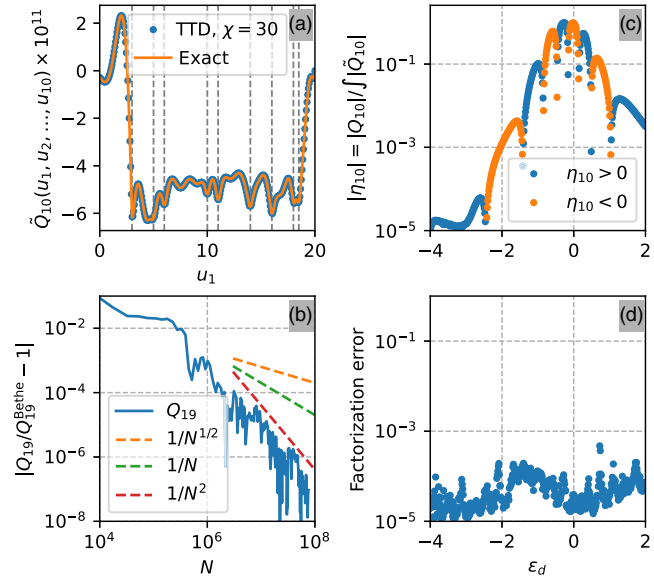

FIG. 1. Overview of the main results. (a) Slice of the corresponding integrand $\tilde{Q}_{10}$ (orange line) compared to the MPS approximation (blue dots). The values of $u_2, u_3, \ldots, u_{10}$ are arbitrarily fixed (vertical dashed lines). (b) Relative error of the nth coefficient (for $n = 19$) in the perturbative expansion of the charge Q of the Anderson quantum impurity model (compared with the exact Bethe ansatz solution) versus the number N of evaluation of $\tilde{Q}_{19}$. (c) Average sign defined as $\eta_n = Q_n / \int |\tilde{Q}_n|$ for $n = 10$ versus on-site energy ϵ_d. (d) Relative error of the rank-50 MPS approximation [pivot error, as defined in Eq. (24), divided by the value of the function $\tilde{Q}_n$ at the first pivot] versus ϵ_d.

The TTD has two fundamental differences with DMRG and its higher-dimensional generalizations. First, the tensor decomposition applies to n-body correlators instead of many-body wave functions. Second, in DMRG, the unknown wave function is represented by a TN ansatz which is variationally optimized. Here, the function $\tilde{Q}_n$ is known (it is the input of the problem). We compress it in order to avoid an exponential integration cost. TCI belongs to the class of "active machine learning" algorithms: The tensor approximation is constructed by evaluating an n-body correlator and finding the region in its n-dimensional space with the largest approximation error.

The TTD has two major advantages compared to diagrammatic quantum Monte Carlo. First, we observe a faster convergence rate of $O(1/N^2)$ instead of $O(1/\sqrt{N})$. Second, the ϵ-factorization is completely unrelated to the average sign of the integral, as illustrated in Figs. 1(c) and 1(d). The average sign η_{10} [Fig. 1(c)] varies over 5 orders of magnitude as a function of one parameter of the model (here, ϵ_d, the on-site energy of the Anderson model), while the error of the factorization at fixed tensor rank χ [Fig. 1(d)] is constant with ϵ_d. A small value of η implies a major sign problem for diagrammatic Monte Carlo—cf. Sec. V—whereas the TTD has no such

problem. The limiting factors of TTD and Monte Carlo are, therefore, completely different.

Finally, let us discuss the quasi-Monte-Carlo technique which was recently introduced by some of the authors [33]. It represents an intermediate step between Monte Carlo and TTD, since it combines a (much weaker) ϵ-factorizability for the tails of $\tilde{Q}_n$ at large v with a quasi-Monte-Carlo technique to compute the Feynman integrals. While it produces convergence as fast as $1/N$ in good cases, it is, in our benchmarks, much less robust than the TTD. Furthermore, as a (nonstochastic) sampling technique, it also suffers from a sign problem when $\tilde{Q}_n$ is highly oscillatory.

III. TENSOR TRAIN CROSS INTERPOLATION

We start with a review of tensor cross interpolation. Most of the material in this section is not original (see Refs. [6,7,12–17]) except, to our knowledge, the environment-aware error function in Sec. III B 4. We present it here in detail so that the article is self-contained. We also show explicitly that most of the results initially derived for matrices and tensors are directly generalizable to multidimensional functions. The appendixes include explicit proofs of the statements made here in the main text. Note that, in this class of algorithm, the main difficulty lies in the bookkeeping of the various slices of the tensor held in memory. Hence, the choice of notation plays a particularly important role.

A. Matrix cross interpolation

Given an $M \times N$ matrix A, the *cross interpolation* technique (CI) yields an approximate rank χ factorization of A. It is distinct from the truncated singular value decomposition (SVD), in which one approximates A by its SVD with all but the largest χ singular values set to zero. Although the truncated SVD yields an optimal rank χ approximation of A in the spectral norm, CI has the advantage that it may be constructed by querying only a small subset of the entries of A. CI is quasioptimal in the sense that its error is at most $O(\chi^2)$ times the optimal one [37,38].

We begin by establishing our notation. Let $\mathcal{I} = \{i_1, i_2, \ldots, i_\chi\}$ (respectively, $\mathcal{J} = \{j_1, j_2, \ldots, j_\chi\}$) denote a list of rows (columns) of A, $\mathcal{I}_a \equiv i_a$ its ath element, and $\mathbb{I} = \{1, 2, \ldots, M\}$ ($\mathbb{J} = \{1, 2, \ldots, N\}$) the list of the indices of all rows (columns). Following the Python and MATLAB convention, we denote by $A(\mathcal{I}, \mathcal{J})$ the submatrix of A comprised of the rows $\mathcal{I}$ and columns $\mathcal{J}$; $A(\mathcal{I}, \mathcal{J})_{ab} \equiv A_{\mathcal{I}_a, \mathcal{J}_b}$. In particular, $A(\mathbb{I}, \mathbb{J}) = A$.

The matrix cross interpolation formula reads

$$A = A(\mathbb{I}, \mathbb{J}) \approx A(\mathbb{I}, \mathcal{J}) A(\mathcal{I}, \mathcal{J})^{-1} A(\mathcal{I}, \mathbb{J}). \tag{6}$$

Equation (6) is illustrated graphically in Fig. 2. It has two main properties:

(P1) It is an interpolation; i.e., it is exact for any $i \in \mathcal{I}$ or $j \in \mathcal{J}$. This can be straightforwardly checked from the definition as, e.g., $A(\mathcal{I}, \mathcal{J}) A(\mathcal{I}, \mathcal{J})^{-1} A(\mathcal{I}, \mathbb{J}) = A(\mathcal{I}, \mathbb{J})$.

(P2) It is exact if the matrix A has rank χ (cf. Appendix B for a simple proof).

The elements of the nonsingular submatrix $A(\mathcal{I}, \mathcal{J})$ are called the *pivots* and $A(\mathcal{I}, \mathcal{J})$ the *pivot matrix*. The pivots should be chosen to minimize the error in the approximation (6). There is an exponentially large number of possible choices of pivot matrix, so it is impossible in practice (for a large matrix A) to try all of them. However, well-established heuristic algorithms exist which provide good quality pivots, by maximizing the magnitude of the determinant of the pivot matrix. This is known as the *maxvol principle* (i.e., maximum volume) [15,17].

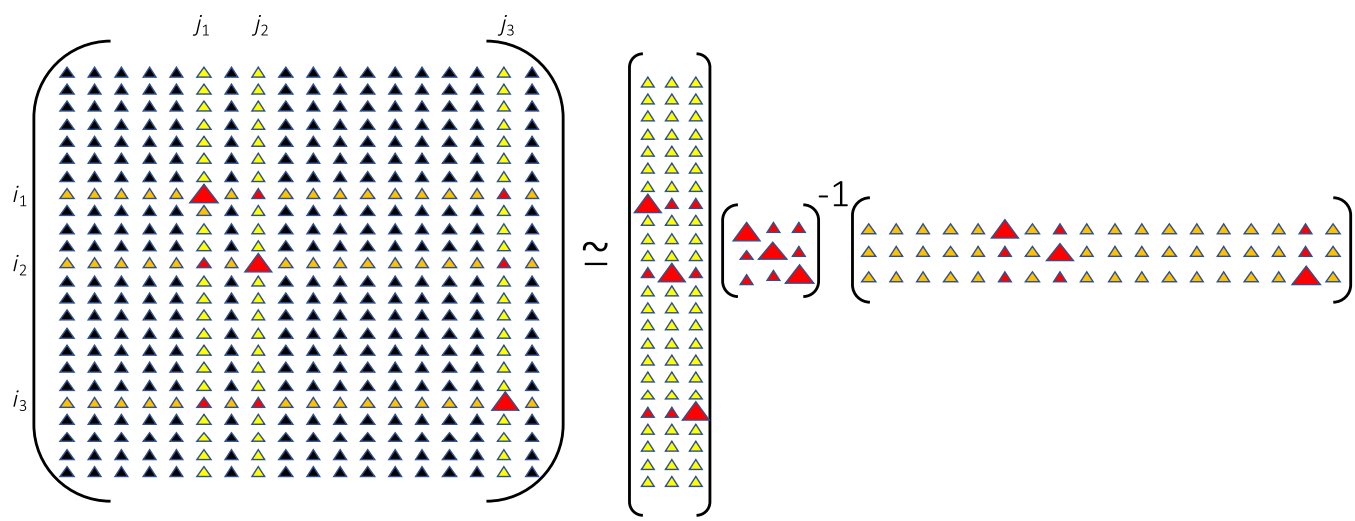

FIG. 2. Illustration of the CI of a matrix. The large red triangles indicate real pivots, and the smaller red triangles indicate automatically generated pivots. The right-hand side contains only small subparts of the matrix.

In this work, we need a generalization of the CI to the continuum [16,18,37]. We refer to a real-valued function $A(x, y)$ as *ϵ-factorizable* in the CI sense with *finite* rank χ if it can be approximated with error ϵ as

$$A(x, y) \approx \sum_{ab} A(x, y_a)[A(\mathcal{I}, \mathcal{J})^{-1}]_{ab} A(x_b, y). \quad (7)$$

Here, $\mathcal{I} = (x_1, x_2, \ldots, x_\chi)$ and $\mathcal{J} = (y_1, y_2, \ldots, y_\chi)$ are *finite* sets of x and y values. The CI (7) uses a *finite* number 2χ of one-dimensional functions $A(x, y_a)$ and $A(x_b, y)$. Using implicit summation, we rewrite Eq. (7) as

$$A(x, y) \approx A(x, \mathcal{J}) A(\mathcal{I}, \mathcal{J})^{-1} A(\mathcal{I}, y). \quad (8)$$

The continuous version of the CI also has the properties (P1) and (P2).

Integrating an *ϵ-factorizable* function is greatly simplified by its approximate separability of variables, as only one-dimensional integrals need to be performed:

$$\int dx dy A(x, y) \approx \left[\int dx A(x, \mathcal{J})\right] \times A(\mathcal{I}, \mathcal{J})^{-1} \left[\int dy A(\mathcal{I}, y)\right]. \quad (9)$$

The CI has other similar properties. For instance if the one-dimensional slices are sufficiently well represented (i.e., a good interpolant is built for each of them), then we can also obtain an approximation of the gradient $\vec{\nabla} A(x, y)$, from which one may perform optimization:

$$\frac{\partial A}{\partial y}(x, y) \approx A(x, \mathcal{J}) A(\mathcal{I}, \mathcal{J})^{-1} \frac{\partial A}{\partial y}(\mathcal{I}, y). \quad (10)$$

For practical implementations, it is important to note that evaluating Eq. (7) directly may be numerically unstable, since for large values of χ the pivot matrix becomes almost singular. An equivalent but stable evaluation method using the QR decomposition is explained in Appendix B for the TCI.

B. Tensor train interpolation

We know turn to the TCI, as introduced in Ref. [12], which is the generalization of the matrix cross interpolation to n-dimensional tensors and functions. TCI is also quasioptimal if the maxvol principle is used [14]. We consider an n-dimensional function $A(u_1, \ldots, u_n)$, where the u_i are either discrete or continuous variables. The TCI literature typically deals with the discrete case, in which A is a tensor, and each index u_i can take d different values, so that A has d^n entries. In this work, we also use a generalization to the continuous case. Following standard notation, tensor networks are depicted as a rectangle with n "legs" (indices); see Fig. 3.

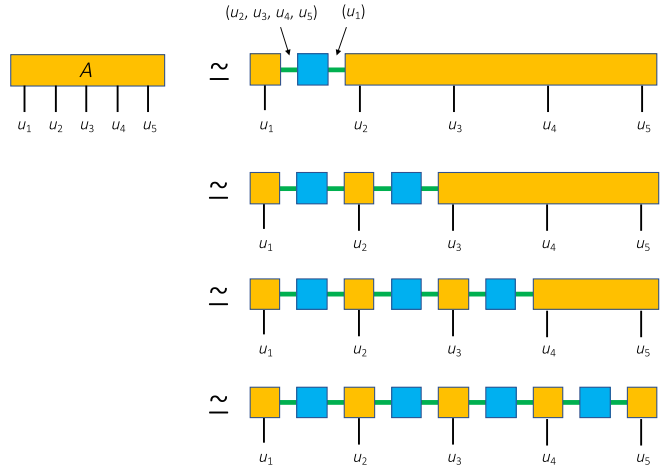

FIG. 3. Step by step representation of a simple algorithm to factorize a multidimensional tensor into a tensor train. The blue squares represent the inverses of the pivot matrices. Summation is implicit over the indices connecting two tensors (green lines).

1. Naive approach

Let us first present a simple algorithm, illustrated in Fig. 3, to decompose a tensor. While it is not efficient and not used in practice, it provides a pedagogical introduction to TCI for the unfamiliar reader.

First, we view the tensor A as a matrix $A_{(u_1),(u_2,u_3,\ldots,u_n)}$ by regrouping the indices into u_1 and a *multi-index* $(u_2, u_3, u_4 \ldots u_n)$. Second, we apply the CI to this matrix and decompose it as a product of three matrices, as shown in the right-hand side of the first line in Fig. 3. Here, the blue square stands for the inverse of the pivot matrix. Crucially, since we keep only a finite number χ of pivots, the summation over the repeated indices (green lines) involves only a small number of terms, even if the variable u_i is continuous. Next, we consider the (orange) tensor on the right side of the first line in Fig. 3. We regroup the χ values of u_1 of the pivots and the d values of u_2 into a multi-index (u_1, u_2) and form the matrix $A_{(u_1,u_2),(u_3,u_4,\ldots,u_n)}$. Applying CI to this matrix yields the second line in Fig. 3. This process is continued until all the orange tensors have only one black leg, which yields the tensor train represented in the last line in Fig. 3.

From this simple algorithm, we can already observe the extension of property (P2) from matrices to tensors: If a tensor has rank χ (which we define as each of the above matrices has rank χ), then all the steps above are exact for a correct choice of pivots, and the tensor train is an exact representation of the tensor. Furthermore, like in the CI, the orange rectangle (respectively, blue square) tensors in Fig. 3 correspond to subtensors (respectively, matrix inverses of subtensors) of the initial tensor A.

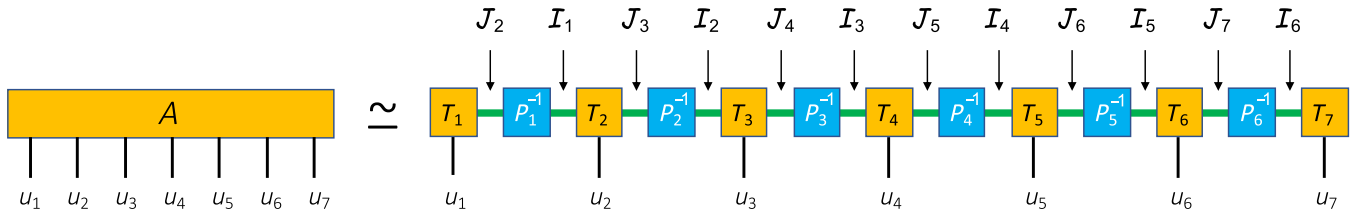

FIG. 4. Pictorial representation of TCI formula for A defined in Eq. (16). The T_α tensors and P_α^{-1} pivot matrices are represented by orange and blue squares, respectively. The green lines correspond to contracted discrete indices $\mathcal{I}_\alpha$ or $\mathcal{J}_\alpha$, as indicated by the arrows. The thin black lines correspond to the variables u_α.

2. *Tensor train interpolation*

The goal of TCI is to perform the decomposition in Sec. III B 1 using only a few calls to the function $A(u_1, \ldots, u_n)$. Let us now introduce our notation and definitions for TCI, in particular, the multi-index notation. A graphical illustration of the tensor train notation is shown in Fig. 4.

We consider a tensor $A(u_1, \ldots, u_n)$, with u_i taking a finite set of d discrete values. The generalization to continuous variables is discussed below. For any α such that $1 \le \alpha \le n$, we use *multi-indices* of the following form: $i = (u_1, u_2, \ldots, u_\alpha)$ and $j = (u_\alpha, u_{\alpha+1}, \ldots, u_n)$. We let $\mathcal{I}_\alpha = \{i_1, i_2, \ldots, i_\chi\}$ denote *a set of* χ *multi-indices* of size α, and let $\mathcal{J}_\alpha = \{j_1, j_2, \ldots, j_\chi\}$ similarly denote a set of χ multi-indices of size $n - \alpha + 1$. Since each of its elements is a multi-index, $\mathcal{I}$ is a "list of lists" of values of the variables u_i. For notational convenience, we define $\mathcal{I}_0$ and $\mathcal{J}_{n+1}$ as singleton sets each comprised of an empty multi-index. In the following, we reserve the notation i and j for such multi-indices, without emphasizing their dependence on α explicitly.

We use the symbol $\oplus$ to denote concatenation of multi-indices:

$$(u_1, u_2, \ldots, u_{\alpha-1}) \oplus u_\alpha \oplus (u_{\alpha+1}, \ldots, u_n) \equiv (u_1, \ldots, u_n). \tag{11}$$

Note that, when using the $\oplus$ operator, we omit parentheses for a multi index of size 1. We also define $\mathbb{K}_\alpha$ as the set of all values of the multi-index (u_α) of size 1, with $1 \le \alpha \le n$. Finally, we define $\mathcal{I} \oplus \mathcal{J}$ as the set of all concatenations of the elements of $\mathcal{I}$ and $\mathcal{J}$: $\mathcal{I} \oplus \mathcal{J} \equiv \{i \oplus j | i \in \mathcal{I}, j \in \mathcal{J}\}$.

To illustrate these notations, let us give a concrete example for $n = 4$ and $\chi = 2$ and $0 \le u_1 < u_2 < u_3 < u_4 \le 1$. A possible choice is $\mathcal{I}_2 = \{(0.2, 0.45), (0.1, 0.6)\}$, $\mathcal{I}_3 = \{(0.2, 0.45, 0.7), (0.1, 0.6, 0.8)\}$, $\mathcal{J}_3 = \{(0.72, 0.92), (0.76, 0.92)\}$, and $\mathcal{J}_4 = \{(0.98), (0.92)\}$. Note that this choice respects the nesting condition defined below. Operations provide, e.g., $\mathcal{I}_2 \oplus \mathcal{J}_3 = \{(0.2, 0.45, 0.72, 0.92), (0.1, 0.6, 0.72, 0.92), (0.2, 0.45, 0.76, 0.92), (0.1, 0.6, 0.76, 0.92)\}$.

We now define the tensors T_α and P_α by the expressions

$$T_\alpha(i, u_\alpha, j) \equiv A(i \oplus u_\alpha \oplus j) = A(u_1, u_2, \ldots, u_n), \tag{12}$$

with $i \in \mathcal{I}_{\alpha-1}$, $j \in \mathcal{J}_{\alpha+1}$, and

$$P_\alpha(i, j) \equiv A(i \oplus j) = A(u_1, u_2, \ldots, u_n), \tag{13}$$

with $i \in \mathcal{I}_\alpha$, $j \in \mathcal{J}_{\alpha+1}$. Here, $1 \le \alpha \le n$ and $1 \le \alpha \le n - 1$ for T_α and P_α, respectively. More abstractly, we can write

$$T_\alpha \equiv A(\mathcal{I}_{\alpha-1} \oplus \mathbb{K}_\alpha \oplus \mathcal{J}_{\alpha+1}), \tag{14a}$$

$$P_\alpha \equiv A(\mathcal{I}_\alpha \oplus \mathcal{J}_{\alpha+1}). \tag{14b}$$

For notational convenience, we define P_0 and P_n as the 1×1 unit matrix. For fixed α, T_α is therefore of dimension $\chi \times d \times \chi$, except T_1 and T_n, which are of dimension $1 \times d \times \chi$ and $\chi \times d \times 1$, respectively. Similarly, P_α is of dimension $\chi \times \chi$. T_α is, therefore, a three-leg tensor (whose name comes from its "T" shape), and P_α is a matrix. From these definitions, we see that if one selects one of the χ multi-indices $i \in \mathcal{I}$ and one of the χ multi-indices $j \in \mathcal{J}$, then T_α defines a one-dimensional slice of the original tensor A along the variable u_α. We lastly note that the position of the indices in P^{-1} is transposed compared to P due to the inversion. The T_α tensors and the P_α matrices are given a schematic representation shown in the right-hand side in Fig. 4 as, respectively, a three-leg orange tensor and a blue matrix with the discrete indices i and j in green, while the u_α are in black.

