9 Learning Feynman Diagrams with Tensor Trains

Xavier Waintal
PHELIQS
Université Grenoble Alpes, CEA, Grenoble INP, IRIG
Grenoble 38000, France

Contents

E. Pavarini, E. Koch, A. Lichtenstein, and D. Vollhardt (eds.)
Understanding Correlated Materials with DMFT
Modeling and Simulation Vol. 15
Forschungszentrum Jülich, 2025, ISBN 978-3-95806-813-1
http://www.cond-mat.de/events/correl25

1 Introduction

Feynman diagrams form an exact formal solution of quantum field theories and quantum many-body problems in terms of a (huge) sum of (multi-dimensional) integrals of products of free propagators (which are known). For a long time these diagrams were mostly used and studied analytically. Countless articles are devoted to, e.g., analyze their structure or find subsequences of diagrams that could be summed exactly and form an appropriate approximate solution of the many-body problem in this or that regime. The idea to teach a computer how to calculate these diagrams is of course very appealing. A particular route in this direction is known as "diagrammatic Monte Carlo" and can be traced down to, e.g., [1, 2] for equilibrium and [3] in the out-of-equilibrium context. Diagrammatic Monte Carlo combines two ideas: First, since calculating a Feynman diagram amounts to calculating a multi-dimensional integral, and since Monte Carlo (e.g. the Metropolis algorithm) is the primary method for that, one aims at sampling these integrals through a Markov process. Second, since there are a very large number of Feynman diagrams (typically $O(n!)$ for a calculation at order n), it is not possible to calculate them all so one might as well use the Markov process for both integration *and* summing the diagrams. In other words, in diagrammatic Monte Carlo, a configuration is a diagram with its set of vertices (times and positions where an interaction event occurs) and the Markov process introduces a random walk between different diagrams and different sets of vertices. The original version of diagrammatic Monte Carlo had some successes but suffered from three difficulties:

- Problem A: The number of diagrams grows too fast with n, making it difficult to obtain converged results beyond $n = 6$–7.
- Problem B: The integrands may oscillate (depending on the regime) and become intractable through Monte Carlo. This is known as the "sign problem"
- Problem C: The expansion itself may not converge even if large values of n can be obtained. This depends on the nature of the series (asymptotic, finite radius of convergence...).

We will discuss these three aspects, in turn, in details. The present notes describe a set of works [4–10] my collaborators and I did to address these problems in the context of quantum nanoelectronics. There are no new results here, merely this is a high level view of what we did, designed to be much less formal and more accessible than the original research articles. Among the different authors, these notes owe much to my old partner Olivier Parcollet who has been my constant collaborator on this topic. Most of the actual work, as often, has been carried out by young researchers including (using time-ordering as is fit for a diagrammatic paper) Elio Profumo, Corentin Bertrand, Marjan Macek, Philipp Dumitrescu, Matthieu Jeannin, Thomas Kloss and Yuriel Nunez Fernandez.

The most urgent problem was Pb. A and it was also how we got started on the topic. In [4], we found a way to group the $n!$ diagrams into a much smaller sets of 2^n determinants of $n \times n$ matrices, dramatically reducing the complexity. This was done in the context of the Keldysh

formalism suitable to treat out-of-equilibrium problems. A similar reduction in the context of imaginary-time diagrams was found two years later in [11]. These developments made calculations possible up to $n \sim 15$ or so in the absence of a sign problem (Pb. B).

To calculate the resulting integrals (Pb. B) we took a somewhat circumvoluted path. We started with Monte Carlo sampling as was done in diagrammatic Monte Carlo [4,5]. It became quickly obvious that this was inefficient: the integrand was called typically 10^9 times in a practical calculation (each call having a cost of up to $O(2^{15})$ floating-point operations for the largest $n = 15$ that we could reach) but we did not take any advantage of the associated accumulated information. An initial idea was that an approximation of the integrand could be learned along the way (machine learning is the fashion). In turn, this approximation could be used to either speed-up the Markov process (decrease its correlation time through smarter proposed moves) or, as it turned out, obtain better convergence using low discrepancy sequences (better known as quasi-Monte Carlo) [7]. We shall not follow this line of thought here because it was eventually supplanted by a much better technique. Indeed, we found that the integrand could be learned directly using tensor network techniques and that the resulting tensor network could be integrated exactly without any need for Monte Carlo [8]. Hence, the technique in its current form does not use diagrams anymore and does not use Monte Carlo either. It is a sort of "Non-diagrammatic Non-Monte Carlo" technique.

For the last problem (Pb. C), we took two different routes. The first was to consider the resummation as an analytical continuation problem [4, 6] which was effective in some situations but difficult to turn into an automatic technique (i.e. that works without human supervision). Here, I will focus on our latest approach that uses "cross extrapolation" [9] and that is routed in the same mathematics as our tensor network learning technique.