We also use matrix notation for T_α by defining $T_\alpha(u)$ as the matrix of values of the tensor with fixed $u_\alpha = u$. We have

$$[T_\alpha(u)]_{ij} \equiv T_\alpha(i, u, j), \tag{15a}$$

$$(P_\alpha)_{ij} \equiv P_\alpha(i, j). \tag{15b}$$

Using these notations, the TCI representation of A takes a simple form in terms of matrix multiplications. It is a tensor train of the form

$$A(u_1, \ldots, u_n) \approx A_{\mathrm{TCI}}(u_1, \ldots, u_n) \equiv \prod_{\alpha=1}^{n} T_\alpha(u_\alpha) P_\alpha^{-1}. \tag{16}$$

Note that, given the dimensions of T_1, T_n, and P_n, this product is a scalar. This TCI representation is illustrated in

Fig. 4. Each green line corresponds to a set of multi-indices $\mathcal{I}_\alpha$ ("rows") or $\mathcal{J}_\alpha$ ("columns"). It is important to notice that the TCI representation is defined entirely by the selected sets of rows and columns $\mathcal{I}_\alpha$ and $\mathcal{J}_\alpha$, so that constructing an accurate representation of A amounts to optimizing the selection of $\mathcal{I}_\alpha$ and $\mathcal{J}_\alpha$ for $1 \le \alpha \le n$.

We impose a restriction on the possible choices of $\mathcal{I}_\alpha$ and $\mathcal{J}_\alpha$ called the *nesting condition* [7,14]—$\mathcal{I}_\alpha$ ($\mathcal{J}_\alpha$) is constructed from elements of $\mathcal{I}_{\alpha-1}$ ($\mathcal{J}_{\alpha+1}$), except for the last (first) variable, which is taken from $\mathbb{K}_\alpha$:

$$\mathcal{I}_\alpha \subset \mathcal{I}_{\alpha-1} \oplus \mathbb{K}_\alpha, \tag{17a}$$

$$\mathcal{J}_\alpha \subset \mathbb{K}_\alpha \oplus \mathcal{J}_{\alpha+1}. \tag{17b}$$

In other words, if $i \in \mathcal{I}_\alpha$, then there is a $k \in \mathcal{I}_{\alpha-1}$ such that $i = k \oplus u_\alpha$ for some $u_\alpha \in \mathbb{K}_\alpha$. Similarly, if $j \in \mathcal{J}_\alpha$, then there is a $k \in \mathcal{J}_{\alpha+1}$ such that $j = u_\alpha \oplus k$ for some $u_\alpha \in \mathbb{K}_\alpha$. We show in Appendix C 1 that imposing the nesting condition guarantees the generalization of the interpolation property (P1) for the tensor train, namely,

$$A_{\mathrm{TCI}}(\mathcal{I}_{\alpha-1}, \mathbb{K}_\alpha, \mathcal{J}_{\alpha+1}) = A(\mathcal{I}_{\alpha-1}, \mathbb{K}_\alpha, \mathcal{J}_{\alpha+1}) \tag{18}$$

for $1 \le \alpha \le n$. In other words, the approximation is exact for any indices that define one of the tensors T_α and *a fortiori* to those that define one of the P_α matrix.

The TCI approximation of $A(u_1, u_2, \ldots, u_n)$ is built from one-dimensional slices (i.e., partial evaluations) of A (the T_α tensors with fixed α, i, and j). Therefore, only $O(nd\chi^2) \ll d^n$ entries of A are used in the approximation.

As for matrix cross interpolation, this construction can be directly generalized to continuous variables. We call a function $A(u_1, u_2, \ldots, u_n)$ ϵ-factorizable if the factorization (16) satisfies $||A - A_{\mathrm{TCI}}||_\infty < \epsilon$ with χ finite and increasing "slowly" as ϵ is decreased. The TCI is particularly useful to compute the n-dimensional integral of A, which is our goal in this paper. Indeed, it separates the variables, reducing the calculation of the n-dimensional integral to that of $O(n\chi^2)$ one-dimensional integrals, followed by the tensor contraction (16):

$$\int du_1 \cdots du_n A(u_1, \ldots, u_n) \approx \prod_{\alpha=1}^{n} \int du_\alpha T_\alpha(u_\alpha) P_\alpha^{-1}. \tag{19}$$

3. Algorithm to construct the TCI

We now turn to the algorithm used to construct the TCI and, in particular, to find the set of pivots. Our implementation is essentially equivalent to that described in Ref. [7]. We start with an initial point $(u_1, \ldots, u_n) = (u_1, \ldots, u_\alpha) \oplus (u_{\alpha+1}, \ldots, u_n)$, which we split in $n-1$ different ways to obtain one element for each of the sets $\mathcal{I}_\alpha$ and $\mathcal{J}_\alpha$. This yields the initial $\chi = 1$ TCI, which is exact if the function $A(u_1, \ldots, u_n)$ factorizes as a product of functions of one variable.

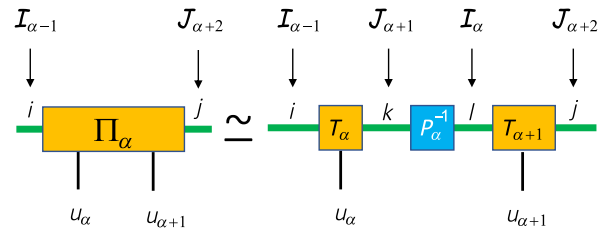

FIG. 5. Pictorial representation of the Π_α tensor and its cross interpolation. The notation is the same as in Fig. 4.

Let us now define the tensors Π_α, named for their four-legged shapes (see Fig. 5), by

$$\Pi_\alpha \equiv A(\mathcal{I}_{\alpha-1} \oplus \mathbb{K}_\alpha \oplus \mathbb{K}_{\alpha+1} \oplus \mathcal{J}_{\alpha+2}). \tag{20}$$

A pictorial representation of Π_α is shown in Fig. 5. Considering Π_α as a matrix with (i, u_α) being the row index and $(u_{\alpha+1}, j)$ the column index, one can build a cross interpolation of Π_α using the pivots $\mathcal{I}_\alpha$ and $\mathcal{J}_{\alpha+1}$. The resulting approximation of Π_α reads

$$\Pi_\alpha(i, u_\alpha, u_{\alpha+1}, j) \approx \sum_{kl} T_\alpha(i, u_\alpha, k) P_\alpha^{-1}(k, l) T_{\alpha+1}(l, u_{\alpha+1}, j) \tag{21}$$

or, equivalently, using matrix notation, as

$$\Pi_\alpha(u_\alpha, u_{\alpha+1}) \approx T_\alpha(u_\alpha) P_\alpha^{-1} T_{\alpha+1}(u_{\alpha+1}). \tag{22}$$

We introduce the *error function* ϵ_Π:

$$\epsilon_\Pi(i, u_\alpha, u_{\alpha+1}, j) \equiv \left| \Pi_\alpha(i, u_\alpha, u_{\alpha+1}, j) - \sum_{kl} T_\alpha(i, u_\alpha, k) \times P_\alpha^{-1}(k, l) T_{\alpha+1}(l, u_{\alpha+1}, j) \right|, \tag{23}$$

where $i \in \mathcal{I}_{\alpha-1}$, $u_\alpha \in \mathbb{K}_\alpha$, $u_{\alpha+1} \in \mathbb{K}_{\alpha+1}$, and $j \in \mathcal{J}_{\alpha+2}$. We show in Appendix C 2 that, as a result of the nesting condition, the error function satisfies

$$\epsilon_\Pi(i, u_\alpha, u_{\alpha+1}, j) = |A - A_{\mathrm{TCI}}|(i, u_\alpha, u_{\alpha+1}, j). \tag{24}$$

In other words, the error of the factorization of Π_α is, in fact, the error of the interpolation A_{TCI} with respect to A, computed on the two-dimensional slice determined by i and j. Hence, improving the factorization of Π_α does indeed improve the overall TCI representation of A.

The algorithm adds more pivots to the sets $\mathcal{I}_\alpha$ and $\mathcal{J}_{\alpha+1}$ in order to improve the approximation of Π_α while maintaining the nesting condition. It finds a local maximum $(i, u_\alpha, u_{\alpha+1}, j)$ of the error function ϵ_Π for $i \in \mathcal{I}_{\alpha-1}$ and $j \in \mathcal{J}_{\alpha+2}$, adds the new pivots $i \oplus u_\alpha$ and $u_{\alpha+1} \oplus j$ to $\mathcal{I}_\alpha$

and $\mathcal{J}_{\alpha+1}$, respectively, and then updates the pivot matrix P_α. This procedure preserves the nesting condition. The rationale for adding the pivot for which the error is *maximum* is that this choice of pivot yields the largest improvement in the accuracy of the TCI approximation, since the corresponding point becomes exact. Another way to understand this choice, as shown in Appendix B 2, is that this choice gives the largest determinant for the corresponding P_α matrix, i.e., follows the maxvol principle.

In the *full search* variant of the algorithm, the maximum of ϵ_Π is determined by a brute force search over all $(\chi d)^2$ values of Π_α. In the much faster *alternate search* variant, one searches for a local maximum of ϵ_Π by starting from a random point and scanning (i, u_α) and $(u_{\alpha+1}, j)$ alternatively. The search ends when a local maximum is found or a maximum number of iterations (typically three or four) is reached. The computational cost of the alternate search variant is only $O(d\chi)$ for adding a new pivot and, hence, $O(d\chi^2)$ globally. In practice, for the cases considered in this paper, we observe little difference in the quality of the approximation obtained using the two variants, and we therefore use the alternate search variant for all results presented below.

For the case of continuous (u_α), we explore two approaches: (i) search for the pivots on a predefined grid and (ii) search for a local maximum directly in the continuum, using standard optimization algorithms. Since we do not observe obvious advantages in using the second approach, we use the first method for the results presented below.

We perform $n_{\rm sw}$ *sweeps* of this procedure, each consisting of a forward sweep, which improves all Π tensors from Π_1 to Π_{n-1}, and a backward sweep, which improves all Π tensors from Π_{n-1} to Π_1. Each sweep increases the bond dimension χ by two (at most; see Sec. VI E) so that $\chi \leq 2n_{\rm sw} + 1$.

4. Improved pivoting using an environment-aware error function

The error function ϵ_Π is quite natural and is used in Refs. [7,14]. The standard choice in the literature is to follow the maxvol principle [12,13], where one looks for pivots that maximize the determinant of the pivot matrix P_α. Appendix B 2 shows that the two criteria are closely related. Since our goal is to compute n-dimensional integrals, we find that another error function, directly associated to the error of the integral, yields significantly better results in the cases we study.

Let us consider a single Π_α tensor in the TCI (16) and integrate over all variables u_β except u_α and $u_{\alpha+1}$. We have (in matrix notation)

$$I \equiv \int du_1 \ldots du_n A(u_1, \ldots, u_n)$$
$$\approx \int du_\alpha du_{\alpha+1} \sum_{i,j} L_i [T_\alpha(u_\alpha) P_\alpha^{-1} T_{\alpha+1}(u_{\alpha+1})]_{ij} R_j, \quad (25)$$

where

$$L \equiv \left(\int T_1\right) P_1^{-1} \ldots \left(\int T_{\alpha-1}\right) P_{\alpha-1}^{-1}, \quad (26)$$

$$R \equiv P_{\alpha+1}^{-1} \left(\int T_{\alpha+2}\right) \ldots P_{n-1}^{-1} \left(\int T_n\right) \quad (27)$$

are vectors of length χ and

$$\int T_\alpha \equiv \int du_\alpha T_\alpha(u_\alpha). \quad (28)$$

A better approximation of the n-dimensional integral, which replaces the part of the factorization involving the u_α and $u_{\alpha+1}$ variables with the exact slice Π_α, is

$$I \approx \int du_\alpha du_{\alpha+1} \sum_{i,j} L_i \Pi_\alpha(u_\alpha, u_{\alpha+1})_{ij} R_j. \quad (29)$$

Here, we see that the Π_α tensor in the integral is weighted by the factors L and R, which we refer to as the *environment* in a manner reminiscent of DMRG. We, therefore, propose taking the difference of Eqs. (25) and (29) and using the modulus of the resulting integrand as an error function:

$$\epsilon_\Pi^{\rm env}(i, u_\alpha, u_{\alpha+1}, j) \equiv |L_i R_j| \epsilon_\Pi(i, u_\alpha, u_{\alpha+1}, j). \quad (30)$$

We refer to this as the *env* variant of the algorithm. In practice, we also multiply this error with another weight $W_a W_b$ defined in the next section (weighted learning variant).

We show that the env variant significantly outperforms the standard algorithm using ϵ_Π in the cases considered below. Indeed, it leads to the selection of pivots in regions of large volume in which the integrand is small. This is illustrated by analogy with the following integral: $\int_0^\infty dx (e^{-x} + e^{-x/100}/100) = 2$. One of the terms in the integrand is rapidly decaying, and the other is slowly decaying and small, but both contribute equally to the integral. The ordinary choice of pivots leads to sampling the integrand based on its absolute value, ignoring the weighting of the corresponding contribution by its volume. The corresponding algorithm would focus on improving the description of the large term and start adding points in the tail region $x \gg 1$ only once the large term is known very accurately. Instead, the two terms should be approximated with an error weighted by their respective contributions to the integral. By including the corresponding volumes in the weight, the error function $\epsilon_\Pi^{\rm env}$ implements this idea.

5. Quadrature rules for numerical integration

It remains to specify a method of calculating the one-dimensional integrals Eq. (28). The integration in the case that the underlying domain is a simplex, rather than a hypercube, is discussed below. This question is independent

of tensor factorizability and the TCI construction. The behavior of the functions $T_\alpha(u_\alpha)$ varies from model to model, as does the precise domain of integration, which is discussed in the next section on real-time computations.

In this work, we use rules based on either Chebyshev polynoms or Gauss-Kronrod quadratures to perform these one-dimensional integrals [39]. The quadrature rule associated to Chebyshev polynoms is known as the Clenshaw-Curtis quadrature. However, we also use Chebyshev interpolants to perform integrations on domains smaller than the initial domain used to construct the Chebyshev interpolant, and in that case we get different weights. We note CH_x (respectively, GK_x) the rules for Chebyshev polynoms (respectively, Gauss-Kronrod quadrature) with x points. In one application, we encounter highly oscillatory and slowly decaying integrals. Although these integrals could be calculated with standard quadratures by brute force, we find that building a specialized quadrature yields a significant improvement in efficiency (see Appendix I).

These quadratures specify a set of d points x_a and weights W_a such that

$$\int du T_\alpha(u) \approx \sum_{a=1}^{d} W_a T_\alpha(x_a). \tag{31}$$

The multidimensional integral I then reduces to the full contraction of the tensor with the weights W_a, i.e., the contraction of $A(x_{i_1}, \ldots, x_{i_n}) W_{i_1} \ldots W_{i_n}$. In the *weighted learning* variant of the algorithm, the TCI is constructed for this weighted tensor rather than the original tensor A. It is argued in the literature [7] that weighted learning improves the convergence of I with the bond dimension. We show below that, in the cases considered here, the improvement is marginal.

IV. REAL-TIME MANY-BODY FORMALISM

The formalism used in this article follows the Keldysh approach in real time that is used in the context of diagrammatic quantum Monte Carlo calculations [25]. For completeness, we review the main definitions and expressions which are needed later. We refer to Ref. [30] for proofs and additional details.

A. Perturbation theory with Wick determinants

Our starting point is a Hamiltonian $H = H_0 + H_{\mathrm{int}}\lambda(t)$ consisting of an arbitrary noninteracting term H_0 and an interaction term H_{int} that is switched on at $t = 0$. The noninteracting part is arbitrary:

$$H_0 = \sum_{ii',\sigma\sigma'} (H_0)_{ii',\sigma\sigma'} c^\dagger_{i\sigma} c_{i'\sigma'}. \tag{32}$$

However, for concreteness, we focus on H_0 that are diagonal in the spin sector. Here, the fermionic operator $c^\dagger_{i\sigma}$ ($c_{i\sigma}$) creates (destroys) an electron with spin σ on site i. We consider systems directly in the thermodynamic limit, i.e., with an infinite number of sites i. The interaction term can, in principle, be an arbitrary quartic Hamiltonian. However, since all of the calculations in this article are performed using a Hubbard-like interaction, we concentrate on this specific form for concreteness:

$$H_{\mathrm{int}} = U\lambda(t) \sum_{i\in\mathcal{C}} (c^\dagger_{i\uparrow} c_{i\uparrow} - \bar{\alpha})(c^\dagger_{i\downarrow} c_{i\downarrow} - \bar{\alpha}). \tag{33}$$

Here, the sum is taken over a finite subset $\mathcal{C}$ of interacting sites. The $\bar{\alpha}$ term shifts a quadratic term between the noninteracting and the interacting part of the Hamiltonian and, therefore, provides a mathematically different perturbation expansion in powers of U [25,40] of the same physical problem. The function $\lambda(t)$ captures the time dependence of the interaction. One of the remarkable features of TTD is that $\lambda(t)$ is needed only after the factorization is performed, in the postprocessing step in which the integration is carried out. Calculating the time evolution of an observable for different functions $\lambda(t)$ therefore comes essentially for free. Most of the examples treated in this article use $\lambda(t) = \theta(t)$, a Heaviside function, but we also describe a nontrivial example, where $\lambda(t)$ is given by Eq. (57) to illustrate the algorithm's capabilities.

The dynamics of H_0 can be formally solved through the introduction of the corresponding noninteracting Green's functions. The lesser and greater Green's functions $g^<$ and $g^>$ can be computed explicitly from H_0 and comprise, together with the value of U and $\bar{\alpha}$, the actual input of the problem.

We have

$$g^<_{ii',\sigma\sigma'}(t) = i\langle c^\dagger_{i'\sigma'}(0) c_{i\sigma}(t)\rangle, \tag{34}$$

$$g^>_{ii',\sigma\sigma'}(t) = -i\langle c_{i\sigma}(t) c^\dagger_{i'\sigma'}(0)\rangle, \tag{35}$$

where the time dependence of an operator $\mathcal{O}$ is given by $\mathcal{O}(t) = e^{iH_0 t}\mathcal{O}e^{-iH_0 t}$. These Green's functions can be computed analytically for simple models or numerically in more complex cases using, e.g., Tkwant [41]. Their explicit forms for the specific models considered here are given below.

We introduce the general coordinate $X = (i, \sigma, t, a)$, which describes the site index i, the spin σ, the time t, and a "Keldysh index" a taking the values 0 or 1. The Keldysh Green's function $g(X, X')$ is defined as

$$g(X, X') \equiv \begin{cases} g^>_{ii',\sigma\sigma'}(t)\theta(t) + g^<_{ii',\sigma\sigma'}(t)\theta(-t) & \text{for } a = a' = 0, \\ g^>_{ii',\sigma\sigma'}(t)\theta(-t) + g^<_{ii',\sigma\sigma'}(t)\theta(t) & \text{for } a = a' = 1, \\ g^<_{ii',\sigma\sigma'}(t) & \text{for } a = 0, a' = 1, \\ g^>_{ii',\sigma\sigma'}(t) & \text{for } a = 1, a' = 0, \end{cases} \tag{36}$$

where $\theta(t)$ is the Heaviside function and we take $t' = 0$ since $g(X, X')$ is a function of $t - t'$. We also introduce the full interacting Green's functions $G(X, X')$ associated to the full Hamiltonian H. Observables can be related to $G(X, X')$ in a simple manner. For instance, the occupation of an orbital (i, σ) at time t is given by $-iG^{<}_{ii,\sigma\sigma}(t, t)$.