In the remaining of these notes, I will walk the reader through what I just described, i.e., describing the problem, then our technique to try and solve problem A, then problem B and ultimately problem C. I will end by showing some actual data in the context of the out-of-equilibrium Anderson model (SIAM), the main model for which we have had results at the time of this writing [10]. Our main success there was to be able to compute the differential conductance versus bias and gate voltages "exactly" including its most prominent features, the "Kondo ridge" and the "Coulomb diamonds". By "exactly" here, we mean that the technique is controlled: it has error bars that (*i*) are known and (*ii*) can be systematically improved by increasing the computing time (in contrast to, e.g., mean field techniques).

2 Problem formulation

2.1 General model

The type of models we want to study correspond to a finite interacting quantum nanoelectronic system connected to infinite (metallic) electrodes, following the approach of Ref. [12]. The Hamiltonian consists of a quadratic term and an electron-electron interaction term,

$$\hat{\mathbf{H}}(t) = \hat{\mathbf{H}}_0(t) + U\hat{\mathbf{H}}_{\rm int}(t) \tag{1}$$

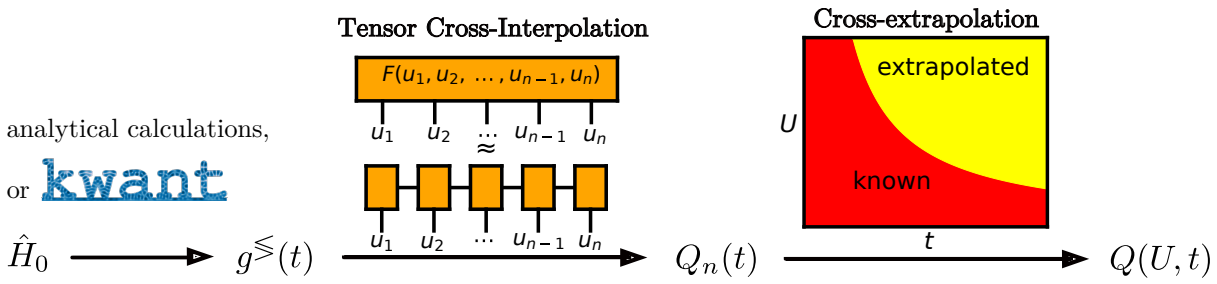


Fig. 1: *Schematic of the overall approach. Starting from non-interacting Green functions, a decomposition of the multidimensional integral is obtained using Tensor Cross-Interpolation to compute the perturbative expansion, from which the physical observables $Q(U,t)$ are reconstructed as a function of interaction U and time t using cross-extrapolation. (Adapted from [10])*

where the parameter U controls the magnitude of the interaction. The non-interacting Hamiltonian takes the form

$$\hat{\mathbf{H}}_0(t) = \sum_{i,j} H^0_{ij}(t)\, \hat{\mathbf{c}}^\dagger_i \hat{\mathbf{c}}_j \tag{2}$$

where $\hat{\mathbf{c}}^\dagger_i$ ($\hat{\mathbf{c}}_j$) are the usual fermionic creation (annihilation) operators of a one-particle state on the site i.

$$\{\hat{\mathbf{c}}^\dagger_i, \hat{\mathbf{c}}_j\} = \delta_{ij}\,. \tag{3}$$

The site index i is general and can include different kinds of degrees of freedom: space, spin, orbitals. A crucial aspect is that the number of "sites" is *infinite* so that the non-interacting system has a well-defined density of states (as opposed to a sum of delta functions for a finite system) while interactions only take place in a finite region. This will ensure the infrared convergence of the various terms of the perturbation expansion while using a discretized system provides ultraviolet convergence. So in contrast to quantum field theory where some diagrams may diverge, here each diagram taken separately will be finite. The interaction Hamiltonian takes the form

$$\hat{\mathbf{H}}_{\rm int}(t) = \sum_{ijkl} V_{ijkl}(t)\, \hat{\mathbf{c}}^\dagger_i \hat{\mathbf{c}}^\dagger_j \hat{\mathbf{c}}_k \hat{\mathbf{c}}_l \tag{4}$$

In contrast to the non-interacting part, it is confined to a *finite* region. We also suppos that the interaction vanishes for negative time and is slowly or abruptly switched on at $t = 0$.

2.2 Summary of the overall approach

Our goal is to perform a systematic expansion of some observable in power of U. For instance, if we are interested in the occupation $Q(U,t)$ of a given site i_0 at time t, we will write its expansion as,

$$Q(U,t) \equiv \left\langle \hat{\mathbf{c}}^\dagger_{i_0} \hat{\mathbf{c}}_{i_0} \right\rangle = \sum_{n=0}^{+\infty} Q_n(t)\, U^n. \tag{5}$$

Our goal will be to first compute the coefficients $Q_n(t)$ for as large a n as possible and then sum up the series. A schematic of the method is shown in Fig. 1. It consists of three main steps:

(*i*) solve the non-interacting model to obtain the corresponding non-interacting Green functions (such as the lesser $g^<_{ij}(t)$ or upper $g^>_{ij}(t)$ Green functions that will be introduced below). (*ii*) Calculating the coefficients $Q_n(t)$ with high precision and (*iii*) resumming this perturbative expansion.

The first step is carried out either analytically in simple models or numerically for more complex geometries using, e.g., the Kwant [13] or Tkwant [14] software. Note that the non-interacting Green function may already be out-of-equilibrium in presence of a bias voltage V_b.