Using this notation, we can write the perturbative expansion of $G(X, X')$ in powers of the interaction coupling U. We obtain

$$G(X, X') = \sum_{n=0}^{\infty} G_n(X, X') U^n, \tag{37}$$

where $G_n(X, X')$ is defined as

$$G_n = \sum_{i_1 i_2 \ldots i_n} \int_{S_u} du_1 du_2 \ldots du_n \lambda(u_1)\lambda(u_2)\ldots\lambda(u_n)\tilde{G}_n. \tag{38}$$

The integration is carried out inside the simplex S_u defined by $0 \le u_n \le \cdots \le u_2 \le u_1 \le t$. Assuming for simplicity that H_0 conserves spin, the integrand $\tilde{G}_n(X, X', i_1, i_2 \ldots i_n, u_1, u_2 \ldots . u_n)$ is given explicitly by

$$\tilde{G}_n = i^n \sum_{a_1,\ldots,a_n} (-1)^{\sum a_k} \left[\!\!\left[\begin{matrix} X, U_1, \ldots, U_n \\ X', U_1, \ldots, U_n \end{matrix} \right]\!\!\right] \left[\!\!\left[\begin{matrix} U_1, \ldots, U_n \\ U_1, \ldots, U_n \end{matrix} \right]\!\!\right], \tag{39}$$

where $U_k = (i_k, u_k, a_k)$, and the "Wick determinant" $[\![\cdots]\!]$ is defined, for $A_1, \ldots, A_m$ and $B_1, \ldots, B_m$ any collections of points on the Keldysh contour, as

$$\left[\!\!\left[\begin{matrix} A_1, \ldots, A_m \\ B_1, \ldots, B_m \end{matrix} \right]\!\!\right] = \begin{vmatrix} g(A_1, B_1) & \cdots & g(A_1, B_m) \\ \vdots & \ddots & \vdots \\ g(A_m, B_1) & \cdots & g(A_m, B_m) \end{vmatrix}. \tag{40}$$

For the case in which $\bar{\alpha} \neq 0$, the diagonal terms of the Wick determinants must be shifted by $-i\bar{\alpha}$ [25].

In the following, we illustrate the method for the calculation of the total charge on site $i = 0$ at a time t after switching on the interaction:

$$Q(t, U) = \langle e^{i\int dtH} c^{\dagger}_{0\uparrow} c_{0\uparrow} e^{-i\int dtH} \rangle. \tag{41}$$

Here, the evolution operator $e^{-i\int dtH}$ is a time-ordered exponential. This observable admits an expansion $Q(t, U) = \sum_n Q_n(t) U^n$, and we refer to the corresponding integrand (39) as $\tilde{Q}_n(i_1, \ldots, i_n, u_1, \ldots, u_n, t)$, so that

$$Q_n(t) = \sum_{i_1 \ldots i_n} \int_{S_u} du_1 du_2 \ldots du_n \lambda(u_1)\lambda(u_2)\ldots\lambda(u_n)\tilde{Q}_n. \tag{42}$$

The nth-order contribution to the expansion is given by an n-dimensional integral. The integrand $\tilde{Q}_n$ is given by a sum of 2^n Wick determinants, which can be computed explicitly from the knowledge of the noninteracting dynamics. The complexity of computing the integrand, therefore, appears to be $O(n^3 2^n)$, but there are known algorithms [42,43] to compute it with $O(2^n)$ complexity. In Appendix D, we present a simpler version of such an algorithm using only a few lines of codes. Hence, the computational problem is reduced to that of computing the high-dimensional integrals above.

B. Models

We consider three different models in this article: a single quantum dot weakly coupled to electrodes, a quantum dot strongly coupled to a two-dimensional infinite electrode, and a double quantum dot weakly coupled to electrodes. The inputs to the TTD method are the corresponding noninteracting Green's functions g. Their explicit forms are given in Appendixes E and F. Figure 6 shows specific examples of these Green's functions for the three different problems. Note that all the examples considered here are invariant with respect to spin rotations, so the Green's functions do not depend on spin.

1. Single-impurity Anderson model (SIAM)

The first system is an interacting quantum dot connected to noninteracting leads: the SIAM. The Hamiltonian is given by $H = H_0 + H_{\rm int}\theta(t)$, with

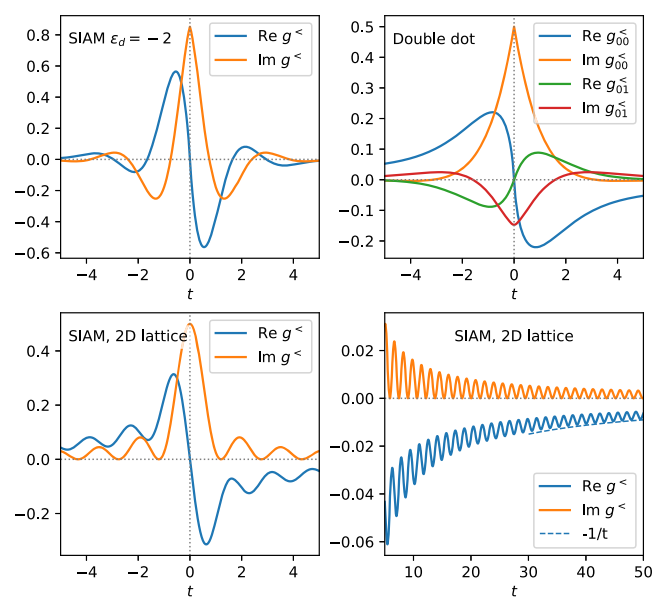

FIG. 6. Real-time noninteracting lesser Green's functions for the three models considered in this paper. Upper panel: single quantum dot (SIAM) and double quantum dot connected to leads in the flat-band limit. Lower panel: one interacting site in the 2D lattice, including the tail of $g^<(t)$.

$$H_0 = \sum_{i\sigma} \gamma_i (c_{i\sigma}^\dagger c_{i+1,\sigma} + \text{H.c.}) + \epsilon_d \sum_\sigma c_{0\sigma}^\dagger c_{0\sigma}, \quad (43)$$

$$H_{\text{int}} = U c_{0\uparrow}^\dagger c_{0\uparrow} c_{0\downarrow}^\dagger c_{0\downarrow}. \quad (44)$$

The hopping parameters are all equal, $\gamma_i = \gamma$, except for the connection of the quantum dot to the leads, $\gamma_0 = \gamma_{-1} \neq \gamma$. We work in the flat-band limit (see Appendix E) in which $\gamma_0, \epsilon_d \ll \gamma$ but $\Gamma = 2\gamma_0^2/\gamma$ is finite. Γ is used as our unit of energy. In this limit, the electron-hole symmetric case $\epsilon_d = 0$ is taken as a benchmark. There, the exact expression for the charge $Q(U)$ at equilibrium is given by the Bethe ansatz [33].

The upper-left panel in Fig. 6 shows an example of the noninteracting Green's function g on the quantum dot for $\epsilon_d = -2$. In this case, the integrand $\tilde{Q}_n$ has many oscillations; see Fig. 1(c) (even more oscillations appear in other cases, for example, $\epsilon_d = -4$).

2. Single impurity in a 2D lattice

Our second system is a quantum dot strongly coupled to a more complex electronic bath. The system is an infinite two-dimensional lattice:

$$H_0 = \gamma \sum_{\langle ij \rangle \sigma} c_{i\sigma}^\dagger c_{j\sigma}, \quad (45)$$

with only one interacting site $i = 0$:

$$H_{\text{int}} = U c_{0\uparrow}^\dagger c_{0\uparrow} c_{0\downarrow}^\dagger c_{0\downarrow}. \quad (46)$$

Here, $\langle ij \rangle$ corresponds to the nearest-neighbor indices in 2D, and γ is taken as the unit of energy. The noninteracting correlators are highly oscillatory, with slow decay $\sim 1/t$; see the lower panel in Fig. 6. The corresponding integrand $\tilde{Q}_n$ decays very slowly, with rapid oscillations.

3. Double quantum dot

The last system is a double quantum dot connected to two electrodes:

$$H_0 = \sum_{i\sigma} \gamma_i (c_{i\sigma}^\dagger c_{i+1,\sigma} + \text{H.c.}), \quad (47)$$

$$H_{\text{int}} = U \sum_{i=0,1} c_{i\uparrow}^\dagger c_{i\uparrow} c_{i\downarrow}^\dagger c_{i\downarrow}, \quad (48)$$

with $\gamma_i = \gamma$, except for $\gamma_{-1} = \gamma_1 \neq \gamma$ and $\gamma_0 \neq \gamma$. We work in the flat-band limit in which $\gamma_0, \gamma_1 \ll \gamma$, and $\Gamma = \gamma_1^2/\gamma$ is taken as the unit of energy. Double quantum dots play an important role in semiconducting quantum technologies, as qubit systems or detectors.

V. TENSOR TRAIN DIAGRAMMATICS

We now turn to the discussion of the TTD technique, i.e., the application of TCI to the calculation of high-order perturbation expansions in the interacting coupling strength, as in Eq. (42). We use the TCI algorithm to factorize $\tilde{Q}_n$ and perform the corresponding n-dimensional integral. This section focuses on impurity models, for which the integral involves n time variables but no spatial sums. The extension to multiorbital models is discussed in Sec. VII.

The calculation can be split into two fundamentally different steps. In Sec. V A, we discuss the factorization of the tensor $\tilde{Q}_n$ appearing in Eq. (42) using a TCI decomposition. In Sec. V B, we discuss the computation of the integral, along the lines outlined above. We discuss the various sources of error in Sec. V C. Detailed benchmarks and numerical results are then presented in the next section.

A. Factorizability in time differences

The *integration domain* in Eq. (42) is the simplex S_u defined by $0 \leq u_n \leq \cdots \leq u_2 \leq u_1 \leq t$. The TCI decomposition itself is constructed in a different domain, the hypercube $[0, t]^n$. We could simply integrate over the whole hypercube and divide the result by $n!$, since the integrand $\tilde{Q}_n$ is symmetric in the u variables as a result of the anticommutativity of fermionic operators under the time-ordered product. However, $\tilde{Q}_n$ has a cusp whenever two of the u_i are equal because of the Heaviside functions introduced by time ordering. Some of these cusps can be seen in Fig. 1(a) (e.g., when $u_1 = u_2$). Such a function does not factorize well; consider, for example, the Heaviside function itself, $\theta(u_1 - u_2)$. We check explicitly in Fig. 15(a) that a direct decomposition in the u variables fails.

We therefore change to the time difference variables v_i, defined by

$$v_1 = t - u_1, \quad (49a)$$

$$v_i = u_{i-1} - u_i \quad \text{for } 2 \leq i \leq n. \quad (49b)$$

This change of variable has a Jacobian $|\det[\partial u_i / \partial v_j]| = 1$. In the v variables, the integration domain becomes

$$v_i \geq 0, \quad (50a)$$

$$\sum_{i=1}^n v_i \leq t \leq t_M, \quad (50b)$$

where t_M is the maximum time of the calculation (possibly infinite for a steady state calculation). The condition (50a) enforces the time ordering in u. As a result, the function

$\tilde{Q}_n(v_i)$ has no cusps due to time ordering inside the hypercube $[0, t_M]^n$ in the v variables.

Using the TCI algorithm, we first obtain a factorization of $\tilde{Q}_n(v_i)$ on $[0, t_M]^n$, as in Eq. (16):

$$\tilde{Q}_n(v_1, \ldots, v_n) \approx \tilde{Q}_n^{\mathrm{TCI}}(v_1, \ldots, v_n) \equiv \prod_{\alpha=1}^{n} T_\alpha(v_\alpha) P_\alpha^{-1}. \quad (51)$$

In practice, we enforce the simplex condition (50b) not only for the integration, but also when choosing new pivots: When searching for pivots which maximize the error ϵ_Π or $\epsilon_\Pi^{\mathrm{env}}$, we consider only candidates that satisfy Eq. (50b). Indeed, we need only to improve our approximation in the integration region. Since the simplex is $n!$ times smaller than the hypercube, this provides a significant speedup. We observe numerically that, for t_M sufficiently large, the disregarded candidate pivots would almost never have been selected anyway. We also observe that the number of evaluations of $\tilde{Q}_n$ used in the pivot search is only approximately 30% of the number of evaluations required to construct the TCI approximation given the pivots. In that sense, the algorithm is close to optimal.

Even in the v variables, the ϵ-factorizability of $\tilde{Q}_n(v_1, \ldots, v_n)$ is far from obvious. For large v_i, some indications can be found. First, it is shown numerically in Ref. [33] that a rank $\chi = 1$ approximation of $\tilde{Q}_n$ is reasonably accurate. This is an essential ingredient in the construction of the n-dimensional change of variables required in quasi-Monte-Carlo methods. Second, when the Green's function g decays exponentially, we expect $\tilde{Q}_n$ to be dominated by a single Feynman diagram (the nested tadpole diagram) and, hence, to factorize with rank $\chi = 2$:

$$\tilde{Q}_n(v_1, \ldots, v_n) \underset{v_i \to \infty}{\sim} \mathrm{Tr} \prod_i M(v_i), \quad (52)$$

where the 2×2 matrix M is given by $M_{aa'}(v_i) = (-1)^a [\hat{g}_{aa'}(v_i)]^2$.

In this paper, we demonstrate numerically a much stronger property: *$\tilde{Q}_n$ is ϵ-factorizable in the whole hypercube in v*, even for small time differences.

B. Integration in the simplex

After obtaining the factorization (51), we perform the integral over the times u. For steady-state calculations ($t \to \infty$), we perform the integration in the full v hypercube; see Eq. (50b). For calculations at finite t, we need to integrate over the simplex in v defined by Eq. (50b).

The integration is carried out as a postprocessing step after the factorization, which is the most time-consuming task. A *single* tensor train interpolation is sufficient to obtain the full curve $Q_n(t)$ for $0 \le t \le t_M$, and for any $\lambda(t)$. We proceed as follows; Appendix G contains further details. We define the one-dimensional functions Ψ by

$$\Psi_n(x) = \int_0^x dy \lambda(y) T_n(x - y) \quad (53)$$

and

$$\Psi_p(x) = \int_0^x dy \lambda(y) T_p(x - y) P_p^{-1} \Psi_{p+1}(y) \quad (54)$$

for $p < n$. We then have

$$Q_n(t) \approx Q_n^{\mathrm{TCI}}(t) = \Psi_1(t). \quad (55)$$

An alternative method using a Fourier transform to perform the integration in v is presented in Appendix H. However, the direct approach above is more general [arbitrary $\lambda(t)$] and numerically faster, so it is preferred in practice.

Calculations in the steady state ($t \to \infty$) limit or at large t_M present a specific difficulty. The tail of $\tilde{Q}_n(v_i)$ at large v_i is small, so it has little effect on the factorization but can make a significant contribution to the integral because of its large volume. This problem is addressed by the env error function $\epsilon_\Pi^{\mathrm{env}}$. Alternatively, this problem could be solved manually in simple cases using a second change of variables mapping $[0, t_M]$ onto [0, 1]:

$$w_i(v_i) \equiv 2v_i/(t_M + v_i). \quad (56)$$

Since this change of variables is diagonal, it does not affect the factorizability of the tensor. However, decomposing the function in w_i introduces a large weight $dv_i/dw_i = 2t_M/(2 - w_i)^2$ in the tail region from the Jacobian, which affects the choice of pivots. These techniques are illustrated in Sec. VI E.

C. Error estimation

In this section, we present a practical error monitoring scheme for TCI calculations. The TTD method has three main sources of errors, which are controlled by χ (factorization error), d (discretization error), and t_M (time truncation error; only for steady state calculations), respectively.

The *factorization error*, which comes from the approximation of the integrand by a tensor train (51), is the most important source of error. We use an estimator of this factorization error defined, for a given value of χ, as the maximum over the sweep at rank χ of the maximum error ϵ_Π (respectively, $\epsilon_\Pi^{\mathrm{env}}$) in the regular (respectively, env) variant of the TCI algorithm. In machine learning terminology, this is an *in-sample error*, since it is computed solely from the data used in the construction of the approximation. It is, nevertheless, a conservative in-sample error, as we actively seek pivots with large values of the error. In Fig. 7(a), we show this estimator for the SIAM model with $\epsilon_d = 0$, for which the Bethe ansatz solution gives high-accuracy benchmarks. In this case, the error

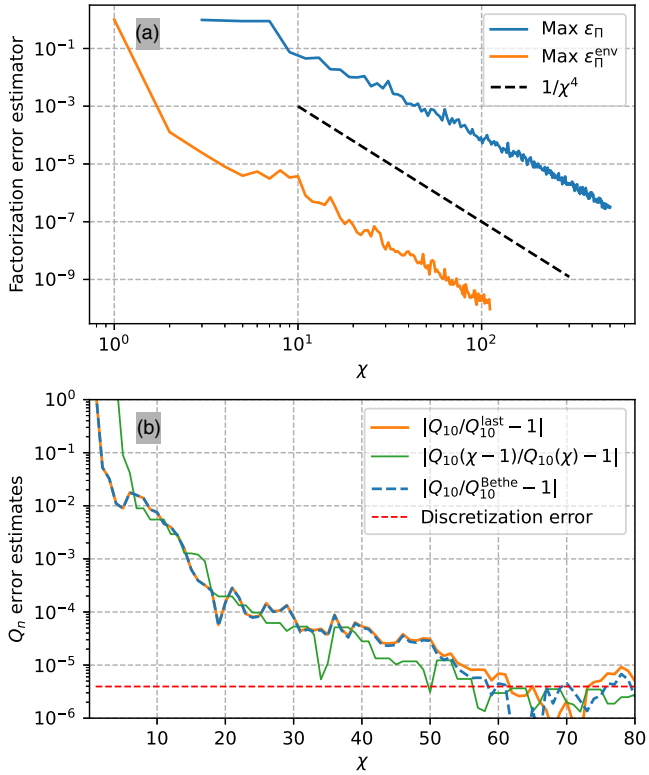

FIG. 7. (a) Upper: factorization error estimator as a function of the tensor rank χ, for the SIAM model, for $n = 10$, $\epsilon_d = 0$, $t_M = 15$, Gauss-Kronrod points with 63 points, using the error functions $\epsilon_\Pi^{\rm env}$ (orange curve) and ϵ_Π (blue curve); see the text. (b) Lower: error of the integral Q_n versus χ, measured using different estimators. In this case, $Q_{10}^{\rm last}$ is Q_{10} for $\chi = 100$. The red line is the error due to the integral discretization obtained by varying the number of integration points.

decays quickly as $\sim 1/\chi^4$. This fast decay is the signature of ϵ-factorizability.

The error on the actual physical quantity Q_{10} in the SIAM model with $\epsilon_d = 0$ is shown in Fig. 7(b). We compare different estimates of the error: the exact error (known here from the Bethe ansatz solution but unavailable, in general), the *running relative error* between $Q_n(\chi - 1)$ and $Q_n(\chi)$, where $Q_n(\chi)$ denotes the approximation of Q_n at rank χ, and the error with respect to the largest value of χ used, available only at the end of the calculation. We find that these estimators are in excellent agreement. In practice, the running relative error yields a satisfactory estimate of the true error. The saturation observed for large χ is the result of discretization error.

Apart from the factorization error, three more sources of error need to be controlled. First, the *discretization error* stems from the one-dimensional integration and is determined by the number of integration points d. In highly oscillatory cases, one may require a specialized integration technique, as discussed in Sec. VII A. Second, for calculations at infinite time in the steady state $t_M = \infty$, the convergence with t_M must be verified as well. Third, in some cases, *rounding errors* can become significant, in particular, when very high precision (approximately 10^{-8} or smaller) is sought with large expansion orders n. These result from cancellations in the summation over Keldysh indices in the calculation of $\tilde{Q}_n$ (see Appendix D).

In practice, monitoring the different errors discussed above is sufficient. Nevertheless, in order to illustrate the quality of the TCI approximation, we now also present an *out-of-sample error* estimate, obtained as follows. Starting from a random point $(v_1, \ldots, v_n)$, we sweep over the variables $v_1, v_2, \ldots, v_n$ several times. At the ith step of each sweep, we update v_i to maximize the error, with all other variables fixed. This eventually yields a point $(v_1^*, \ldots, v_n^*)$ that maximizes the error locally. We find that the value of the error obtained by this procedure is robust with respect to the starting point. The results are illustrated in Fig. 8, which shows one-dimensional slices of $\tilde{Q}_7$ along every possible direction starting from v_i^*. The blue circles indicate the position of v_i^*. The TCI approximation is already qualitatively correct for $\chi = 2$, quantitatively correct for $\chi = 4$, and indistinguishable from the exact solution

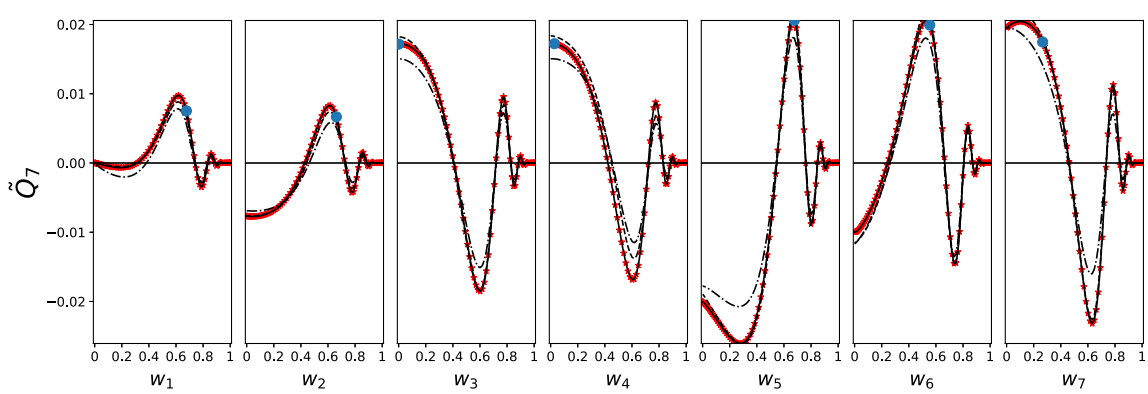

FIG. 8. Comparison of the integrand $\tilde{Q}_7$ for $\epsilon_d = -2$ (red stars) with the TCI of rank $\chi = 2$, 4, 40 (dash-dotted, dashed, and solid black lines, respectively). The function is plotted in the w variable (56) with $t_M = 5$. The maximum error point is (0.71,0.62,0.0,0.07,0.45,0.60,0.25) (blue circles). Each panel corresponds to the variation of one variable w_i starting from this point.

for $\chi = 40$. Despite the presence of strong oscillations (the average sign is 10^{-3} in this case), the maximum relative error that we observe for this calculation is of the order of 1%, only one order of magnitude larger than the in-sample factorization error.