The second step is to perform the actual expansion in powers of U. We shall see that this expansion gives an n-dimensional integral that contains 2^n terms constructed out of products of the non-interacting Green functions [4, 6]. We compute this integral using the Tensor Cross-Interpolation (TCI) algorithm which vastly outperforms previous quantum Monte Carlo and quasi Monte Carlo approaches for this problem [8]. A large fraction of these notes will be devoted to an introduction to TCI which has applications far beyond the present problem. See [15] for an in-depth description of TCI. An associated open source library may also be found at `https://tensor4all.org`. With this we will be able to calculate all Feynman diagrams up to $n = N$ with typically $N = 20$–25 for our example problem.

The last step of the method consist in reconstructing the function $Q(U, t)$, if possible for both short and long times t and interaction strength U from the knowledge of only N coefficients $Q_n(t)$. The analytical structure of $Q(U, t)$ is quite interesting: At any fine time t, it has an infinite radius of convergence [5] since

$$Q_n \sim \frac{t^n}{n!} \to Q_n U^n \sim \left(\frac{eUt}{n}\right)^n. \tag{6}$$

which implies that $N \sim O(Ut)$ terms are required to converge the sum using the finite sum

$$Q(U, t) \approx \sum_{n=0}^{N} Q_n(t)\, U^n. \tag{7}$$

On the other hand, in the steady state $t \to \infty$, it has a finite radius of convergence R and $Q_n \sim 1/R^n$ and the above naive summation fails for $U > R$. In other words, the two limits $t \to \infty$ and $N \to \infty$ do not commute. Here, we will focus on one approach to perform the reconstruction, the *cross-extrapolation* [9]. It is based on two ideas,

- First, using the naive sum we can obtain $Q(U, t)$ in two different regimes
  - Arbitrary times and small interactions $U < R$.
  - Large interactions but small times ($U \sim N/t$).

  To calculate $Q(U, t)$ at both large t *and* U, it is therefore tempting to perform a *double* extrapolation from these two limits simultaneously.
- Second, we will use the fact that $Q(U, t)$ *almost* factorizes, i.e., seen as a matrix where U are the rows and t the columns, it is of low rank.

As we shall see, cross-extrapolation is a general idea that could be used in many other situations.

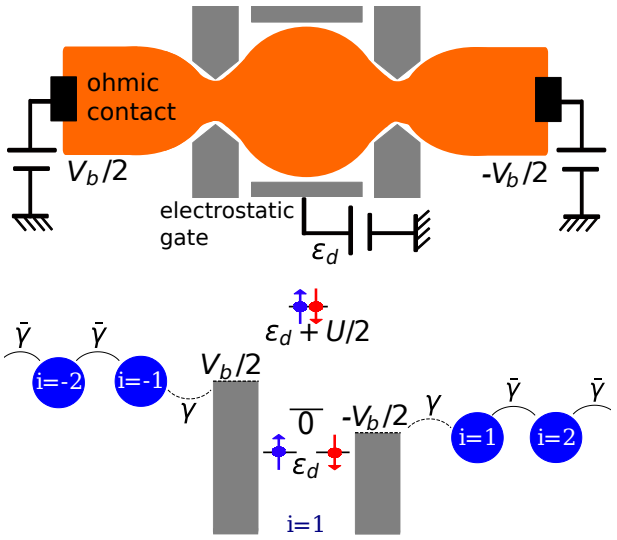


Fig. 2: *Top: electrical diagram of a quantum dot coupled to two electrodes. Bottom: schematic of the SIAM. (Adapted from [10])*

2.3 The SIAM model

Although the approach is in principle general, in practice we will demonstrate it for a concrete model, the out-of-equilibrium SIAM. The model corresponds to a quantum dot, typically constructed by confining electrons with electrostatic gates inside a semiconductor or a semiconductor heterostructure. The quantum dot is modeled by a single interacting level (site $i = 0$) weakly connected to two semi-infinite one dimensional electrodes ($i < 0$ and $i > 0$, respectively) at chemical potential $\mu_{l/r} = \pm V_b/2$, as depicted in Fig. 2. Its Hamiltonian reads

$$\hat{\mathbf{H}} = \sum_{i=-\infty}^{+\infty} \sum_{\sigma\in\{\uparrow,\downarrow\}} \left(\gamma_i \hat{\mathbf{c}}^\dagger_{i,\sigma}\hat{\mathbf{c}}_{i+1,\sigma} + \text{h.c.}\right) + \varepsilon_d\big(\hat{\mathbf{n}}_\uparrow + \hat{\mathbf{n}}_\downarrow\big) + U\,\Theta(t)\,\hat{\mathbf{n}}_\uparrow\hat{\mathbf{n}}_\downarrow \tag{8}$$

where the former sites i are now extended to include the spin $\sigma =\uparrow,\downarrow$ degree of freedom. ε_d is the on-site energy on the dot controlled by an electrostatic gate and $\hat{\mathbf{n}}_\sigma = \hat{\mathbf{c}}^\dagger_{0,\sigma}\hat{\mathbf{c}}_{0,\sigma}$. We work directly in the thermodynamic limit with an infinite number of bath sites. $\Theta(t)$ is the Heaviside function. The hopping between two neighboring sites is denoted by $\gamma_i = \bar{\gamma}$, except for the coupling to the dot $\gamma_0 = \gamma_{-1} = \gamma$. Energies are expressed in units of the tunneling rates to the dot $\Gamma = 2\gamma^2/\bar{\gamma}$. For concreteness, all calculations are performed at zero temperature.