VI. RESULTS FOR THE SIAM MODEL

In this section, we present comprehensive numerical results for the SIAM model in various regimes, in order to illustrate the practical performance of TTD.

A. High-precision benchmarks using the Bethe ansatz

We begin with calculations obtained using TTD on the SIAM model at $\epsilon_d = \bar{\alpha} = 0$ and $t_M \to \infty$. One primary motivation for studying this particular regime is the existence of an analytic Bethe ansatz solution [33,44], from which we can extract the perturbative expansion with arbitrary accuracy. Such a high-precision benchmark is essential to study convergence and the scaling of errors using different parameters and variants of the algorithm. In most figures in this section, we show the relative error $\epsilon_n^Q \equiv |Q_n - Q_n^{\mathrm{Bethe}}|/|Q_n^{\mathrm{Bethe}}|$ of Q_n calculated by TTD, compared with the exact Bethe solution.

The error ϵ_n^Q is presented in Fig. 9(a) as a function of the rank χ of the tensor train. The total number N of $\tilde{Q}_n$ evaluations scales as $N \sim nd\chi^2$, so the different dashed lines correspond to different scalings ranging from $1/\chi$ ($\sim 1/\sqrt{N}$, the scaling of Monte Carlo calculations) to $1/\chi^8$ ($\sim 1/N^4$). Remarkably, one can reach very high precision—better than 10^{-8}—even for large perturbation orders $n \sim 20$, using a moderate value of χ. We observe an effective scaling of the error as $1/N^2$ for Q_{10}, Q_{15}, and Q_{19} and a faster scaling for lower orders. The exact asymptotic scaling is unknown. In Ref. [7], a stretched exponential is observed at very high precision. This behavior cannot be excluded by our data. In any case, the convergence we observe is dramatically faster than that of Monte Carlo methods, which scales as $1/\sqrt{N}$, or even of quasi-Monte-Carlo techniques [33], which are at best approximately $1/N$ in favorable cases but much less robust.

The error is presented as a function of N in Fig. 9(b) for lower-precision calculations, $\epsilon_n^Q \geq 10^{-5}$, for $t_M = 15$. Here, we use a coarser discretization (smaller d) than that used in Fig. 9(a), which is less costly but limits the accuracy. We see that 10^6 evaluations of $\tilde{Q}_n$ are sufficient to obtain an error below 10^{-4} in all cases. We include the total computation time for three points, using a single core on a recent modern workstation, e.g., 17 seconds for Q_{10} with four-digit accuracy. Reaching the same accuracy using

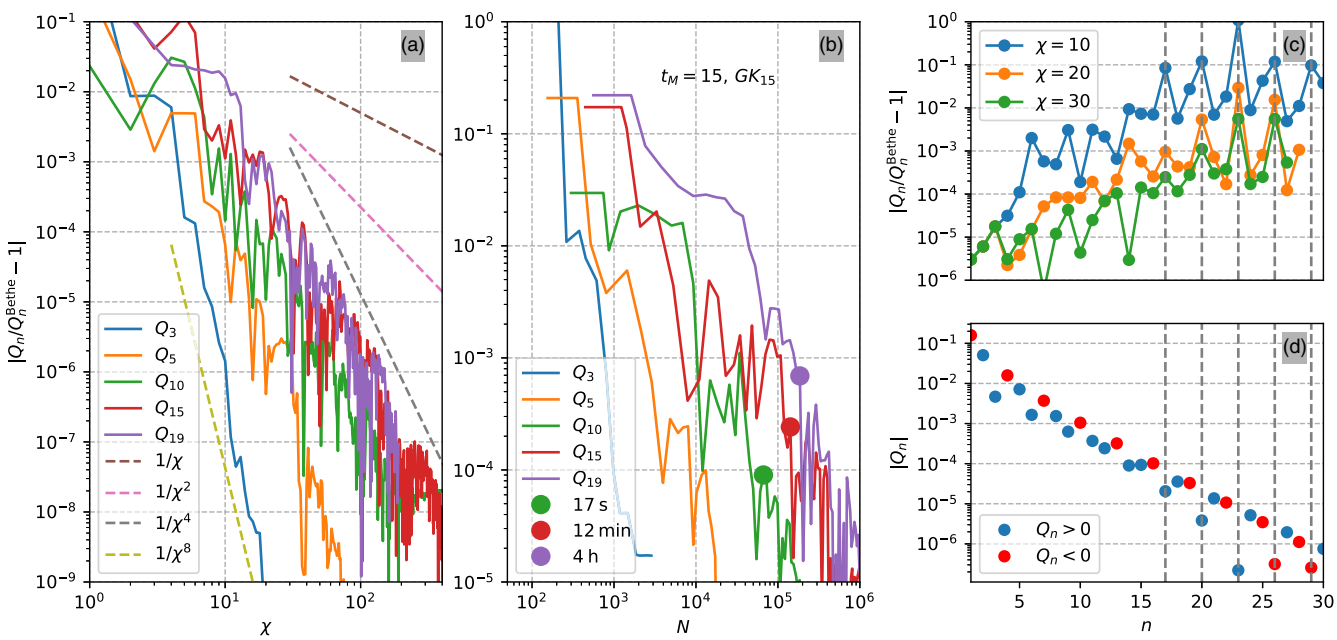

FIG. 9. Relative error of the coefficients Q_n with respect to the exact solution for $n = 3$ (blue), $n = 5$ (orange), $n = 10$ (green), $n = 15$ (red), and $n = 19$ (purple). All calculations are carried out using the env and alternate search variants of the algorithm. (a) Error versus tensor rank χ. Large values of t_M and d are used so that the accuracy is limited only by χ: $t_M = 20$, CH_{255} ($n = 3$), $t_M = 25$, CH_{255} ($n = 5$), and $t_M = 30$, GK_{63} ($n = 10$, $n = 15$, $n = 19$) The dashed lines are guides to the eye for $1/\chi^p$ scaling, which corresponds to $1/N^{p/2}$ in terms of the number N of function calls. $p = 1$ corresponds to Monte Carlo scaling. (b) Error versus N in a low-precision calculation with $t_M = 15$ and $d = 15$ (GK_{15}). The colored circles indicate the CPU time of the calculation when performed on a single core. (c) Error versus n for $\chi = 10$ (blue), $\chi = 20$ (orange), and $\chi = 30$ (green) ($t_M = 30$ and GK_{15}). (d) $|Q_n|$ versus n, distinguishing positive and negative coefficients. The vertical lines indicate values of n where $|Q_n|$ is very small, leading to larger relative errors.

our previous Monte Carlo implementation [25] would require thousands of CPU hours.

The error of Q_n is plotted against n in Fig. 9(c), up to $n = 30$, and $|Q_n|$ is plotted in Fig. 9(d). We note that we can obtain Q_{30} with better than two-digit accuracy in approximately 10^4 CPU hours, a significant improvement over our previous works ($n = 10$–15 using Monte Carlo [25] and $n = 20$–22 using quasi-Monte-Carlo [33]).

Crucially, we observe that the ϵ-factorizability does not deteriorate significantly with increasing n: In Fig. 9(a), we observe a similar error for $n = 10$, 15, and 19; in Fig. 9(c), the error stabilizes after $n = 15$ (note that the indicated values at which the relative error is peaked simply correspond to values of n for which Q_n is particularly small).

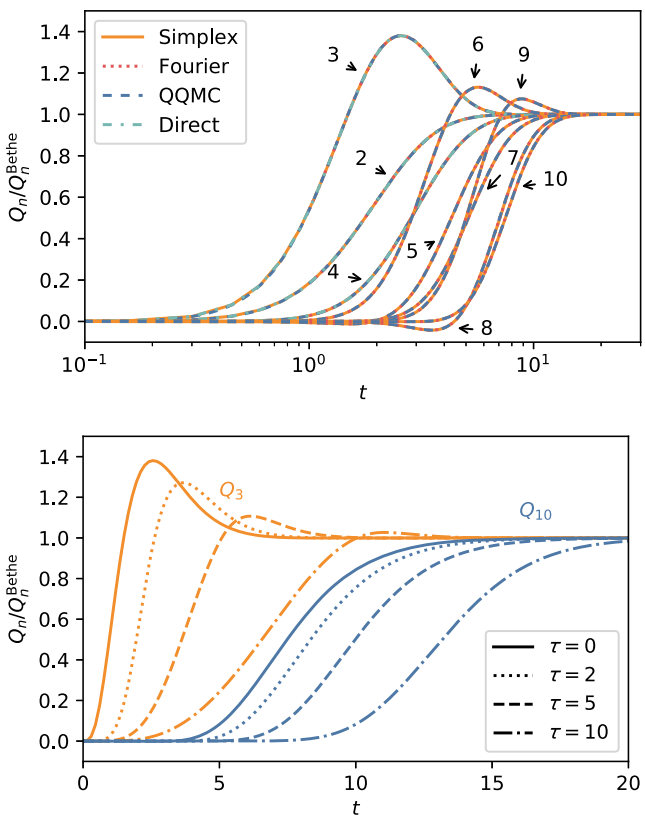

FIG. 10. $Q_n(t)$ after a quench at $t = 0$ for the SIAM model at $\epsilon_d = \bar{\alpha} = 0$, in equilibrium. The coefficients are normalized by their exact asymptotic values $Q_n(t = \infty) = Q_n^{\mathrm{Bethe}}$. Upper: abrupt quench $\lambda(t) = \theta(t)$. Different curves correspond to different integration techniques, with indistinguishable results: simplex integration (55) (yellow continuous line), Fourier technique (H4) (red dotted line), quantum quasi-Monte-Carlo (QQMC) [33] (blue dashed line), and direct numerical integration of Eq. (42) for $n = 2, 3, 4$ (green dashed-dotted line). The arrows indicate the value of n. Lower: $Q_n(t)$ for $n = 3$ (orange) and $n = 10$ (blue) using a continuous increase of the interaction strength $\lambda(t) = \sin(\pi t/2\tau)$ for $t < \tau$, and $\lambda(t) = 1$, for $t \geq \tau$ (with simplex integration).

B. Real-time dynamics

As explained in Sec. V B, the full time dependency of the charge $Q(t)$ and its expansion coefficient $Q_n(t)$, after switching on the interaction at $t = 0$, can be obtained from a *single* factorization of $\tilde{Q}_n$ at negligible additional cost. Figure 10 (upper) shows an example of $Q_n(t)$ curves for different orders n. At large t, each $Q_n(t)$ converges toward the Bethe ansatz equilibrium value.

The time integration (53) and (54) can be performed with any time-dependent coupling constant $U\lambda(t)$ at a negligible increase in cost. In Fig. 10(b), we show $Q_n(t)$ in two cases: (i) an abrupt turning on of the interaction $\lambda(t) = \theta(t)$ and (ii) a continuously differentiable $\lambda(t)$ given by

$$\lambda(t) = \begin{cases} \sin(\pi t/2\tau) & 0 \leq t \leq \tau, \\ 1 & t > \tau, \\ 0 & t < 0. \end{cases} \tag{57}$$

The ability to quickly calculate the effect of any time-dependent coupling constant suggests interesting possibilities for studying the effect of various types of quenches. It might also be used to optimize the convergence to the asymptotic value. Indeed, although the series (2) has an infinite radius of convergence for any *finite* time [29], a very high-order expansion may still be required at intermediate and long times. An interesting open question is whether a smooth adiabatic turning on of the interaction could lead to an easier resummation of the perturbative series (i.e., using fewer terms) at intermediate times than an abrupt quench, which puts the system far out of equilibrium.

In Fig. 11, we show an example of the actual physical observable $Q(U, t)$ for two values of the switching time τ and $U = 2$. These results are obtained by truncating the

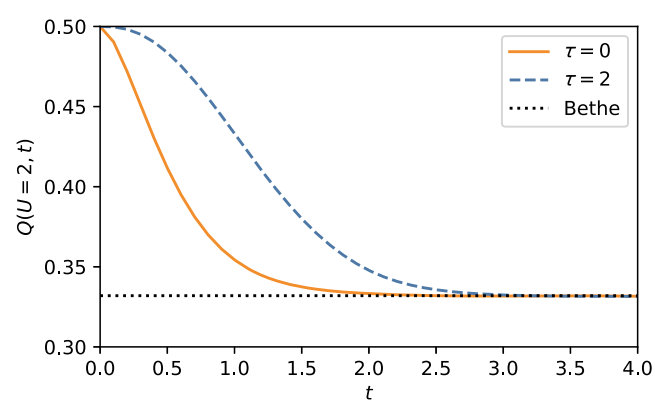

FIG. 11. Charge $Q(U, t)$ on the dot for the SIAM model after switching on the interaction abruptly (yellow solid line, $\tau = 0$) or smoothly (blue dashed line, $\tau = 2$) with $\lambda(t \leq \tau) = \sin(\pi t/2\tau)$ and $\lambda(t \geq \tau) = 1$. Both curves include terms $Q_n(t)$ up to $n = 14$. The black dotted line corresponds to the resummed series using the Euler transform of the first 20 Q_n^{Bethe} coefficients, as in Ref. [25]. In all calculations, we use $\epsilon_d = \bar{\alpha} = 0$ with $U = 2, t_M = 20$.

series (2) to a finite number of terms, varying the expansion order to check convergence. Both curves converge to the asymptotic Bethe ansatz value with high accuracy. Note that $U = 2$ is beyond the radius of convergence of the series for $t \to \infty$, so the two curves can be obtained only up to a finite time without resummation [29].

C. Factorizing the sign problem

In the previous benchmark, the integral is fairly non-oscillatory. When ϵ_d is nonzero, $\tilde{Q}_n$ oscillates much more, a challenge for high-dimensional integration techniques like Monte Carlo and quasi-Monte-Carlo. We illustrate the issue using a simple toy function

$$A(v_1, \ldots, v_n) = \prod_{i=1}^{n} (1 + a \cos 2\pi v_i) \tag{58}$$

defined on the hypercube $[0, 1]^n$, which is completely factorizable. A direct Monte Carlo estimator of the integral of A using N_{MC} random points is given by $\bar{A} = (1/N_{\mathrm{MC}}) \sum_{\alpha=1}^{N_{\mathrm{MC}}} A(v^\alpha)$. The variance of $\bar{A}$ is exponentially large in n:

$$\mathrm{var}\bar{A} = \frac{1}{N_{\mathrm{MC}}} \left[\left(1 + \frac{a^2}{2} \right)^n - 1 \right]. \tag{59}$$

By contrast, using the TTD of A simply requires computing n one-dimensional integrals. The central question is then the robustness of ϵ-factorizability in strongly oscillating cases.

A consequence of the presence of oscillations is the well-known sign problem: The *average sign* at order n,

$$\eta_n = \frac{\int du_1 \ldots du_n \tilde{Q}_n(u_1, \ldots, u_n)}{\int du_1 \ldots du_n |\tilde{Q}_n(u_1, \ldots, u_n)|}, \tag{60}$$

may be small as a consequence of cancellations in the integral. If one is interested in maintaining *relative* accuracy, then a small η_n poses an additional challenge, as the absolute precision of the factorization of $\tilde{Q}_n$ must be increased. In our calculations, however, small η_n usually means that the corresponding contribution to the observable is small, and we are interested in *absolute* accuracy (or, more precisely, in relative accuracy with respect to the largest contribution usually found at low order).

Let us now review our empirical observations of the behavior of TTD in the presence of strong oscillations in the integral and a small average sign. The numerator and denominator in the definition (60) of η_{10} are presented in Fig. 12, as a function of ϵ_d, for the SIAM model. The sign η_{10} is of magnitude approximately 1 for $\epsilon_d \approx 0$. Away from $\epsilon_d = 0$, η_{10} decreases, reaching $\eta_{10} \approx 10^{-5}$ for $\epsilon_d = -4$. A Monte Carlo simulation in this regime would be prohibitively expensive. On the other hand, as illustrated in Fig. 1(d), the TCI decomposition is performed with an

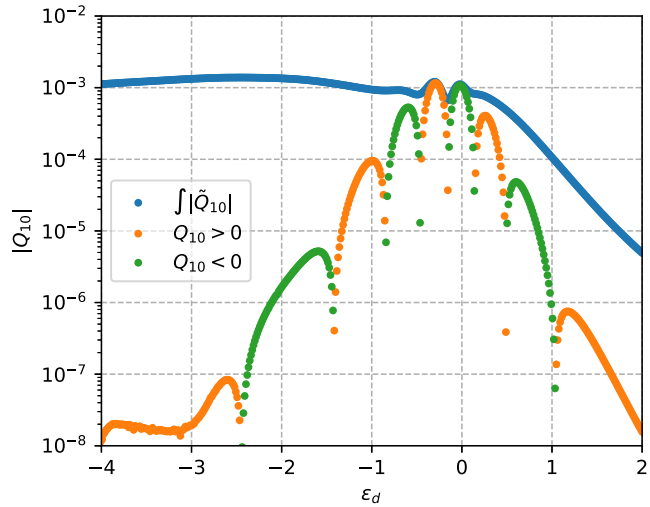

FIG. 12. Integrals $Q_n = \int \tilde{Q}_n$ and $\int |\tilde{Q}_n|$ for $n = 10$ versus ϵ_d for the SIAM model. The green and orange colors correspond $Q_{10} < 0$ and $Q_{10} > 0$, respectively.

approximately constant computational time for every value of ϵ_d, indicating its insensitivity to a small average sign.

Since the ϵ-factorizability is independent of the oscillatory character of the integrand, the difficulty is reduced to that of integrating oscillatory one-dimensional functions, a much less formidable problem. In practice, this may still be a difficult task, and we discuss our approach for specific cases in Sec. VII A and Appendix I.

However, a small average sign leads to a very general and simple issue: The relative error involves division by a small number. In order to keep it at a given level when varying ϵ_d, a smaller absolute error is required by at most $1/\eta_n$. This effect is illustrated in Fig. 13(a), where the relative error of Q_{10} is presented as a function of χ for various values of ϵ_d. The convergence rate is the same $(1/N^2)$ for every ϵ_d, reflecting again that a small average sign does not affect the quality of the ϵ-factorization. However, we observe that a small η_{10} implies a larger relative error. This effect is difficult to predict quantitatively, as it depends on cancellations in integrating the error in Eq. (51), i.e., $\tilde{Q}_n - \tilde{Q}_n^{\mathrm{TCI}}$. In Fig. 13(b), the relative error is plotted as a function of η_{10}, for fixed $\chi = 50$. We observe that it increases approximately like $1/\sqrt{\eta_{10}}$, slower than $1/\eta_{10}$. Since $Q_{10} = O(\eta_{10})$, it follows that the absolute error actually *decreases* when η_{10} gets smaller (not shown). Because of the $1/N^2$ convergence rate, keeping the same relative error when varying ϵ_d therefore corresponds to a moderate increase in computing time $\sim 1/\eta^{1/4}$ in this model. Keeping a constant absolute error is actually easier in the presence of a sign problem.