Our goal will be to calculate the charge Q on the dot and the current I flowing from the dot to the right electrode after an interaction quench at $t = 0$.

$$Q(U,t) = \sum_{\sigma\in\{\uparrow,\downarrow\}} \left\langle \hat{\mathbf{c}}^\dagger_{0,\sigma}\hat{\mathbf{c}}_{0,\sigma}\right\rangle \tag{9}$$

$$I(U,t) = -i\gamma \sum_{\sigma\in\{\uparrow,\downarrow\}} \left[\left\langle \hat{\mathbf{c}}^\dagger_{1,\sigma}\hat{\mathbf{c}}_{0,\sigma}\right\rangle - \left\langle \hat{\mathbf{c}}^\dagger_{0,\sigma}\hat{\mathbf{c}}_{1,\sigma}\right\rangle\right]. \tag{10}$$

3 Interaction expansion (problem A)

This section explains the first step of the approach: how to reduce the calculation of $Q_n(t)$ to a multidimensional integral in terms of known objects.

3.1 A short reminder of Keldysh formalism

Our starting point for this work is a formal expansion of the out-of-equilibrium (Keldysh) Green function in powers of electron-electron interactions. This is a standard step [16] which we briefly sketch to introduce our notation. We would also like to demystify Keldysh formalism which is much simpler than often thought to be. In our opinion it is actually conceptually simpler than the imaginary-time or zero temperature formalism at the price of the actual calculations being a bit more cumbersome. Since these calculations will be performed by the computer, this is not necessarily an issue.

One starts by the standard step of "integrating out" the (supposedly known) dynamics of $\hat{\mathbf{H}}_0$, i.e., work in the interaction representation. Using the interaction representation, one defines $\hat{\mathbf{c}}_i(t) = \hat{\mathbf{U}}_0(0,t)\hat{\mathbf{c}}_i\hat{\mathbf{U}}_0(t,0)$ where $\hat{\mathbf{U}}_0(t',t)$ is the evolution operator from t to t' associated with $\hat{\mathbf{H}}_0$. For a time-independent $\hat{\mathbf{H}}_0$, it is simply $\hat{\mathbf{U}}_0(t',t) = e^{-i\hat{\mathbf{H}}_0(t'-t)}$. In this representation, one defines $\tilde{\mathbf{H}}_{\rm int}(\bar{u})$, which is equal to $\hat{\mathbf{H}}_{\rm int}(u)$ with the operators $\hat{\mathbf{c}}_i, \hat{\mathbf{c}}_j^\dagger$ replaced by $\hat{\mathbf{c}}_i(\bar{u}), \hat{\mathbf{c}}_j^\dagger(\bar{u})$. Starting from a non-interacting density matrix ρ_0 at $t = 0$, the average of an observable $\hat{\mathbf{O}}$ at time t is given by

$$\langle\hat{\mathbf{O}}\rangle = \mathrm{Tr}\left[Te^{+i\int d\bar{u}\, U\tilde{\mathbf{H}}_{\rm int}(\bar{u})}\,\tilde{\mathbf{O}}\,Te^{-i\int d\bar{u}\, U\tilde{\mathbf{H}}_{\rm int}(\bar{u})}\,\rho_0\right], \tag{11}$$

where T is the usual time-ordering operator (now necessary because in the interaction representation the interacting Hamiltonian *is* time-dependent even if the original Hamiltonian is not). The above equation is very natural; for instance if $\rho_0 = |\Psi_0\rangle\langle\Psi_0|$ then it corresponds to $\langle\hat{\mathbf{O}}\rangle = \langle\Psi(t)|\hat{\mathbf{O}}|\Psi(t)\rangle$ with $|\Psi(t)\rangle = Te^{-i\int d\bar{u}\, U\tilde{\mathbf{H}}_{\rm int}(\bar{u})}|\Psi_0\rangle$ the state of the system at time t. Looking at Eq. (11), we see that when we perform the expansion in powers of U, there will be two kinds of terms: the ones coming before the operator $\hat{\mathbf{O}}$ and the ones coming after it. To remember if a terms originates from one or the other (before of after), one introduces an index $a = 0$ (before) or $a = 1$ (after). This is the "Keldysh index" and that is all there is to it: the Keldysh formalism is merely a technique for book-keeping the position of the different terms in the expansion.

A bit more formally, one defines the contour ordering for pairs $\bar{t} = (t,a)$: $(t,0) < (t',1)$ for all t,t', $(t,0) < (t',0)$ if $t<t'$ and $(t,1) < (t',1)$ if $t>t'$. The contour ordering operator T_c acts on products of fermionic operators $A, B, C\ldots$ labeled by various "contour times" $\bar{t}_A = (t_A, a_A), \bar{t}_B, \bar{t}_C\ldots$ and reorder them according to the contour ordering: $T_c\, A(\bar{t}_A)B(\bar{t}_B) = AB$ if $\bar{t}_A>\bar{t}_B$ and $T_c\, A(\bar{t}_A)B(\bar{t}_B) = -BA$ if $\bar{t}_A<\bar{t}_B$. The non-interacting contour Green function is defined as

$$g^c_{ij}(\bar{t},\bar{t}') = -i\big\langle T_c\,\hat{\mathbf{c}}_i(\bar{t})\hat{\mathbf{c}}_j^\dagger(\bar{t}')\big\rangle, \tag{12}$$

where $\hat{\mathbf{c}}_i(\bar{t})$ is just $\hat{\mathbf{c}}_i(t)$, the Keldysh index serving only to define the position of the operator after contour ordering. The contour Green function has a matrix structure in a, a' which reads

$$g^c_{ij}(t,t') = \begin{pmatrix} g^T_{ij}(t,t') & g^<_{ij}(t,t') \\ g^>_{ij}(t,t') & g^{\bar{T}}_{ij}(t,t') \end{pmatrix} \tag{13}$$

where $g^T_{ij}(t,t')$, $g^<_{ij}(t,t')$, $g^>_{ij}(t,t')$ and $g^{\bar{T}}_{ij}(t,t')$ are respectively the time-ordered, lesser, greater, and anti-time-ordered Green functions. These non-interacting Green functions will form the actual input of our approach, we will briefly discuss how they are obtained in the next section. Finally, one defines the full Green function $G^c_{ij}(\bar{t},\bar{t}')$ with definitions identical to the above except that $\hat{\mathbf{U}}_0$ is replaced by $\hat{\mathbf{U}}$, the evolution operator associated to the full Hamiltonian $\hat{\mathbf{H}}$. The fundamental expression for $G^c_{ij}(\bar{t},\bar{t}')$ reads

$$G^c_{ij}(\bar{t},\bar{t}') = -i\big\langle T_c\, e^{-i\int d\bar{u}\, U\tilde{\mathbf{H}}_{\rm int}(\bar{u})}\, \hat{\mathbf{c}}_i(\bar{t})\hat{\mathbf{c}}^\dagger_j(\bar{t}')\big\rangle, \tag{14}$$

where the integral over $\bar{u}$ is taken along the Keldysh contour, i.e., increasing u for $a=0$ and decreasing for $a=1$. We are almost ready to perform the expansion.

3.2 Non-interacting Green functions

The dynamics of the non-interacting problem is, in principle, "trivial" in the sense that one simply needs to solve the one-body problem and "fill up" the states up to the Fermi energy. In practice it may not be entirely straightforward but there are well known and mature techniques to calculate both the stationary [17] and time-dependent properties [18]. There are also associated open source software such as Kwant [13] for the stationary problem and TKwant [14] for the time-dependent one. The latter explicitly supports the calculation of $g^T_{ij}(t,t')$, $g^<_{ij}(t,t')$, $g^>_{ij}(t,t')$ and $g^{\bar{T}}_{ij}(t,t')$.

One approach is to relate the non-interacting Green functions to the (Scattering) wave functions in the system [18]

$$g^<_{ij}(t,t') = i\sum_\alpha \int \frac{dE}{2\pi}\, f_\alpha(E)\, \Psi_{\alpha E}(t,i)\Psi^*_{\alpha E}(t',j). \tag{15}$$

Here, α labels the various propagating channels of the leads, $\Psi_{\alpha E}(t,i)$ the scattering state at energy E (in the electrode) and $f_\alpha(E)$ the corresponding Fermi distribution function. The greater Green function $g^>_{ij}(t,t')$ is obtained by replacing the Fermi functions $f(E)$ with $f(E)-1$. The actual calculations performed in this article are restricted to a stationary non-interacting system, where the above expression further simplifies to

$$g^<_{ij}(t-t') = i\sum_\alpha \int \frac{dE}{2\pi}\, f_\alpha(E)\, \Psi_{\alpha E}(i)\Psi^*_{\alpha E}(j)\, e^{-iE(t-t')}. \tag{16}$$

Here again, the stationary scattering wave functions $\Psi_{\alpha E}(i)$ are standard objects [17]. They are the eigenvectors of $\hat{\mathbf{H}}_0$ with one caveat: they need to be classified into states incoming from the

left and states incoming from the right. These objects are in fact direct outputs of the Kwant software [13]. Once the lesser and greater Green functions are known, one completes the 2×2 Keldysh matrix with the standard relations

$$g^T_{ij}(t,t') = \Theta(t-t')\, g^>_{ij}(t,t') + \Theta(t'-t)\, g^<_{ij}(t,t') \tag{17}$$
$$g^{\bar{T}}_{ij}(t,t') = \Theta(t'-t)\, g^>_{ij}(t,t') + \Theta(t-t')\, g^<_{ij}(t,t')\,. \tag{18}$$