Since our implementation uses double-precision arithmetic, we cannot go beyond an absolute precision of 10^{-9} in the integrand due to rounding errors in the Keldysh sum of determinants (see Appendix D). This translates to a

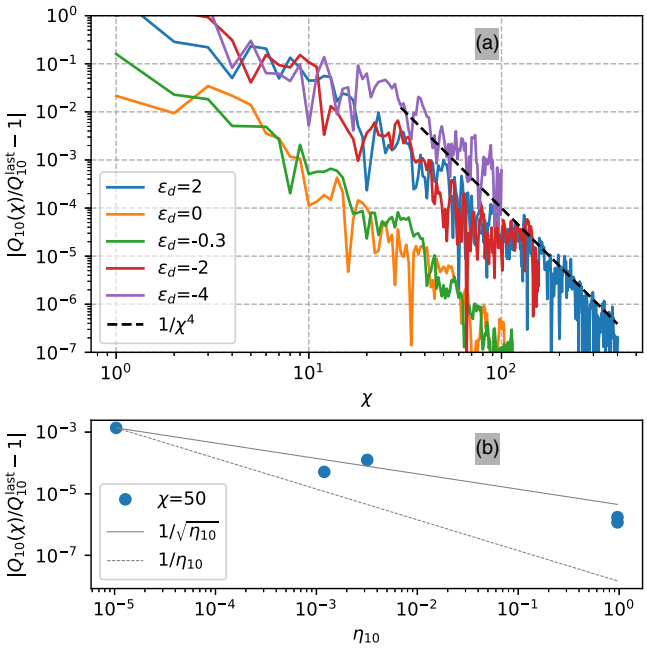

FIG. 13. (a) Upper: error of Q_{10} versus χ in the SIAM model, for different ϵ_d. Gauss-Kronrod integration is used with $d = 31$ points for $\epsilon_d = 2, 0, -0.3$ and $d = 255$ points for $\epsilon_d = -2, -4$. (b) Lower: the same error, but for fixed $\chi = 50$, varying η_{10}.

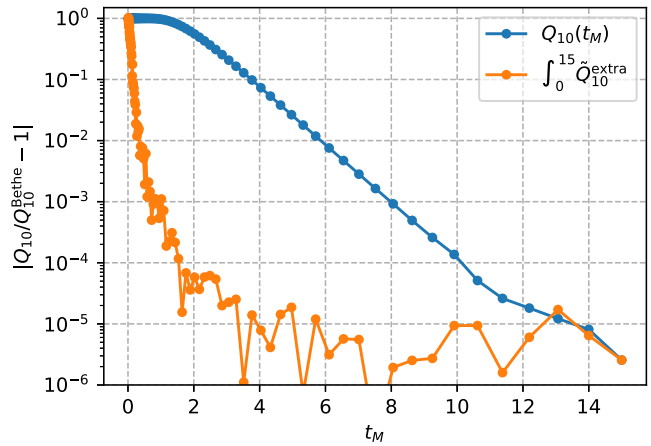

FIG. 14. Relative error with respect to the Bethe ansatz for $Q_{10}(t_M) = \int_0^{t_M} \tilde{Q}_{10}$ and $\int_0^{15} \tilde{Q}_{10}^{\rm extra}$, versus t_M. $\tilde{Q}_{10}^{\rm extra}$ is the extrapolation of the TCI of $\tilde{Q}_{10}$ obtained for $v_i \le t_M$. We used Gauss-Kronrod rule with 63 points for the integration. The error saturates at 10^{-5} due to the discretization error.

relative error in Q_{10} of 10^{-4} for the worst case $\epsilon_d = -4$. Beyond this point, the error saturates.

In sharp contrast with Monte Carlo, computing $\int |\tilde{Q}_n|$ with TTD is, in fact, significantly *harder* than computing $\int \tilde{Q}_n$. Indeed, the function $|\tilde{Q}_n|$ is not ϵ-factorizable with a low rank, most likely as a result of the cusps introduced by taking the absolute value.

This section illustrates a central point of this paper: The property of the integrand $\tilde{Q}_n(v_1, \ldots, v_n)$ that makes the problem amenable to integration with TTD (ϵ-factorizability) is orthogonal to the property that would make it amenable to a solution with Monte Carlo sampling (positivity). In particular, TTD works seamlessly in some situations in which Monte Carlo fails. This is a strong incentive to revisit problems that suffer from a strong sign problem in Monte Carlo algorithms using the TCI algorithm.

D. Extrapolation versus interpolation

In this section, we discuss a remarkable feature of our integrand discovered by the TCI decomposition. As mentioned earlier, the crucial and nontrivial property of $\tilde{Q}_n$ is the ϵ-factorizability of the *core* of the function (e.g., with all variables confined to a small pocket $v_i \in [0, 1]$), while the factorizability at large v is easier to understand. It turns out that this core factorization is also an excellent *extrapolation* at large v (e.g., $v_i \in [0, 8]$). The factorization of $\tilde{Q}_n$ at short times (difference) is not only possible, it is, in fact, sufficient to approximate the whole function.

We illustrate this observation with Fig. 14, where two calculations of Q_{10} are presented. The first one (blue curve) is simply $Q_{10}(t_M) = \int_0^{t_M} \tilde{Q}_{10}$, the direct hypercube integral. At large t_M, as expected, the error with respect to the stationary value decreases quickly. The second calculation consists in computing the integral for the hypercube $v_i \le 15$, but with a TCI approximation computed for smaller hypercube $v_i \le t_M \le 15$, i.e., with the pivots confined to $[0, t_M]^n$. The second computation converges to the equilibrium value much faster than the first one. In summary, in order to obtain a precision of three digits, the integration must be done on a large volume $[0, 8]^n$, but it is sufficient to perform the learning part inside a volume $[0, 1]^n$, which is exponentially smaller with n.

E. Effect of various parameters and variants on the convergence

We next examine the convergence of TTD for different parameter choices and variants of the algorithm. Results for the error ϵ_{10}^{Q} are summarized in Fig. 15.

We first compare the factorizability in the u, v, and w variables given in Eqs. (49) and (56). Figure 15(a) shows the error in the factorization of $\tilde{Q}_{10}$ for the three choices of variables, using the pivot error function ϵ_Π of Eq. (23). We observe that $\tilde{Q}_{10}(u_1, \ldots, u_n)$ *is not* ϵ-factorizable (green curve). As discussed above, this is likely a consequence of the cusps on the boundaries between the $n!$ different smooth components of the function, corresponding to different orderings of the u_i. By contrast, the error decreases quickly when the v variables are used, and using the w variables gives a further reduction by 2 orders of magnitude. The observed saturation around 10^{-6} is a consequence of making the cutoff $t_M = 15$, as shown below. The factorizability is the same in v and w, and it would

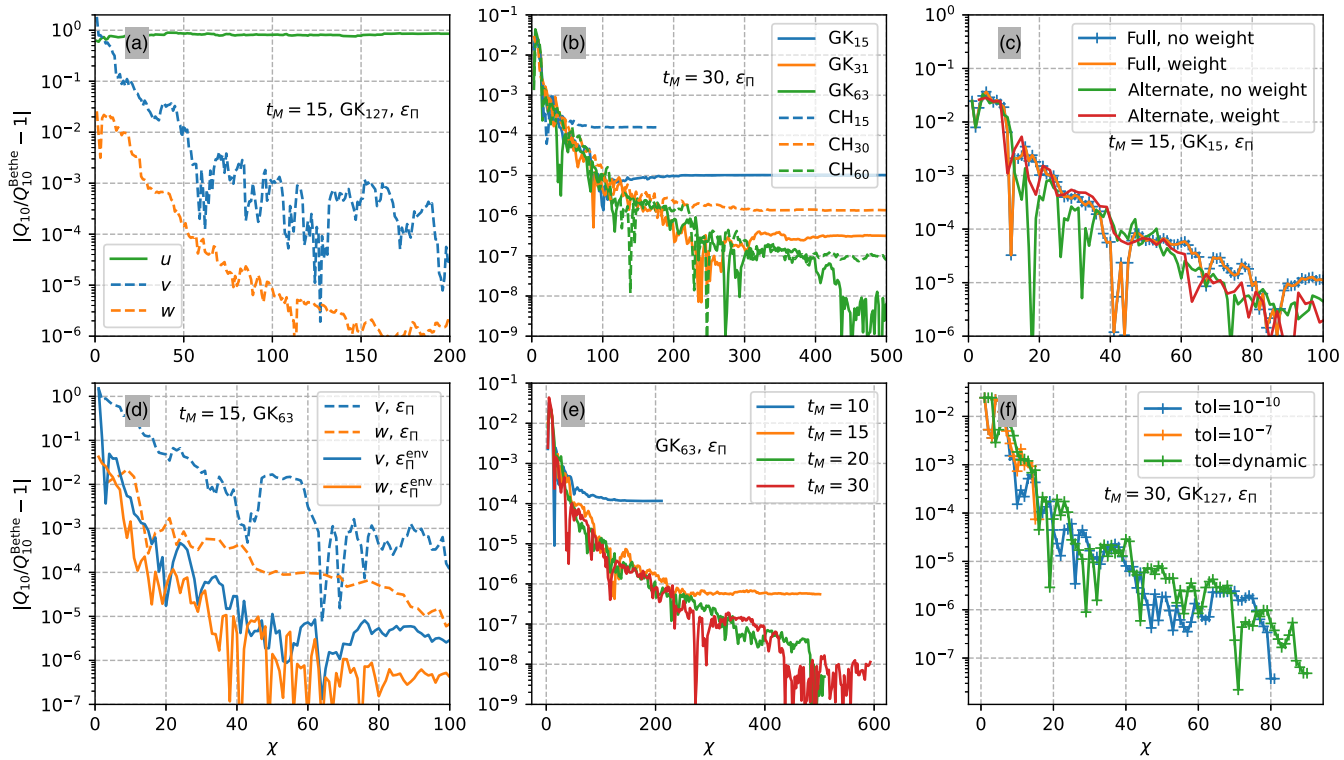

FIG. 15. Convergence with respect to the tensor rank χ of the ten-dimensional integral Q_{10}, for the SIAM model. All panels show the relative error $\epsilon_{10}^{Q} = |Q_{10}(\chi)/Q_{10}^{\mathrm{Bethe}} - 1|$ measured against the Bethe ansatz solution versus the bond dimension χ. (a) Comparison of the choices of variables u, v, and w. (b) Comparison of different one-dimensional quadrature rules of d nodes: Gauss Kronrod (GK_d) and Chebyshev (CH_d). (c) Comparison of different pivot selection algorithms, *full search* and *alternate search*, defined in Sec. III B 3, and the *weighted learning* variant defined in Sec. III B 5. (d) Effect of using ϵ_Π (23) or $\epsilon_\Pi^{\mathrm{env}}$ (30) as the error function in the pivot selection. (e) Saturation of the error due to the maximum time cutoff t_M. (f) Absence of a dependence on the pivot acceptance condition; see the text.

provide the same approximation if the same pivots had been selected. The use of w clearly produces better pivots.

Figure 15(d) establishes that using the error function $\epsilon_\Pi^{\mathrm{env}}$ defined in Eq. (30) removes the need for introducing a problem-dependent change of variable w. We use this function for all computations in this paper, unless otherwise specified.

Figure 15(b) illustrates that the discretization of the one-dimensional integrals limits the overall accuracy. The different curves are essentially on top of each other until the number of points becomes a limitation in the precision. The closeness of the curves before this limit is reached suggests a robustness relative to the precise position of the pivots, which are different for the different curves, since they are chosen from different grids. We also see that, for this model, Gauss-Kronrod integration has slightly better convergence properties than Chebyshev integration and has the additional advantage of providing a built-in estimate of the integration error. We have also tried Gauss-Legendre quadrature rules (not shown), with similar convergence to Gauss-Kronrod.

In Fig. 15(e), we show the error of the steady state value with respect to the maximum time t_M, since the interaction is switched on in the Keldysh formalism. A large choice of t_M is required for high accuracy. We note that increasing t_M may also require increasing d to maintain the accuracy of one-dimensional integrals.

In Fig. 15(c), we test the effect of using the alternate search and full search pivot selection methods, defined in Sec. III B 3. We find no significant difference in the result, so for all calculations in this paper we use the alternate search approach, which has computational complexity $\propto n\chi^2 d$ rather than $\propto n\chi^2 d^2$.

Figure 15(f) illustrates the robustness of the TCI algorithm with respect to different criteria to accept pivots. We introduce a pivot acceptance condition: We accept a new pivot only if the error ϵ_Π is above a given threshold, i.e., not adding pivots that improve the error only marginally. Figure 15(f) shows the convergence for two different levels for this threshold as well as a dynamical algorithm where the threshold is fixed to 1% of the current typical pivot error ϵ_Π. We observe no significant effect. For very large bond dimensions ($\chi > 10^3$), where the contraction of the tensor train might require a significant computing time, using such a condition might become useful. However, for the rather small values of χ used in this article, the gain is marginal.

VII. PRELIMINARY STUDIES BEYOND THE SINGLE-IMPURITY MODEL

The next step, after the benchmarks on a single-impurity model, is to generalize the TTD method to more complex systems like multisite and lattice models and to other perturbative expansions. In this section, we take the first steps in this direction with two preliminary studies which indicate that the ϵ-factorizability property is robust beyond a single-site model in a flat bath.

A. Single impurity embedded in a two-dimensional lattice

We first consider a single-site model with a more complex bath than the SIAM: an infinite 2D lattice in which a single site is interacting, defined in Sec. IV B 2. As a result of the band edges, the noninteracting Green's functions have strong oscillations at a frequency set by the bandwidth (see lower panels in Fig. 6), so $g^<(t) \sim \cos(4t)/t$ is both highly oscillatory and slowly decaying.

The upper panels in Fig. 16 show a one-dimensional slice of the integrand $\tilde{Q}_5$, demonstrating strong oscillations with several harmonics of $\omega_0 = 4$ present. Such a calculation would be very challenging for Monte Carlo approaches [25]. Nevertheless, the TCI approximation, also shown in Fig. 16, works well, indicating ϵ-factorizability despite the strong oscillations (note, however, that $\chi = 150$).

In the lower panels in Fig. 16, we show the first 15 coefficients of the interaction expansion of the charge. The rapid oscillation and slow decay of the integrand is so severe that even the calculation of the one-dimensional integrals in TTD is nontrivial. While for the SIAM a small cutoff $t_M = 15$ is sufficient to obtain several digits of accuracy, we find in this case that $t_M = 10000$ is required to go beyond two digits or even a single digit at large n. To perform these one-dimensional integrals efficiently, we use a specifically tailored quadrature rule, described in Appendix I, which makes use of an asymptotic expansion of the integrand. The interval of integration is broken into a short time region [0, 10], on which we use a 63-point Clenshaw-Curtis rule, and a large time region $[10, t_M]$, on which the custom quadrature rule is used. We note that the slow decay with t_M is specific to the $T = 0$ case. We check that, at a higher temperature $T = 0.1$, convergence is reached for a much smaller $t_M \approx 10$ (not shown). However, the factorizability appears to be independent of temperature.

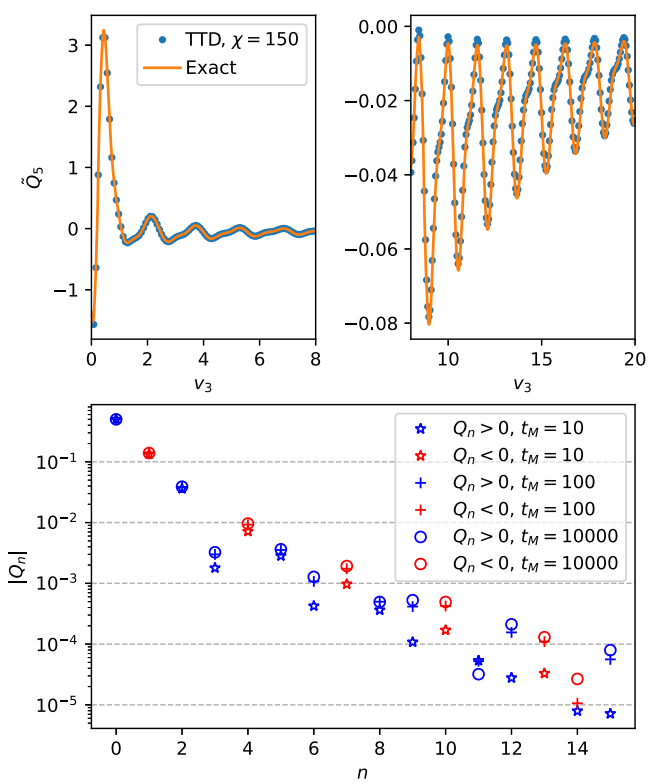

FIG. 16. One interacting site in an infinite two-dimensional lattice. Upper: integrand $\tilde{Q}_5(v_1, v_2, v_3, v_4, v_5)$ versus v_3 for a random choice of the values v_1, v_2, v_4, and v_5. The right panel enlarges the tail. Actual integrand (orange curve) and TTD approximation at $\chi = 150$ (blue circles) are both shown. Lower: $|Q_n|$ versus n for three values of the maximum time, $t_M = 10$ (stars), $t_M = 100$ (pluses), and $t_M = 10000$ (circles). Blue (red) symbols correspond to positive (negative) values of Q_n.

B. Double quantum dot

We next consider a double quantum dot, i.e., with *two interacting sites*. This system plays a central role in various approaches to semiconducting qubits. Apart from its intrinsic importance in mesoscopic physics, it is the simplest case in which the perturbative expansion involves sums over both spatial indices $x_i \in \{0, 1\}$ and time differences v_i, since $\tilde{Q}_n(x_1, \ldots, x_n, v_1, \ldots, v_n)$ now depends on both. It is, therefore, a good starting point from which to extend the TTD to a function of space and time.

We first emphasize that there are multiple ways to include spatial indices in the tensor network form. Our goal is to find the one with the lowest rank χ and the best

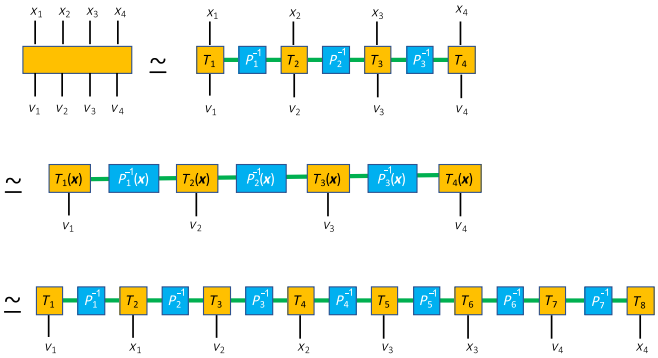

FIG. 17. Three different factorizations used for the double quantum dot problem. From top to bottom: vertex factorization, time factorization, and full factorization.

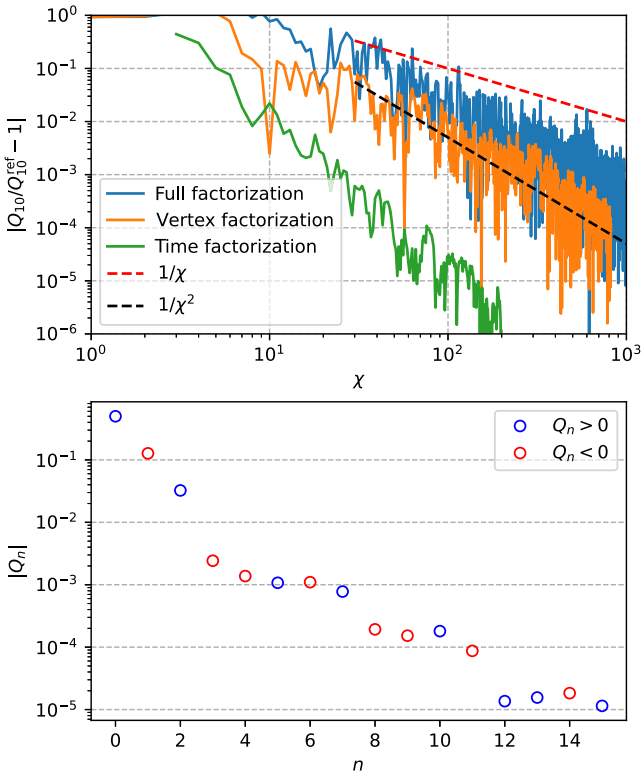

FIG. 18. Double quantum dot. Upper: error of Q_{10} versus χ for the three algorithms introduced in the text. The reference value Q_{10}^{ref} is obtained using the time factorization algorithm. Lower: $|Q_n|$ versus n, obtained using the vertex factorization algorithm. Blue (red) symbols correspond to positive (negative) values of Q_n. We use $\gamma_0 = 0.5\Gamma$ and $t_M = 15$.

convergence rate with N. We study three possibilities, depicted in Fig. 17. Corresponding results are presented in Fig. 18.

First, we can take T tensors that depend on both a spatial and a time variable, $T_\alpha(x_\alpha, v_\alpha)$ (first line in Fig. 17). We refer to this approach as *vertex factorization*, since the factorization is done vertex by vertex. The computational cost is increased only by a factor of 2 compared to the SIAM and scales linearly with the number L of dots ($L = 2$ here), as $\propto Ld$. In Fig. 18 (orange curve), we observe that this method converges quite quickly, as $1/N$, but more slowly than the SIAM, which is $1/N^2$.

Second, we can fix the spatial indices $x_1, \ldots, x_n$ and use TTD for the times (second line in Fig. 17). We refer to this approach as *time factorization*. After integrating over times, we obtain an intermediate function $\hat{Q}_n(x_1, \ldots, x_n)$ given by

$$\hat{Q}_n(x_1, \ldots, x_n) \equiv \int \prod_i dv_i \tilde{Q}_n(x_1, \ldots, x_n, v_1, \ldots, v_n). \quad (61)$$

The summation over the x_i can be carried out in two ways. We can explicitly simply sum over the L^n combinations, with an exponential computational scaling $\propto L^n d$, which is manageable for $L = 2$. In Fig. 18 (green curve), we observe that this method converges as $1/N^2$, like the SIAM. We also observe this convergence rate for each fixed set of spatial indices x_i. Alternatively, we can use TCI again on the spatial variables to factorize $\hat{Q}_n$. We find (not shown) that this approach also converges. However, for the small value $L = 2$, the $\hat{Q}$ tensor is not large enough to draw a definite conclusion on the performance of this technique for large L.

Third, we can use an MPS form, with alternating v and x variables (third line in Fig. 17). The computational cost of this approach is essentially the same as that for the SIAM, with d replaced by $L + d$. In Fig. 18 (blue curve), we observe that this method converges more slowly with N—only slightly faster than $1/\sqrt{N}$—indicating that the "entanglement" between space and time variables has a nontrivial structure, which is not captured efficiently by this simple tensor train.

Thus, various tensor forms can be used to apply TTD to the double or multiple dots. The three methods presented here are all convergent, but with different rates, and for this example vertex factorization is the most efficient. However, many further possibilities could be explored, e.g., using spatial position differences, different orderings of the variables in the MPS, or a PEPS generalization of the tensor form in space-time. The search for an optimal tensor form for the lattice case is an interesting open question which we leave for future work.

VIII. CONCLUSION

Tensor network methods offer a new approach to high-dimensional integration and, in particular, to computing high-order diagrammatic perturbative expansions. The n-body (bare) correlation functions have a mathematical structure that allows a parsimonious representation in term of a tensor network, which can be efficiently obtained using the TCI algorithm. While a naive direct integration in n dimensions would scale exponentially with n, the TCI algorithm can reveal the underlying structure and perform the sum in a number of calls of the integrand that scales linearly with n. We illustrate this approach for quantum impurity models (single and double dots) within the real-time Schwinger-Keldysh formalism, with high-precision benchmarks. It significantly outperforms previous Monte Carlo and quasi-Monte-Carlo methods. In particular, it is insensitive to the infamous sign problem appearing in parameter regimes in which the integrals are highly oscillatory. Furthermore, it allows calculations of the full time dependency and of the effect of a time-dependent coupling constant, at negligible additional cost.

The main open question at this stage is the generality of the ϵ-factorizability property and its potential application to other diagrammatic techniques, e.g., for multiorbital or lattice models, imaginary time perturbative expansions, and

inchworm algorithm in real or imaginary time [45–50]. For example, it is necessary to investigate whether a simple MPS is sufficient to handle the lattice case (with spatial and time indices), or whether a more sophisticated tensor network like PEPS is needed.

We point out, more generally, that the limiting factor of the TCI approach (i.e., the rank of the ϵ-factorization) is entirely orthogonal to that of sampling methods like Monte Carlo (the sign problem). This suggests reexamining various cases (e.g., partition function calculations) which are known to be limited by the sign problem when Monte Carlo methods are used.

ACKNOWLEDGMENTS

O. P., X. W., and P. T. D. thank Miles Stoudenmire for numerous enlightening discussions on tensor network techniques. We thank Fedor Šimkovic and Michel Ferrero for sharing the results of Ref. [43] prior to publication. The Flatiron Institute is a division of the Simons Foundation. X. W. thanks the Plan France 2030 ANR-22-PETQ-0007 and the French-Japanese ANR QCONTROL for funding. T. K. and X. W. acknowledge funding from the European Union's Horizon 2020 research and innovation programme under Grant agreement No. 862683 (UltraFastNano).

APPENDIX A: SCHUR COMPLEMENT

Two important components of this article (the cross interpolation formula and the principal minor algorithm) are based on the concept of Schur complement [51] that we recall here briefly for completeness. We consider an arbitrary matrix A that we put in a 2×2 block form:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}. \tag{A1}$$

It is straightforward to show that, provided the A_{11} block is invertible, one has

$$\begin{pmatrix} 1 & 0 \\ -A_{21}A_{11}^{-1} & 1 \end{pmatrix} \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} 1 & -A_{11}^{-1}A_{12} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} A_{11} & 0 \\ 0 & A_{22} - A_{21}A_{11}^{-1}A_{12} \end{pmatrix}, \tag{A2}$$

from which we obtain that

$$\det A = \det[A_{11}] \det[A_{22} - A_{21}A_{11}^{-1}A_{12}]. \tag{A3}$$

The matrix $A_{22} - A_{21}A_{11}^{-1}A_{12}$ is called the Schur complement of A with respect to the 11 block. We refer to Eq. (A3) as the Schur complement theorem. The 11 block is referred to as the "pivot."

APPENDIX B: PROPERTIES OF THE CROSS INTERPOLATION

1. Proof of property (P2)

We begin with the proof of the property (P2) introduced in the main text, i.e., that if a matrix A is of rank r, then a cross interpolation with $\chi = r$ is exact. Let us consider an arbitrary point (x_0, y_0) and form the $(r+1) \times (r+1)$ block matrix by adding one row and one column to the pivot matrix $A(\mathcal{I}, \mathcal{J})$:

$$\begin{pmatrix} A(\mathcal{I}, \mathcal{J}) & A(\mathcal{I}, y_0) \\ A(x_0, \mathcal{J}) & A(x_0, y_0) \end{pmatrix}. \tag{B1}$$

This submatrix of A has a vanishing determinant. Since the determinant of the pivot matrix is nonzero, applying Eq. (A3) to Eq. (B1) gives

$$A(x_0, y_0) - A(x_0, \mathcal{J})A(\mathcal{I}, \mathcal{J})^{-1}A(\mathcal{I}, y_0) = 0, \tag{B2}$$

which proves property (P2) using (P1).

2. Link between the pivot error and the volume of the pivot matrix

The construction of the previous subsection can also be used to show that, when adding a new pivot to a cross interpolation, looking for the pivot that maximizes the error of the approximant is equivalent to trying to maximize the volume of the new pivot matrix. Indeed, suppose that we have a pivot matrix $A(\mathcal{I}, \mathcal{J})$ and we want to enlarge it with a new pivot (x_0, y_0). Using Eq. (A3), the determinant of the new pivot matrix reads

$$\left| \det \begin{pmatrix} A(\mathcal{I}, \mathcal{J}) & A(\mathcal{I}, y_0) \\ A(x_0, \mathcal{J}) & A(x_0, y_0) \end{pmatrix} \right| = |\det A(\mathcal{I}, \mathcal{J})| \times |A(x_0, y_0) - A(x_0, \mathcal{J})A(\mathcal{I}, \mathcal{J})^{-1}A(\mathcal{I}, y_0)|. \tag{B3}$$

Since $\det A(\mathcal{I}, \mathcal{J})$ is fixed, it follows that maximizing the volume of the pivot matrix (left-hand side of the above equation) is equivalent to finding the pivot (x_0, y_0) where the error of the approximant $|A(x_0, y_0) - A(x_0, \mathcal{J})A(\mathcal{I}, \mathcal{J})^{-1}A(\mathcal{I}, y_0)|$ is the largest.

3. Stable *QR* decomposition for tensor train contractions

During the evaluation of the tensor train approximant, one needs to evaluate expressions of the form $T_\alpha(u_\alpha)P_\alpha^{-1}$. As the tensor train approximation becomes better, the volume of the pivot matrices becomes smaller so that an expression of this type, although mathematically well defined, eventually become numerically unstable. Let us consider the $T_\alpha(i, u_\alpha, j)$ tensor as a matrix T_α of $\chi \times d$ rows indexed by (i, u_α) and χ columns indexed by j. The nesting

condition guarantees that the pivot matrix P_α is, in fact, a submatrix of T_α. Using this structure, we perform a QR decomposition of the T_α matrix, and we get

$$T_\alpha = \begin{pmatrix} P_\alpha \\ T'_\alpha \end{pmatrix} = \begin{pmatrix} Q \\ Q' \end{pmatrix} R, \tag{B4}$$

where T'_α contains all the rows of T_α that are not in P_α. The diagonal of the triangular R matrix contains potentially very small values, while the matrices Q and Q' (which together form a unitary matrix) are well conditioned. Using this decomposition, the product $T_\alpha P_\alpha^{-1}$ can be computed explicitly without usage of the R matrix:

$$T_\alpha P_\alpha^{-1} = \begin{pmatrix} 1 \\ Q' Q^{-1} \end{pmatrix}. \tag{B5}$$

APPENDIX C: ROLE OF THE NESTING CONDITION IN TCI

1. Proof of the interpolation property

In this appendix, we show that the nesting condition (17a) and (17b) implies that the TCI form is a proper interpolation of the tensor A as given by Eq. (18). The proof is done in four steps (I)–(IV).

(I) We note that the nesting property (17a) implies

$$\mathcal{I}_\alpha \subset \mathcal{I}_{\alpha-1} \oplus \mathbb{K}_\alpha, \tag{C1}$$

$$\mathcal{I}_\alpha \subset \mathcal{I}_{\alpha-2} \oplus \mathbb{K}_{\alpha-1} \oplus \mathbb{K}_\alpha, \tag{C2}$$

$$\mathcal{I}_\alpha \subset \mathcal{I}_0 \oplus \mathbb{K}_1 \oplus \ldots \oplus \mathbb{K}_\alpha. \tag{C3}$$

In other words, one can see an element of $\mathcal{I}_\alpha$ as an element of $\mathcal{I}_p$ for $p < \alpha$ concatenated with some u values. Similar relations apply for $\mathcal{J}$.

(II) We reinterpret the three indices tensor T_α by regrouping the left and u index, to obtain a matrix $T_\alpha^{(L)}$ of indices $\mathcal{I}_{\alpha-1} \times \mathbb{K}_\alpha$ and $\mathcal{J}_{\alpha+1}$. This matrix is, in general, rectangular. Because of the nesting condition, a subset of its row indices is, in fact, $\mathcal{I}_\alpha$, and the restriction of $T_\alpha^{(L)}$ to these rows is P_α from the definition of T and P (14). Similarly, we introduce $T_\alpha^{(R)}$ by regrouping the u index and the right index, to obtain a matrix of indices $\mathcal{I}_{\alpha-1}$ and $\mathbb{K}_\alpha \times \mathcal{J}_{\alpha+1}$, and we have

$$T_\alpha^{(L)}(\mathcal{I}_\alpha, \mathcal{J}_{\alpha+1}) P_\alpha^{-1}(\mathcal{J}_{\alpha+1}, \mathcal{I}'_\alpha) = \delta(\mathcal{I}_\alpha, \mathcal{I}'_\alpha), \tag{C4}$$

$$P_{\alpha-1}^{-1}(\mathcal{J}_\alpha, \mathcal{I}_{\alpha-1}) T_\alpha^{(R)}(\mathcal{I}_{\alpha-1}, \mathcal{J}'_\alpha) = \delta(\mathcal{J}_\alpha, \mathcal{J}'_\alpha), \tag{C5}$$

where $\delta(\mathcal{I}_\alpha, \mathcal{I}'_\alpha)$ and $\delta(\mathcal{J}_\alpha, \mathcal{J}'_\alpha)$ are identity matrices.

(III) We write the TCI in the following form, with implicit contraction over repeated indices $\mathcal{I}_\alpha$ and $\mathcal{J}_\alpha$, which highlights the role of the different sets of indices:

$$\begin{aligned} A_{\mathrm{TCI}}(u_1, \ldots, u_n) \approx\ & T_1(\mathcal{I}_0, u_1, \mathcal{J}_2) P_1^{-1}(\mathcal{J}_2, \mathcal{I}_1) \\ & \times T_2(\mathcal{I}_1, u_2, \mathcal{J}_3) P_2^{-1}(\mathcal{J}_3, \mathcal{I}_2) \\ & \times T_3(\mathcal{I}_2, u_3, \mathcal{J}_4) P_3^{-1}(\mathcal{J}_4, \mathcal{I}_3) \ldots . \end{aligned} \tag{C6}$$

(IV) We now fix one value of α and evaluate the TCI form on the pivot indices and u_α, as in Eq. (18). Our goal is to show that the T and P on the left and on the right of T_α cancel. For any multi-index $(u_1^*, \ldots, u_{\alpha-1}^*) \in \mathcal{I}_{\alpha-1}$ and $(u_{\alpha+1}^*, \ldots, u_n^*) \in \mathcal{J}_{\alpha+1}$, we evaluate A_{TCI}:

$$\begin{aligned} & A_{\mathrm{TCI}}(u_1^*, \ldots, u_{\alpha-1}^*, u_\alpha, u_{\alpha+1}^*, \ldots, u_n^*) \\ & = T_1^{(L)}(u_1^*, \mathcal{J}_2) P_1^{-1}(\mathcal{J}_2, \mathcal{I}_1) \\ & \quad \times T_2^{(L)}(\mathcal{I}_1 \oplus u_2^*, \mathcal{J}_3) P_2^{-1}(\mathcal{J}_3, \mathcal{I}_2) \cdots \\ & T_{\alpha-1}^{(L)}(\mathcal{I}_{\alpha-2} \oplus u_{\alpha-1}^*, \mathcal{J}_\alpha) P_{\alpha-1}^{-1}(\mathcal{J}_\alpha, \mathcal{I}_{\alpha-1}) \cdots \\ & T_\alpha(\mathcal{I}_{\alpha-1}, \mathbb{K}_\alpha, \mathcal{J}_{\alpha+1}) \\ & \quad \times P_\alpha^{-1}(\mathcal{J}_{\alpha+1}, \mathcal{I}_\alpha) T_{\alpha+1}^{(R)}(\mathcal{I}_\alpha, u_{\alpha+1}^* \oplus \mathcal{J}_{\alpha+2}) \cdots \\ & P_{n-2}^{-1}(\mathcal{J}_{n-1}, \mathcal{I}_{n-2}) T_{n-1}^{(R)}(\mathcal{I}_{n-2}, u_{n-1}^* \oplus \mathcal{J}_n) \\ & \quad \times P_{n-1}^{-1}(\mathcal{J}_n, \mathcal{I}_{n-1}) T_n^{(R)}(\mathcal{I}_{n-1}, u_n^*). \end{aligned} \tag{C7}$$

Using Eq. (C4), the first line reduces to $\delta(\mathcal{I}_1, u_1^*)$; hence, the second line becomes $T_2^{(L)}((u_1^*, u_2^*), \mathcal{J}_3) \times P_2^{-1}(\mathcal{J}_3, \mathcal{I}_2) = \delta(\mathcal{I}_2, (u_1^*, u_2^*))$ since $(u_1^*, u_2^*) \in \mathcal{I}_2$. The $T^{(L)}$ cancel telescopically from the left until T_α. The same happens from the right, and we obtain finally

$$\begin{aligned} & A_{\mathrm{TCI}}(u_1^*, \ldots, u_{\alpha-1}^*, u_\alpha, u_{\alpha+1}^*, \ldots, u_n^*) \\ & = T_\alpha(u_1^*, \ldots, u_{\alpha-1}^*, u_\alpha, u_{\alpha+1}^*, \ldots, u_n^*) \\ & = A(u_1^*, \ldots, u_{\alpha-1}^*, u_\alpha, u_{\alpha+1}^*, \ldots, u_n^*), \end{aligned} \tag{C8}$$

where we use the definition of T (14) in the last line. This is exactly Eq. (18).

2. Proof of Eq. (24)

Here we prove Eq. (24), i.e., that the error between the Π tensor and its cross interpolation is equal to the global error of the TCI of A on the corresponding subset of points. Let us define $A_{\mathrm{TCI}}^{\Pi_\alpha}$ by the TCI form in which the product $T_\alpha P_\alpha^{-1} T_{\alpha+1}$ is replaced by Π_α in Eqs. (C6) and (C7). In other words, we keep Π_α whole and factorize only the other degrees of freedom. The proof of

Appendix C 1 can be straightforwardly extended to show that, $\forall\,(u_\alpha, u_{\alpha+1}) \in \mathbb{K}_\alpha \times \mathbb{K}_{\alpha+1}$,

$$\begin{aligned} A_{\mathrm{TCI}}(\mathcal{I}_{\alpha-1}, u_\alpha, u_{\alpha+1}, \mathcal{J}_{\alpha+2}) &= T_\alpha(u_\alpha) P_\alpha^{-1} T_{\alpha+1}(u_{\alpha+1}), \\ A_{\mathrm{TCI}}^{\Pi_\alpha}(\mathcal{I}_{\alpha-1}, u_\alpha, u_{\alpha+1}, \mathcal{J}_{\alpha+2}) &= \Pi_\alpha(\mathcal{I}_{\alpha-1}, u_\alpha, u_{\alpha+1}, \mathcal{J}_{\alpha+2}) \\ &= A(\mathcal{I}_{\alpha-1}, u_\alpha, u_{\alpha+1}, \mathcal{J}_{\alpha+2}), \end{aligned}$$

where the last line is due to Eq. (20). From the definition of the error function (23), we get

$$\epsilon_\Pi(i, u_\alpha, u_{\alpha+1}, j) = |A - A_{\mathrm{TCI}}|(i, u_\alpha, u_{\alpha+1}, j) \tag{C9}$$

for $i \in \mathcal{I}_{\alpha-1}$, $u_\alpha \in \mathbb{K}_\alpha$, $u_{\alpha+1} \in \mathbb{K}_{\alpha+1}$, and $j \in \mathcal{J}_{\alpha+2}$.

3. Canonical form and the nested condition

We end this appendix with a short remark. In analogy with the same standard notations in the DMRG literature, we introduce the mixed *canonical* forms of the TCI approximation. Noting

$$\vec{B}_\alpha(u) = T_\alpha(u) P_\alpha^{-1}, \tag{C10}$$

$$\overleftarrow{B}_\alpha(u) = P_{\alpha-1}^{-1} T_\alpha(u), \tag{C11}$$

the TCI approximation can be written in the mixed canonical form centered around T_β:

$$A_{\mathrm{TCI}}(u_1, \ldots, u_n) = \prod_{\alpha=1}^{\beta-1} \vec{B}_\alpha(u_\alpha) T_\beta(u_\beta) \prod_{\alpha=\beta+1}^{n} \overleftarrow{B}_\alpha(u_\alpha). \tag{C12}$$

In this form, the interpolation property (P1) is the direct analog of the norm computation of a canonical MPS [52]: When Eq. (C12) is evaluated on the elements of the T tensor, the product of B matrices telescopically reduces to identity.

APPENDIX D: FAST SUMMATION OVER KELDYSH INDICES

The calculation of the integrand $\tilde{Q}_n(u_i)$ amounts to summing up 2^n determinants of size $(2n+1) \times (2n+1)$ [25]. These determinants factorize into products of a $n \times n$ with a $(n+1) \times (n+1)$ determinant in the case considered in this article. A naive calculation of a determinant requires a computing time $\propto n^3$ so that the overall computational price of one call to the integrand is $2^n n^3$. Reference [42] proposes an algorithm to calculate all the principal minors of a $n \times n$ matrix M (the determinants of all the submatrices of M) at a much smaller cost of 2^n. This algorithm was later adapted to speed up both imaginary and real-time diagrammatic quantum Monte Carlo calculations [43]. Here, we propose an algorithm that is equivalent to the one developed in Ref. [43] yet does not require the use of nilpotent polynomials and as a result is perhaps more transparent. We also discuss the techniques used to avoid numerical instabilities or loss of precision.

1. Algorithm

The problem can be formulated as follows. Let $g(a, \alpha; a', \alpha')$ be a (Green) function that depends on the Keldysh indices $a, a' \in \{0, 1\}$ and on all the other degrees of freedom (time, space, and possibly spin, orbitals, etc.) labeled collectively as the α, α' variables. Let a_i (α_i) be a list of n values of the Keldysh (other variables). Calculating the integrand $\tilde{Q}(\alpha_1 \ldots \alpha_n)$ amounts to performing sums of the form

$$\tilde{Q} = \sum_{a_1 \ldots a_n} (-1)^{\sum_p a_p} \det M\{a_p\}, \tag{D1}$$

where the matrix $M\{a_i\}$ is defined as

$$M\{a_1 a_2 \ldots a_n\}_{ij} = g(a_i, \alpha_i; a_j, \alpha_j). \tag{D2}$$

Note that Eq. (D2) defines only the first n rows and columns. The matrix can be completed by adding more columns and rows of arbitrary value depending on which observables is computed. The first step of the algorithm is to introduce a matrix $\mathcal{M}$ that contains all the matrices $M\{a_p\}$ defined in Eq. (D2) as submatrices. $\mathcal{M}$ is obtained by stacking the two values of the Keldysh indices one after the other. More precisely, using the "C" convention where matrix indices start from zero, we write $p = 2i + a$ and $p' = 2i' + a'$ with $a, a' \in \{0, 1\}$ and $i, i' \in \{0, 1, 2, \ldots, n-1\}$ and define

$$\mathcal{M}_{pp'} = g(a, \alpha_i; a', \alpha_{i'}). \tag{D3}$$

The first column and row of $\mathcal{M}$ contains the elements of g corresponding to $a_1 = 0$, the second column and row corresponds to $a_1 = 1$, the third to $a_2 = 0$, and so on.