3.3 The expansion itself

We continue to follow the literature and perform the expansion in powers of U. We obtain

$$G^c_{ij}(\bar{t},\bar{t}')=-i\sum_{n=0}^{+\infty}\frac{(-i)^n}{n!}U^n\sum_{\{a_i\}}(-1)^{\sum_i a_i}\int du_1\cdots du_n\left\langle T_c\tilde{\mathbf{H}}_{\text{int}}(\bar{u}_1)\tilde{\mathbf{H}}_{\text{int}}(\bar{u}_2)\cdots\tilde{\mathbf{H}}_{\text{int}}(\bar{u}_n)\hat{\mathbf{c}}_i(\bar{t})\hat{\mathbf{c}}^\dagger_j(\bar{t}')\right\rangle \tag{19}$$

and we are left to calculate the non-interacting average of a (possibly large) product of creation and destruction operators. This is done, as usual, using Wick's theorem. Wick's theorem states that an average over a non-interacting density matrix of a product of fermionic operators takes the form

$$\left\langle\hat{\mathbf{c}}^\dagger_1\hat{\mathbf{c}}_1\hat{\mathbf{c}}^\dagger_2\hat{\mathbf{c}}_2\cdots\hat{\mathbf{c}}^\dagger_M\hat{\mathbf{c}}_M\right\rangle = \sum_P(-1)^{|P|}\left\langle\hat{\mathbf{c}}^\dagger_1\hat{\mathbf{c}}_{P(1)}\right\rangle\left\langle\hat{\mathbf{c}}^\dagger_2\hat{\mathbf{c}}_{P(2)}\right\rangle\cdots\left\langle\hat{\mathbf{c}}^\dagger_M\hat{\mathbf{c}}_{P(M)}\right\rangle \tag{20}$$

where the sum runs over all the permutation P of M elements, $|P| = \pm1$ is its signature and $P(a)$ the image of a through the corresponding permutation. The numbers $1, 2, \ldots M$ are just shorthands for all the parameters of the fermionic operators: sites i (including spin), times u and Keldysh index a.

This is where the numerical route starts to differ from the analytical one. Usually, one would identify each permutation with a Feynman diagram and start do derive the corresponding Feynman rules. But as already discussed, the problem is that we would have far too many diagrams (there are $M!$ permutations in the above expression). For numerical purposes, we will make a simple remark: that the right-hand-side of Eq. (20) is actually the definition of the determinant of a $M\times M$ matrix:

$$\left\langle\hat{\mathbf{c}}^\dagger_1\hat{\mathbf{c}}_1\hat{\mathbf{c}}^\dagger_2\hat{\mathbf{c}}_2\cdots\hat{\mathbf{c}}^\dagger_M\hat{\mathbf{c}}_M\right\rangle = \det\left\langle\hat{\mathbf{c}}^\dagger_a\hat{\mathbf{c}}_b\right\rangle. \tag{21}$$

Since the calculation of the determinant of a $M\times M$ matrix takes only $M^3 \ll M!$ operations, this sounds very appealing. Now putting things together, we arrive at a formula that is conceptually very simple: the left-hand-side is what we want and the right-hand-side is a set of multi-dimensional integrals of matrices whose entries are known. The problem is therefore "reduced to quadrature"

$$G^c_{ij}(\bar{t},\bar{t}') =\sum_{n=0}^{+\infty}\frac{i^n}{n!}U^n\sum_{\{a_i\}}(-1)^{\sum_i a_i}\int du_1\cdots du_n\sum_{i_1j_1k_1l_1}V_{i_1j_1k_1l_1}(u_1)\cdots\sum_{i_nj_nk_nl_n}V_{i_nj_nk_nl_n}(u_n)\det\mathbf{M}_n \tag{22}$$