The principal minor algorithm uses the Schur complement (see Appendix A) to iteratively "remove" Keldysh indices. One starts with a_1. For a given value of $a_1 \in \{0, 1\}$, one first removes the row and column corresponding to the other value $1 - a_1$. Then one uses the Schur complement to "integrate out" the row and column associated with a_1 and define the matrix $\mathcal{M}^{a_1}$ with the following elements:

$$[\mathcal{M}^{a_1}]_{pp'} = [\mathcal{M}]_{pp'} - [\mathcal{M}]_{p,a_1} \frac{1}{z_1(a_1)} [\mathcal{M}]_{a_1,p'} \tag{D4}$$

with

$$z_1(a_1) = [\mathcal{M}]_{a_1,a_1} \tag{D5}$$

and $p, p' \in \{2, 3 \ldots 2n-1\}$. One can continue and define $\mathcal{M}^{a_1 a_2}$, $\mathcal{M}^{a_1 a_2 a_3}$, etc., by integrating out a_2 and then a_3, etc. We define $\mathcal{M}^{a_1 \ldots a_k}$ iteratively by

$$[\mathcal{M}^{a_1\ldots a_{k+1}}]_{pp'} = [\mathcal{M}^{a_1\ldots a_k}]_{pp'} - [\mathcal{M}^{a_1\ldots a_k}]_{p,2k+a_{k+1}} \times \frac{1}{z_{k+1}} [\mathcal{M}^{a_1\ldots a_k}]_{2k+a_{k+1},p'} \tag{D6}$$

with $p, p' \in \{2k+2, \ldots, 2n-1\}$ and

$$z_{k+1}(a_1, \ldots, a_{k+1}) = [\mathcal{M}^{a_1\ldots a_k}]_{2k+a_{k+1},2k+a_{k+1}}. \tag{D7}$$

The coefficients $z_i(a_i)$ are directly linked to our target determinant:

$$\det M\{a_1 a_2 \ldots a_n\} = \prod_{i=1}^{n} z_i(a_1, \ldots, a_i). \tag{D8}$$

The key remark to prove Eq. (D8) is a property of the Schur complement (A3): If one is interested in the determinant of a submatrix M of $\mathcal{M}$, one can equivalently either apply the Schur complement before or after deleting the corresponding rows or columns; i.e., the Schur complement commutes with row and matrix selection as long as the Schur pivot belongs to the submatrix. Noting $\mathcal{M}/a_0 a_1 \ldots a_p$ the submatrix of $\mathcal{M}$ where one has deleted the rows and columns corresponding to $\bar{a}_0 = 1 - a_0, \ldots, \bar{a}_p = 1 - a_p$, one has $M = \mathcal{M}/a_0 a_1 \ldots a_n$. Using the Schur complement theorem (A3), one can prove iteratively that

$$\det[\mathcal{M}/a_0 a_1 \ldots a_p] = \left(\prod_{i=1}^{p} z_i(a_1, \ldots, a_i)\right) \det \mathcal{M}^{a_1\ldots a_p} \tag{D9}$$

from which Eq. (D8) follows.

With these notations, the algorithm reads as follow. One initializes the algorithm with $a_1 a_2 \ldots a_n = 00\ldots0$ and construct the list of matrices $\mathcal{M}^0, \mathcal{M}^{00}, \ldots, \mathcal{M}^{00\ldots0}$ as well as the associated list of weights $z_1(0) z_2(0) \ldots z_n(0)$. Then, one iterates over the different values of $a_1 a_2 \ldots a_n$ sequentially with the inner loop on a_n. At each stage, we keep the list of matrices $(\mathcal{M}^{a_1}, \mathcal{M}^{a_1 a_2}, \ldots, \mathcal{M}^{a_1 a_2 \ldots a_n})$ and the weights $[z_1(a_1), z_2(a_2), \ldots, z_n(a_n)]$. Upon going from one set of Keldysh indices to the next, one uses Eqs. (D6) and (D7) to update the matrices and weights that have changed. The result of Eq. (D8) gives the contribution of the set $(a_1, a_2, \ldots, a_n)$ to the integrand. One can check that the overall computational cost is $\propto 2^n$.

The algorithm can also be extended straightforwardly to compute integrands of the form

$$\tilde{Q} = \sum_{a_1 \ldots a_n} (-1)^{\sum_p a_p} \det M\{a_p\} \det M'\{a_p\}, \tag{D10}$$

where $M\{a_p\}$ and $M'\{a_p\}$ are two matrices of the form defined by Eq. (D2), possibly with two different functions g and g'. One simply performs the algorithm simultaneously on the two matrices $\mathcal{M}$ and $\mathcal{M}'$. The product of the result of Eq. (D8) for the two matrices gives the contribution of the set $a_1 a_2 \ldots a_n$ to the integrand.

2. Technical implementation

The above algorithm can be implemented in a straightforward way. Below, we show a simple C++ implementation using the "armadillo" library [53]. The input of the function EvalSum is the matrix $\mathcal{M}$ in Eq. (D3). We find that the speedup of the simple implementation below against a direct sum of determinants is a factor 15 for $n = 12$. A more optimized (but less transparent) version can be obtained by preallocating the matrices or using an iterative implementation instead of a recursive one. In the implementation used in this article (using two matrices as input $\mathcal{M}$ and $\mathcal{M}'$), we observe a typical speed up of a factor 40 compared to the direct sum for $n = 12$.

```
cx_mat SchurComplement(cx_mat const& M, bool a)
{
    int s=M.n_rows;
    cx_mat Mc(s-2, s-2);
    for(int j=2;j<s;j++)
    {
      auto f=M(a,j)/M(a,a);
      for(int i=2;i<s;i++)
        Mc(i-2,j-2)=M(i,j)-M(i,a)*f;
    }
    return Mc;
}
cx_double EvalSum(cx_mat const& M,
                  cx_double r=1.0,bool sg=0)
{
    if (M.n_rows<2) return sg ? -r*det(M)
                              : r*det(M);
    cx_double sum=0;
    for(int a=0;a<2;a++)
    {
      cx_mat Mc=SchurComplement(M,a);
      sum+=EvalSum(Mc,r*M(a,a), sg!=a);
    }
    return sum;
}
```

Lastly, we mention two practical issues.

First, one call to the integrand is a summation over 2^n terms $\sum_{a_i} f(a_i)$, and there is a possibility of large cancellation between these terms resulting in a loss of precision. To detect this problem, we compute both $\sum_{a_i} f(a_i)$ and the sum of absolute values $\sum_{a_i} |f(a_i)|$. When these two quantities differ by many orders of magnitude, we recompute $\sum_{a_i} f(a_i)$ using the higher-precision "long double" mode.

Second, the above algorithm is not applicable if the diagonal element $\mathcal{M}_{00}$ vanishes, as the corresponding 1×1 Schur complement is ill defined (or ill conditioned if $\mathcal{M}_{00}$ is nonzero but very small). To address this issue for $\mathcal{M}_{00} \ll \|\mathcal{M}\|$, we switch to a 2×2 Schur complement

and use "partial pivoting" to maximize the determinant of the 2×2 matrix on which we perform the Schur complement (i.e., we reorder the matrix to maximize the magnitude of the incoming 2×2 determinant). The practical implementation of this 2×2 variant is only a factor 2 slower than the 1×1 version.

APPENDIX E: NONINTERACTING GREEN'S FUNCTIONS IN THE FLAT-BAND LIMIT

In this appendix, we discuss how to obtain the noninteracting Green's functions that form the input of the TTD algorithm. These Green's functions can be calculated for arbitrary tight-binding models using approaches developed, e.g., in the Tkwant package [41]. For systems weakly coupled to an environment, such as the quantum dots or double quantum dots studied in this article, an excellent approximation of this Green's function is given by the flat-band limit. This is the limit considered in this article. It is very suitable for benchmarks as (i) it corresponds to the limit for which we have the Bethe ansatz analytical solution at $\epsilon_d = 0$ and (ii) the Green's function can be written in terms of the exponential integral special function for which there exists machine precision implementations.

We partition our system into the "system" S (a set of quantum dots) and an "environment" E (typically the infinite leads). To compute the correlators of a given noninteracting Hamiltonian H_0, we need the retarded Green's function

$$g^R(\omega) = (\omega - H_0)^{-1}. \tag{E1}$$

The one-particle Hamiltonian H_0 has a 2×2 block structure:

$$H_0 = \begin{pmatrix} H_{SS} & H_{SE} \\ H_{ES} & H_{EE} \end{pmatrix}. \tag{E2}$$

Since we are interested only in the correlator g^R_{SS} in the SS subblock, we can write (using the inverse by block of a matrix; see Ref. [51])

$$g^R_{SS}(\omega) = [\omega - H_{SS} - \Delta(\omega)]^{-1}, \tag{E3}$$

where the *hybridization function*

$$\Delta(\omega) = \lim_{\eta \to 0^+} H_{SE} \frac{1}{\omega - H_{EE} + i\eta} H_{ES} \tag{E4}$$

contains all the effect of the bath E.

In many practical situations, the coupling of the system to the bath is sufficiently weak that the hybridization matrix $\Delta(\omega)$ can be considered as constant in the energy range of interest for the system. Neglecting the frequency dependence of the hybridization, we arrive at, i.e., $\Delta(\omega) \approx \Gamma_1 - i\Gamma_2 =$ constant matrix which is known as the *flat-band limit*. In this limit, the local Green's function above is given by

$$g^R_{SS}(\omega) = U(\omega - D)^{-1} U^{-1}, \tag{E5}$$

where U (respectively, D) are the eigenvectors (respectively, eigenvalues) of the effective Hamiltonian

$$H_{\text{eff}} = H_{SS} + \Gamma_1 - i\Gamma_2 = U \cdot D \cdot U^{-1}. \tag{E6}$$

Note that H_{eff} is not Hermitian and the eigenvalues D are complex, in general.

Once the retarded Green's function is known in the energy domain, we can obtain the lesser and greater Green's functions in real time. At thermal equilibrium and zero temperature, the lesser and greater Green's functions are given by,

$$g^{\lessgtr}_{SS}(t) = \frac{\mp i}{\pi} \int d\omega \exp(-i\omega t)\theta(\mp\omega)\text{Im}g^R_{SS}, \tag{E7}$$

where Im stands for the imaginary part.

Since in Eq. (E5) the eigenvectors U do not depend on ω and D is a diagonal matrix, these integrals can be computed explicitly:

$$\begin{aligned} g^{\lessgtr}_{SS}(t) &= \frac{1}{2\pi}[UI^{\lessgtr}(D,t)U^{-1} - U^* I^{\lessgtr}(D^*,t)(U^{-1})^*], \\ g^{<}_{SS}(0) &= \frac{-i}{\pi}\text{Im}\{U[\log(-D) + i\pi\text{sg}(\text{Im}D)]U^{-1}\}, \\ g^{>}_{SS}(0) &= -\frac{i}{\pi}\text{Im}\{U\log(-D)U^{-1}\} \end{aligned} \tag{E8}$$

with

$$\begin{aligned} I^{\lessgtr}(a,t) = \exp(-iat)\{&E_1(-iat) \\ &\pm 2\pi i\text{sg}(t)\theta[-\text{Im}(at)]\theta[\mp\text{Re}(a)]\}, \end{aligned} \tag{E9}$$

where $E_1(z) = \int_z^\infty dt \exp(-t)/t$ is the exponential-integral function E_1 (see Ref. [54]), $\text{sg}(x)$ is the sign function, and $\theta(x)$ is the Heaviside step function with $\theta(0) = 1/2$.

1. Single quantum dot (SIAM)

For a single quantum dot, the effective Hamiltonian matrix becomes a scalar $H_{\text{eff}} = \epsilon_d - i\Gamma$ yielding to

$$\begin{aligned} g^{\lessgtr}_{SS}(t) &= \frac{1}{2\pi}[I^{\lessgtr}(\epsilon_d - i\Gamma, t) - I^{\lessgtr}(\epsilon_d + i\Gamma, t)], \\ g^{<}_{SS}(0) &= \frac{-i}{\pi}\text{Im}[\log(-\epsilon_d + i\Gamma) + i\pi\text{sg}(-\Gamma)], \\ g^{>}_{SS}(0) &= -\frac{i}{\pi}\text{Im}\log(-\epsilon_d + i\Gamma). \end{aligned} \tag{E10}$$

This expressions form the inputs for our SIAM benchmark. All the energies (times) are measured in units of Γ $(1/\Gamma)$.

2. Double quantum dot

We also consider a double quantum dot with local Hamiltonian matrix

$$H_{SS} = \begin{pmatrix} 0 & \gamma_0 \\ \gamma_0 & 0 \end{pmatrix}$$

and hybridization function

$$\Delta(\omega) \approx -i \begin{pmatrix} \Gamma & 0 \\ 0 & \Gamma \end{pmatrix},$$

leading to

$$H_{\mathrm{eff}} = \begin{pmatrix} -i\Gamma & \gamma_0 \\ \gamma_0 & -i\Gamma \end{pmatrix} = U \cdot D \cdot U^{-1}, \tag{E11}$$

$$U = \begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix}, \quad D = \begin{pmatrix} -i\Gamma - \gamma_0 & 0 \\ 0 & -i\Gamma + \gamma_0 \end{pmatrix}. \tag{E12}$$

In our example, we use $\gamma_0 = 0.5\Gamma$ and apply Eq. (E8) to compute the noninteracting Green's functions. All the energies (times) are measured in units of Γ $(1/\Gamma)$.

APPENDIX F: NONINTERACTING GREEN'S FUNCTIONS IN 2D LATTICE

Here, we calculate the noninteracting Green's function for a particle in an infinite two-dimensional lattice used in Sec. VII A. The noninteracting Hamiltonian reads (omitting the spin index, since the problem is diagonal in spin)

$$H_0 = \sum_{\langle ij \rangle} c_i^\dagger c_j \tag{F1}$$

with sum over nearest neighbors.

1. Explicit summation in momentum space

The dispersion relation of H_0 is $E_{\boldsymbol{k}} = 2\cos k_x + 2\cos k_y$. Since the corresponding velocities $\vec{v} = \partial E/\partial \boldsymbol{k}$ are bounded by 2 in both spatial directions, it follows that it is enough to consider a finite lattice of length $L > 2t$ to calculate the Green's function without finite size effects. Hence, we consider a system of $L \times L$ sites with periodic boundary conditions. It can be diagonalized using the operators $\{d_{\boldsymbol{k}}\}$ in the momentum basis

$$c_i = \frac{1}{L} \sum_{\boldsymbol{k}} e^{i\boldsymbol{k}\cdot\boldsymbol{r}_i} d_{\boldsymbol{k}}, \tag{F2}$$

where $\boldsymbol{r}_i$ is the lattice position of site i and $\boldsymbol{k} = (k_x, k_y)$ with $k_{x,y} = (2\pi/L)\kappa_{x,y}$, and $\kappa_{x,y} = 0, 1, \ldots, L-1$. In Heisenberg representation, we simply have $d_{\boldsymbol{k}}(t) = e^{-iE_{\boldsymbol{k}}t} d_{\boldsymbol{k}}$. It follows that the lesser and greater Green's functions in real time at i, j are given by

$$g_{ij}^{<}(t) = \frac{i}{L^2} \sum_{\boldsymbol{k}} e^{i(\boldsymbol{k}\cdot\boldsymbol{r}_{ij} - tE_{\boldsymbol{k}})} f_{\mathrm{FD}}(E_{\boldsymbol{k}}), \tag{F3}$$

$$g_{ij}^{>}(t) = \frac{-i}{L^2} \sum_{\boldsymbol{k}} e^{i(\boldsymbol{k}\cdot\boldsymbol{r}_{ij} - tE_{\boldsymbol{k}})} \bar{f}_{\mathrm{FD}}(E_{\boldsymbol{k}}), \tag{F4}$$

where $\boldsymbol{r}_{ij} = \boldsymbol{r}_i - \boldsymbol{r}_j = (x, y)$ is the position difference with $x, y \in \mathbb{Z}$, $f_{\mathrm{FD}}(E) = 1/[e^{\beta(E-\mu)} + 1]$ is the Fermi-Dirac distribution, $\bar{f}_{\mathrm{FD}}(E) = 1 - f_{\mathrm{FD}}(E)$, β is the inverse of the temperate, and μ is the chemical potential. In practice, we compute the sum above for $L = 500$ and $i = j = 0$ for the single impurity in a lattice problem.

2. Thermodynamic limit

For $L \to \infty$, the previous expression for $g_{ij}^{<}(t)$ can be written as

$$g_{xy}^{<}(t) = \frac{i}{(2\pi)^2} \int_{-\pi}^{\pi} dk_x \int_{-\pi}^{\pi} dk_y e^{i(\boldsymbol{k}\cdot\boldsymbol{r}_{ij} - tE_{\boldsymbol{k}})} f_{\mathrm{FD}}(E_{\boldsymbol{k}}). \tag{F5}$$

At zero temperature, the Fermi function can be expanded as a Fourier integral:

$$\lim_{\beta\to\infty} f_{\mathrm{FD}}(E) = \theta(-E) = \frac{i}{2\pi} \int \frac{dw}{w + i0^+} e^{iwE}, \tag{F6}$$

which allows one to decouple the two integrals on k_x and k_y in Eq. (F5). Using the definition of the Bessel's functions (for $n \in \mathbb{Z}$) [see Eq. (10.9.2) in Ref. [54]],

$$\int_{-\pi}^{\pi} dk e^{i(kn - x\cos k)} = (-i)^n 2\pi J_n(x), \tag{F7}$$

we arrive at

$$g_{xy}^{\lessgtr}(t) = \frac{\mp(-i)^{x+y}}{2\pi} \left[\fint \frac{dw}{w} J_x(2t \mp 2w) J_y(2t \mp 2w) + i\pi J_x(2t) J_y(2t) \right], \tag{F8}$$

where the last integral represents its Cauchy principal value.

APPENDIX G: CALCULATION OF THE INTEGRAL IN THE SIMPLEX DOMAIN

The calculation of the integral in the simplex domain S_u, $0 \le u_n \le u_{n-1} \ldots u_1 \le t$ is not as straightforward as the

hypercube integration and requires an iterative algorithm that we now explain. The multidimensional integral (42) over the simplex in u variables has the explicit form

$$Q_n(t) = \int_0^t \lambda(u_1)du_1 \ldots \int_0^{u_{n-2}} \lambda(u_{n-1})du_{n-1} \times \int_0^{u_{n-1}} \lambda(u_n)du_n \tilde{Q}_n(u_1, \ldots, u_n). \tag{G1}$$

Since the TTD approximation is performed in the v variables, our approximation reads

$$\tilde{Q}_n(u_1, u_2, \ldots, u_n) \approx T_1(t-u_1)P_1^{-1}T_2(u_1-u_2)P_2^{-1} \ldots P_{n-1}^{-1}T_n(u_{n-1}-u_n). \tag{G2}$$

The integrals over the u variables are performed one by one starting with u_n and ending with u_1. The u_n variable is present only in the last tensor T_n. We perform the corresponding one-dimensional integral. Defining

$$\Psi_n(x) \equiv \int_0^x dy\lambda(y)T_n(x-y), \tag{G3}$$

we find that

$$Q_n(t) \approx \int_0^t \lambda(u_1)du_1 \ldots \int_0^{u_{n-2}} \lambda(u_{n-1})du_{n-1}T_1(t-u_1)P_1^{-1} \times T_2(u_1-u_2)P_2^{-1} \ldots T_{n-1}(u_{n-2}-u_{n-1}) \times P_{n-1}^{-1}\Psi_n(u_{n-1}). \tag{G4}$$

We continue with the one-dimensional integral over u_{n-1} which is present only in the terms $T_{n-1}(u_{n-2}-u_{n-1}) \times \Psi_{n-1}(u_{n-1})$. Defining for $p < n$

$$\Psi_p(x) \equiv \int_0^x dy\lambda(y)T_p(x-y)P_p^{-1}\Psi_{p+1}(y), \tag{G5}$$

we find that

$$Q_n(t) \approx \int_0^t \lambda(u_1)du_1 \ldots \int_0^{u_{n-3}} \lambda(u_{n-2})du_{n-2}T_1(t-u_1)P_1^{-1} \times T_2(u_1-u_2)P_2^{-1} \ldots T_{n-2}(u_{n-3}-u_{n-2}) \times P_{p-2}^{-1}\Psi_{n-1}(u_{n-2}). \tag{G6}$$

We continue to perform the integrations one by one until we arrive at the final integration

$$Q_n(t) \approx \int_0^t \lambda(u_1)du_1T_1(t-u_1)P_1^{-1}\Psi_2(u_1) = \Psi_1(t). \tag{G7}$$

In practice, the above algorithm requires the precise knowledge of the $\Psi_p(u_{p-1})$ functions. We use precise Chebyshev interpolants of the $T_n(v_n)$ matrices and $\Psi_p(u_{p-1})$ vectors to define the right-hand side of Eqs. (G3) and (G5) in terms of large-order polynomials whose primitive is known exactly. The result is projected again on Chebyshev polynomials. It is important to note that the last integral (G7) provides the entire t dependence of $Q_n(t)$ and that it is a posttreatment calculation that can be performed for any time-dependent switching on of the interaction $\lambda(t)$.