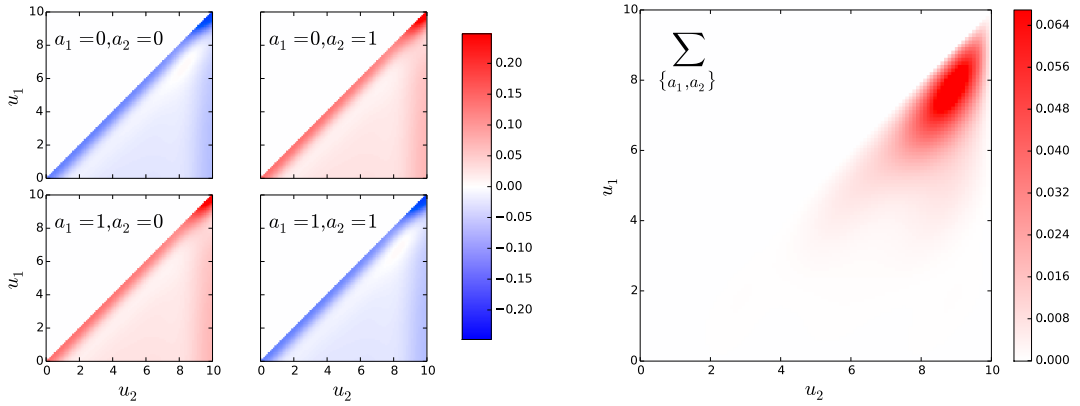


Fig. 3: *Left: Colorplot of the integrand of Q_2 as a function of the two times u_1 and u_2 for SIAM with μ_L=μ_R=0, ε_d=0, T=0 and t=10. The four panels correspond to the four possible values of the two Keldysh indices a_1 and a_2. The explicit form of the integrand is $f(u_1,u_2,a_1,a_2) = -\mathrm{Im}(-1)^{\sum_i a_i} \det \boldsymbol{M}_2(u_1,u_2,a_1,a_2)$. Right: Same parameters as on the left but the integrand has now been summed over Keldysh indices. The colorplot represents $f(u_1,u_2) = i\sum_{a_1,a_2}(-1)^{\sum_i a_i} \det \boldsymbol{M}_2(u_1,u_2,a_1,a_2)$ (f is real). Note that the integrand is now real, positive and concentrated around u_1=u_2=t. (Adapted from [4])*

where the $(2n+1)\times(2n+1)$ matrix $\mathbf{M}_n$ is given by

$$\mathbf{M}_n = \begin{pmatrix} g^<_{k_1 i_1}(\bar{u}_1,\bar{u}_1) & g^>_{k_1 j_1}(\bar{u}_1,\bar{u}_1) & g^c_{k_1 i_2}(\bar{u}_1,\bar{u}_2) & \cdots & g^c_{k_1 j}(\bar{u}_1,\bar{t}') \\ g^<_{l_1 i_1}(\bar{u}_1,\bar{u}_1) & g^<_{l_1 j_1}(\bar{u}_1,\bar{u}_1) & g^c_{l_1 i_2}(\bar{u}_1,\bar{u}_2) & \cdots & g^c_{l_1 j}(\bar{u}_1,\bar{t}') \\ g^c_{k_2 i_1}(\bar{u}_2,\bar{u}_1) & g^c_{k_2 j_1}(\bar{u}_2,\bar{u}_1) & g^<_{k_2 i_2}(\bar{u}_2,\bar{u}_2) & \cdots & g^c_{k_2 j}(\bar{u}_2,\bar{t}') \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g^c_{k_n i_1}(\bar{u}_n,\bar{u}_1) & g^c_{k_n j_1}(\bar{u}_n,\bar{u}_1) & g^c_{k_n i_2}(\bar{u}_n,\bar{u}_2) & \cdots & g^c_{k_n j}(\bar{u}_n,\bar{t}') \\ g^c_{l_n i_1}(\bar{u}_n,\bar{u}_1) & g^c_{l_n j_1}(\bar{u}_n,\bar{u}_1) & g^c_{l_n i_2}(\bar{u}_n,\bar{u}_2) & \cdots & g^c_{l_n j}(\bar{u}_n,\bar{t}') \\ g^c_{i i_1}(\bar{t},\bar{u}_1) & g^c_{i j_1}(\bar{t},\bar{u}_1) & g^c_{i i_2}(\bar{t},\bar{u}_2) & \cdots & g^c_{i j}(\bar{t},\bar{t}') \end{pmatrix} \tag{23}$$

and the zeroth order term is $g^c_{ij}(\bar{t},\bar{t}')$. [Note that there was a typo in the above expression in [4], see [5] for the correction]. To calculate an actual observable, we just recognize that it corresponds to the lesser Green function at equal times, i.e.,

$$\left\langle \hat{\mathbf{U}}(0,t)\, \hat{\mathbf{c}}^\dagger_i \hat{\mathbf{c}}_j\, \hat{\mathbf{U}}(t,0) \right\rangle = -iG^<_{ji}(t,t). \tag{24}$$

We are just left with a simple problem: to obtain $Q_n(t)$ we must integrate over n times and sum over n Keldysh indices (in general we also need to sum over the n different vertices V_{ijkl} but in the case of SIAM, this sum reduces to a single term). There is a caveat however. To see it, let us look at the integrand of Q_2 in the 4-sectors defined by the two Keldysh indices, as shown in the left panel of Fig. 3. We immediately see that the corresponding integrals are not going to behave well: the integrand does not seem to decay when the time gets away from the time where the measurement is made (reminder: we switch on the interaction at $t = 0$ and measure, here, at $t = 10$). This is to be expected and is why the determinant form of the Wick theorem was not super successful before: the expansion includes all Feynman diagrams, *including* the disconnected diagrams. It is well known that these disconnected diagrams do not contribute to

the final result. Yet, in this expansion they are present and a priori only cancel at the end of the calculation after integration and summation over Keldysh indices.
The magic occurs when one performs the summation over the Keldysh indices, as shown in the right panel of Fig. 3: we find that the cancellation of the disconnected diagrams occurs *before* the integration over times. The proof is a bit technical and is shown in the appendix of [4]. The consequence of this property is that, one is left with a well behaved integral to calculate. The cost of each call to the integrand is $O(2^n)$ which is still exponential but very mild compared to the initial $n!$ cost we started with. To calculate these integrals we will use Tensor Cross Interpolation, which we now explain.