APPENDIX H: CALCULATION OF THE SIMPLEX INTEGRAL USING FOURIER TRANSFORM

As an alternative to the integration in u variables in the simplex domain discussed in Appendix G, the multidimensional integral can be as well calculated in the variables v [defined in Eq. (49)], together with the domain condition (50a) and (50b). Note that this alternative route is defined only for the abrupt switching of the interaction $\lambda(t) = \theta(t)$ and cannot be generalized to arbitrary functions $\lambda(t)$. In the v variables, the integral in Eq. (42) is essentially a multidimensional convolution:

$$Q_n(t) = \prod_{\alpha=1}^n \int_0^t dv_\alpha \theta\left[t - \sum_{i=1}^n v_i\right] \tilde{Q}_n(v_1, \ldots, v_n), \tag{H1}$$

where $\theta(x)$ is the Heaviside step function. The Fourier representation of the Heaviside function,

$$\theta(t) = \lim_{\epsilon \to 0^+} \int_{-\infty}^{\infty} \frac{d\omega}{2\pi i} \frac{e^{i\omega t}}{\omega - i\epsilon}, \tag{H2}$$

can be used to remove the constraints on the v variables. Using the tensor train factorization approximation in Eq. (51) for $\tilde{Q}_n(v_1, \ldots, v_n)$, one can write Eq. (H1) as

$$Q_n(t) \approx \int_{-\infty}^{\infty} \frac{d\omega}{2\pi i} \frac{e^{i\omega t}}{\omega - i0^+} \prod_{\alpha=1}^n \int_0^t dv_\alpha e^{-i\omega v_\alpha} T_\alpha(v_\alpha)P_\alpha^{-1}. \tag{H3}$$

With $(x \pm i0^+)^{-1} = \mp i\pi\delta(x) + \text{p.v.}(1/x)$, where p.v. stands for the principal value, one arrives at

$$Q_n(t) = \frac{\tilde{q}_n(0)}{2} + \fint_{-\infty}^{\infty} \frac{d\omega}{2\pi i\omega} e^{i\omega t}\tilde{q}_n(\omega), \tag{H4a}$$

$$\tilde{q}_n(\omega) = \prod_{\alpha=1}^n \int_0^t dv_\alpha e^{-i\omega v_\alpha} T_\alpha(v_\alpha)P_\alpha^{-1}. \tag{H4b}$$

Instead of the initial n-dimensional integral, the above equation for $\tilde{q}_n(\omega)$ is a product of n one-dimensional integrals, which can be computed numerically. Moreover, the function $\tilde{q}_n(\omega)$ can be precomputed once and the entire $Q_n(t)$ curve obtained *a posteriori* by evaluating the remaining one-dimensional integral in Eq. (H4a) for

different values of the time t. In practice, as the integrands in Eqs. (H4a) and (H4b) decrease fast, the integrals are cut off at finite, large enough values of ω and v. Appropriate quadratures [55] are used to compute the principal value integral numerically around the pole at $\omega = 0$, as well as for the oscillatory integrals in Eqs. (H4a) and (H4b). For the precomputation, $\tilde{q}_n(\omega)$ is interpolated using piecewise adaptive polynomials as in Ref. [56].

APPENDIX I: EFFICIENT QUADRATURE FOR $T_\alpha(v)$

In the case of the single impurity embedded in a two-dimensional lattice, the tensors $T_\alpha(v)$ oscillate rapidly and decay slowly with respect to v. We present a specialized quadrature scheme to compute them efficiently.

Since the integrand $\tilde{Q}_n(u_i)$ is a sum of products of noninteracting Green's functions, we are able to characterize the behavior of $T_\alpha(v)$ as $v \to \infty$. For the single impurity embedded in a two-dimensional lattice, we find empirically that we can accurately approximate $T_\alpha(v)$ by an expansion of the type

$$T_\alpha(v) \approx \sum_{p=0}^{N_p} \sum_{n=2}^{N_n} a_{pn} \frac{\cos(4pv)}{v^n} + b_{pn} \frac{\sin(4pv)}{v^n} \quad \text{(I1)}$$

when $v > v_{\text{cut}}$, for v_{cut} a sufficiently large cutoff. The chosen frequencies originate from the bandwidth of the noninteracting Hamiltonian, and the algebraic decay is observed empirically. N_p and N_n are used to control the precision of the expansion, and in practice we observe rapid convergence in these parameters. We therefore split the integral into two parts:

$$\int_0^\infty T_\alpha(v)dv = \int_0^{v_{\text{cut}}} T_\alpha(v)dv + \int_{v_{\text{cut}}}^\infty T_\alpha(v)dv. \quad \text{(I2)}$$

$T_\alpha(v)$ typically contains only a few oscillations on $[0, v_{\text{cut}}]$, so the first integral can be computed efficiently using a standard Gauss-Legendre quadrature rule. For the second integral, we use Eq. (I1) to design a custom quadrature rule, as follows.

We describe the method for a generic collection of functions $f:(a,b) \to \mathbb{C}$ defined as the span of N basis functions ϕ_k:

$$f(x) = \sum_{k=1}^{N} \hat{f}_k \phi_k(x). \quad \text{(I3)}$$

The functions T_α form such a class approximately, with $(a,b) = (v_{\text{cut}}, \infty)$ and the basis functions ϕ_k given by Eq. (I1). Given a collection x_j of N sampling points for the functions ϕ_k, we define the matrix $\Phi_{jk} \equiv \phi_k(x_j)$ and have $f_j \equiv f(x_j) = \sum_{k=1}^N \Phi_{jk}\hat{f}_k$. If $I_k \equiv \int_a^b \phi_k(x)dx$, then

$$\int_a^b f(x)dx = \sum_{k=1}^{N} I_k \hat{f}_k = \sum_{j,k=1}^{N} I_k \Phi_{kj}^{-1} f_j \equiv \sum_{j=1}^{N} w_j f_j, \quad \text{(I4)}$$

where we define the quadrature weights w_j.

The nodes x_j must be chosen properly to ensure stability. To do so, we form the $M \times N$ matrix $\phi_k(\bar{x}_j)$, where $\{\bar{x}_j\}_{j=1}^M$ is a fine grid on (a,b), sufficient to accurately discretize all of the functions ϕ_k. It can be shown that the nodes x_j corresponding to the pivot indices obtained by pivoted Gram-Schmidt orthogonalization on the rows of this matrix yield a stable quadrature rule [57]. Roughly speaking, this procedure chooses the N most linearly independent rows of the matrix, yielding the N most independent nodes in the fine grid. Alternatively, we find in practice that the nodes corresponding to the pivots of the cross interpolation of $\phi_k(\bar{x}_j)$ may be used as well.

We can follow this procedure to compute a quadrature rule for the functions T_α using the expansion (I1). In this case, it is straightforward to write the integrals I_k in terms of the well-known E_n functions:

$$E_n(z) \equiv \int_1^\infty \frac{e^{-zt}}{t^n} dt. \quad \text{(I5)}$$

E_1 is the exponential integral, which can be evaluated using standard libraries [58], and $E_n(z)$ can then be obtained by a simple recurrence [see Eq. (8.19.12) in Ref. [54]]. In practice, we set $v_{\text{cut}} = 10$, use 63 Gauss-Legendre nodes for the integral on $[0, v_{\text{cut}}]$, and set $N_p = 2$, $N_n = 4$ to obtain 15 nodes for the integral on $[v_{\text{cut}}, \infty)$. These 78 quadrature nodes yield 2–3 digits of accuracy in the final result.

[1] R. Blankenbecler, D. J. Scalapino, and R. L. Sugar, *Monte Carlo Calculations of Coupled Boson-Fermion Systems. I*, Phys. Rev. D **24**, 2278 (1981).

[2] W. M. C. Foulkes, L. Mitas, R. J. Needs, and G. Rajagopal, *Quantum Monte Carlo Simulations of Solids*, Rev. Mod. Phys. **73**, 33 (2001).

[3] A. W. Sandvik, *Computational Studies of Quantum Spin Systems*, AIP Conf. Proc. **1297**, 135 (2010).

[4] K. Van Houcke, E. Kozik, N. Prokof'ev, and B. Svistunov, *Diagrammatic Monte Carlo*, Phys. Procedia **6**, 95 (2010).

[5] J. Carlson, S. Gandolfi, F. Pederiva, S. C. Pieper, R. Schiavilla, K. E. Schmidt, and R. B. Wiringa, *Quantum Monte Carlo Methods for Nuclear Physics*, Rev. Mod. Phys. **87**, 1067 (2015).

[6] I. V. Oseledets, *Tensor-Train Decomposition*, SIAM J. Sci. Comput. **33**, 2295 (2011).

[7] S. Dolgov and D. Savostyanov, *Parallel Cross Interpolation for High-Precision Calculation of High-Dimensional Integrals*, Comput. Phys. Commun. **246**, 106869 (2020).

[8] S. Dolgov, K. Anaya-Izquierdo, C. Fox, and R. Scheichl, *Approximation and Sampling of Multivariate Probability*

Distributions in the Tensor Train Decomposition, Stat. Comput. **30**, 603 (2020).

[9] L. I. Vysotsky, A. V. Smirnov, and E. E. Tyrtyshnikov, *Tensor-Train Numerical Integration of Multivariate Functions with Singularities*, Lobachevskii J. Math. **42**, 1608 (2021).

[10] A. Chertkov and I. Oseledets, *Solution of the Fokker-Planck Equation by Cross Approximation Method in the Tensor Train Format*, Front. Artif. Intell. **4**, 668215 (2021).

[11] A. Smirnov, N. Shapurov, and L. Vysotsky, *Fiesta5: Numerical High-Performance Feynman Integral Evaluation*, Comput. Phys. Commun. **277**, 108386 (2022).

[12] I. Oseledets and E. Tyrtyshnikov, *TT-Cross Approximation for Multidimensional Arrays*, Linear Algebra Appl. **432**, 70 (2010).

[13] D. Savostyanov and I. Oseledets, *Fast Adaptive Interpolation of Multi-dimensional Arrays in Tensor Train Format*, in *Proceedings of the 2011 International Workshop on Multidimensional (nD) Systems* (Institute of Electrical and Electronics Engineers (IEEE), New York, NY, 2011), pp. 1–8.

[14] D. V. Savostyanov, *Quasioptimality of Maximum-Volume Cross Interpolation of Tensors*, Linear Algebra Appl. **458**, 217 (2014).

[15] S. A. Goreinov, N. L. Zamarashkin, and E. E. Tyrtyshnikov, *Pseudo-Skeleton Approximations by Matrices of Maximal Volume*, Mathematical notes of the Academy of Sciences of the USSR **62**, 515 (1997).

[16] M. Bebendorf, *Approximation of Boundary Element Matrices*, Num. Math. **86**, 565 (2000).

[17] S. A. Goreinov, I. V. Oseledets, D. V. Savostyanov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, *How to Find a Good Submatrix*, in *Matrix Methods: Theory, Algorithms and Applications* (World Scientific, Singapore, 2010), pp. 247–256.

[18] N. Kishore Kumar and J. Schneider, *Literature Survey on Low Rank Approximation of Matrices*, Linear Multilinear Alg. **65**, 2212 (2017).

[19] S. R. White, *Density Matrix Formulation for Quantum Renormalization Groups*, Phys. Rev. Lett. **69**, 2863 (1992).

[20] W. Huggins, P. Patil, B. Mitchell, K. B. Whaley, and E. M. Stoudenmire, *Towards Quantum Machine Learning with Tensor Networks*, Quantum Sci. Technol. **4**, 024001 (2019).

[21] N. V. Prokof'ev and B. V. Svistunov, *Polaron Problem by Diagrammatic Quantum Monte Carlo*, Phys. Rev. Lett. **81**, 2514 (1998).

[22] N. V. Prokof'ev and B. V. Svistunov, *Bold Diagrammatic Monte Carlo: A Generic Sign-Problem Tolerant Technique for Polaron Models and Possibly Interacting Many-Body Problems*, Phys. Rev. B **77**, 125101 (2008).

[23] A. S. Mishchenko, N. V. Prokof'ev, B. V. Svistunov, and A. Sakamoto, *Comprehensive Study of Fröhlich Polaron*, Int. J. Mod. Phys. B **15**, 3940 (2001).

[24] K. Van Houcke, F. Werner, E. Kozik, N. Prokof'ev, B. Svistunov, M. J. H. Ku, A. T. Sommer, L. W. Cheuk, A. Schirotzek, and M. W. Zwierlein, *Feynman Diagrams versus Fermi-Gas Feynman Emulator*, Nat. Phys. **8**, 366 (2012).

[25] R. E. V. Profumo, C. Groth, L. Messio, O. Parcollet, and X. Waintal, *Quantum Monte Carlo for Correlated Out-of-Equilibrium Nanoelectronic Devices*, Phys. Rev. B **91**, 245154 (2015).

[26] W. Wu, M. Ferrero, A. Georges, and E. Kozik, *Controlling Feynman Diagrammatic Expansions: Physical Nature of the Pseudogap in the Two-Dimensional Hubbard Model*, Phys. Rev. B **96**, 041105(R) (2017).

[27] R. Rossi, *Determinant Diagrammatic Monte Carlo Algorithm in the Thermodynamic Limit*, Phys. Rev. Lett. **119**, 045701 (2017).

[28] K. Chen and K. Haule, *A Combined Variational and Diagrammatic Quantum Monte Carlo Approach to the Many-Electron Problem*, Nat. Commun. **10**, 3725 (2019).

[29] C. Bertrand, S. Florens, O. Parcollet, and X. Waintal, *Reconstructing Nonequilibrium Regimes of Quantum Many-Body Systems from the Analytical Structure of Perturbative Expansions*, Phys. Rev. X **9**, 041008 (2019).

[30] C. Bertrand, O. Parcollet, A. Maillard, and X. Waintal, *Quantum Monte Carlo Algorithm for Out-of-Equilibrium Green's Functions at Long Times*, Phys. Rev. B **100**, 125129 (2019).

[31] A. Moutenet, P. Seth, M. Ferrero, and O. Parcollet, *Cancellation of Vacuum Diagrams and the Long-Time Limit in Out-of-Equilibrium Diagrammatic Quantum Monte Carlo*, Phys. Rev. B **100**, 085125 (2019).

[32] R. Rossi, F. Simkovic, and M. Ferrero, *Renormalized Perturbation Theory at Large Expansion Orders*, Europhys. Lett. **132**, 11001 (2020).

[33] M. Maček, P. T. Dumitrescu, C. Bertrand, B. Triggs, O. Parcollet, and X. Waintal, *Quantum Quasi-Monte Carlo Technique for Many-Body Perturbative Expansions*, Phys. Rev. Lett. **125**, 047702 (2020).

[34] K. Haule and K. Chen, *Single-Particle Excitations in the Uniform Electron Gas by Diagrammatic Monte Carlo*, Sci. Rep. **12**, 2294 (2022).

[35] C. Bertrand, D. Bauernfeind, P. T. Dumitrescu, M. Macek, X. Waintal, and O. Parcollet, *Quantum Quasi Monte Carlo Algorithm for Out-of-Equilibrium Green Functions at Long Times*, Phys. Rev. B **103**, 155104 (2021).

[36] F. Simkovic, R. Rossi, and M. Ferrero, *The Weak, the Strong and the Long Correlation Regimes of the Two-Dimensional Hubbard Model at Finite Temperature*, arXiv:2110.05863.

[37] J. Schneider, *Error Estimates for Two-Dimensional Cross Approximation*, J. Approx. Theory **162**, 1685 (2010).

[38] S. A. Goreinov and E. E. Tyrtyshnikov, *Quasioptimality of Skeleton Approximation of a Matrix in the Chebyshev Norm*, in *Doklady Mathematics* (Springer, New York, 2011), Vol. 83, pp. 374–375.

[39] A. Kronrod, *Nodes and Weights of Quadrature Formulas: Sixteen-Place Tables* (Consultants Bureau, New York, 1965).

[40] A. N. Rubtsov and A. I. Lichtenstein, *Continuous-Time Quantum Monte Carlo Method for Fermions: Beyond Auxiliary Field Framework*, J. Exp. Theor. Phys. Lett. **80**, 61 (2004).

[41] T. Kloss, J. Weston, B. Gaury, B. Rossignol, C. Groth, and X. Waintal, *Tkwant: A Software Package for Time-Dependent Quantum Transport*, New J. Phys. **23**, 023025 (2021).

[42] K. Griffin and M. J. Tsatsomeros, *Principal Minors, Part I: A Method for Computing All the Principal Minors of a Matrix*, Linear Algebra Appl. **419**, 107 (2006).

[43] F. Simkovic and M. Ferrero, *Fast Principal Minor Algorithms for Diagrammatic Monte Carlo*, Phys. Rev. B **105**, 125104 (2022).

[44] P. B. Wiegmann and A. M. Tsvelick, *Exact Solution of the Anderson Model: I*, J. Phys. C **16**, 2281 (1983).

[45] G. Cohen, E. Gull, D. R. Reichman, A. J. Millis, and E. Rabani, *Numerically Exact Long-Time Magnetization Dynamics at the Nonequilibrium Kondo Crossover of the Anderson Impurity Model*, Phys. Rev. B **87**, 195108 (2013).

[46] G. Cohen, D. R. Reichman, A. J. Millis, and E. Gull, *Green's Functions from Real-Time Bold-Line Monte Carlo*, Phys. Rev. B **89**, 115139 (2014).

[47] G. Cohen, E. Gull, D. R. Reichman, and A. J. Millis, *Green's Functions from Real-Time Bold-Line Monte Carlo Calculations: Spectral Properties of the Nonequilibrium Anderson Impurity Model*, Phys. Rev. Lett. **112**, 146802 (2014).

[48] G. Cohen, E. Gull, D. R. Reichman, and A. J. Millis, *Taming the Dynamical Sign Problem in Real-Time Evolution of Quantum Many-Body Problems*, Phys. Rev. Lett. **115**, 266802 (2015).

[49] E. Eidelstein, E. Gull, and G. Cohen, *Multiorbital Quantum Impurity Solver for General Interactions and Hybridizations*, Phys. Rev. Lett. **124**, 206405 (2020).

[50] J. Li, Y. Yu, E. Gull, and G. Cohen, *Interaction-Expansion Inchworm Monte Carlo Solver for Lattice and Impurity Models*, Phys. Rev. B **105**, 165133 (2022).

[51] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. (Johns Hopkins University Press, Baltimore, 1996).

[52] U. Schollwöck, *The Density-Matrix Renormalization Group in the Age of Matrix Product States*, Ann. Phys. (Amsterdam) **326**, 96 (2011).

[53] C. Sanderson and R. Curtin, *Armadillo: A Template-Based C++ Library for Linear Algebra*, J. Open Source Softwaare **1**, 26 (2016).

[54] DLMF, NIST Digital Library of Mathematical Functions, edited by f. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller, B. V. Saunders, H. S. Cohl, and M. A. McClain, http://dlmf.nist.gov/, release 1.1.5, 2022-03-15.

[55] R. Piessens, E. de Doncker-Kapenga, C. W. Überhuber, and D. K. Kahaner, *QUADPACK A Subroutine Package for Automatic Integration*, Springer Series in Comput. Math. (Springer, New York, 1983).

[56] P. Gonnet, *Increasing the Reliability of Adaptive Quadrature Using Explicit Interpolants*, ACM Trans. Math. Softw. **37**, 26 (2010).

[57] J. Bremer, Z. Gimbutas, and V. Rokhlin, *A Nonlinear Optimization Procedure for Generalized Gaussian Quadratures*, SIAM J. Sci. Comput. **32**, 1761 (2010).

[58] B. Gough, *GNU Scientific Library Reference Manual* (Network Theory Ltd., 2009), ISBN 978-0-9546120-7-8, https://dl.acm.org/doi/10.5555/1538674.