4 Tensor cross interpolation for integration (problem B)

We will now discuss a very important algorithm – Tensor Cross Interpolation (TCI) – that has a very special place in the zoo of tensor network algorithms. This algorithm takes as an input a "virtual" tensor $F_{\sigma_1,\sigma_2,\ldots,\sigma_N}$ where each index σ_i takes d_i different values (for simplicity we assume that $d_i \equiv d$ is constant). It returns as an output a Matrix Product State (MPS) that approximates $F_{\vec{\sigma}}$ in the best possible way. For an introduction to MPS, see [19]. For the purpose of this article, the MPS is merely the following expression

$$F_{\sigma_1,\sigma_2,\ldots,\sigma_N} \approx \sum_{\{\alpha_i\}} M^1_{\alpha_1}(\sigma_1) M^2_{\alpha_1\alpha_2}(\sigma_2) \cdots M^N_{\alpha_{N-1}}(\sigma_N), \tag{25}$$

where the matrices $M_a(\sigma)$ have a maximum size χ, known as the bond dimension. $F_{\vec{\sigma}}$ is virtual in the sense that the input of the algorithm is *not* the actual tensor (which would be an exponentially large object with d^N elements). Rather it is a function that takes $\vec{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_N)$ as an input and returns the corresponding value $F_{\vec{\sigma}}$. TCI is very different from many other tensor network algorithms (e.g. DMRG [19]): here $F_{\vec{\sigma}}$ is actually known by the user, what is not known is its MPS representation. Once one has this MPS, one can use it for calculating, e.g., integrals or plenty of other applications that we shall not discuss, see [15]. TCI is really a "gateway" that allows one to take a problem that is *not* formulated in terms of tensor network and transform it into this framework.
Another peculiarity of TCI is that it is a *learning* algorithm akin to what is done in machine learning. More precisely it is an active learning algorithm since TCI decides on the data $(\vec{\sigma}, F_{\vec{\sigma}})$ that will be requested. As in machine learning, only a very tiny fraction of the possible configurations $\vec{\sigma}$ will be explored, and as in machine learning the fact that the resulting model interpolates correctly between the configurations can be spectacular. On the other hand there are strong differences with deep neural networks: the optimization has nothing to do with gradient descent (and is way more effective) and the resulting function much more structured (for instance we can easily calculate, e.g., integrals, we cannot do that with a neural network). The cost for these added features is a more restrictive set of applications: TCI is only effective for problems where the level of "entanglement" is limited (small χ).

The presentation below is mostly based on section III of [8] with a few more advanced aspects borrowed from [15] to which we also refer for the references to the original literature. The readers can also have a look at the tensor4all open source library that implements these algorithms https://tensor4all.org.

4.1 Compressing low rank matrices

Before we can get into TCI, we need a matrix factorization formula for low rank (or approximately low rank) matrices that is based on Gaussian elimination and the concept of the Schur complement [20]. This formula (the "cross interpolation") will be almost as good as the Singular Value Decomposition (SVD, which is optimum) but with a key advantage: it can be performed without the need to access the full matrix A: only a set of χ rows and columns will be needed.

4.1.1 Revisiting Gaussian elimination

We consider an arbitrary matrix A that we put in a 2×2 block form,

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \tag{26}$$

Following the strategy of Gaussian elimination, we can put this matrix in triangular form as (provided the A_{11} block is invertible)

$$\begin{pmatrix} 1 & 0 \\ -A_{21}A_{11}^{-1} & 1 \end{pmatrix} \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22}-A_{21}A_{11}^{-1}A_{12} \end{pmatrix}. \tag{27}$$

We can proceed on the columns to eliminate the A_{12} block and finally obtain

$$\begin{pmatrix} 1 & 0 \\ -A_{21}A_{11}^{-1} & 1 \end{pmatrix} \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} 1 & -A_{11}^{-1}A_{12} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} A_{11} & 0 \\ 0 & A_{22}-A_{21}A_{11}^{-1}A_{12} \end{pmatrix}. \tag{28}$$

This equation will play a key role in multiple places. The quantity $A_{22}-A_{21}A_{11}^{-1}A_{12} \equiv [A/A_{11}]$ will also appear over and over and we shall therefore give it its name: it is called the Schur complement $[A/A_{11}]$ of A with respect to the 11 block. The block triangular matrices can be trivially inverted and we arrive at a "block LDU" decomposition $A = LDU$ in terms of a block lower triangular L, block diagonal D and block upper triangular matrix U

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ A_{21}A_{11}^{-1} & 1 \end{pmatrix} \begin{pmatrix} A_{11} & 0 \\ 0 & [A/A_{11}] \end{pmatrix} \begin{pmatrix} 1 & A_{11}^{-1}A_{12} \\ 0 & 1 \end{pmatrix}. \tag{29}$$

Among the various corollaries of this equation, it provides a close form for the determinant:

$$\det A = \det[A_{11}]\ \det[A/A_{11}] \tag{30}$$

The Schur complement has many other nice properties, see [15] for a discussion. For instance one does not need to take the Schur complement directly with respect to en entire block A_{11}, one may do it sub-block after sub-block and if one does so, the order in which one takes the Schur complements does not matter.

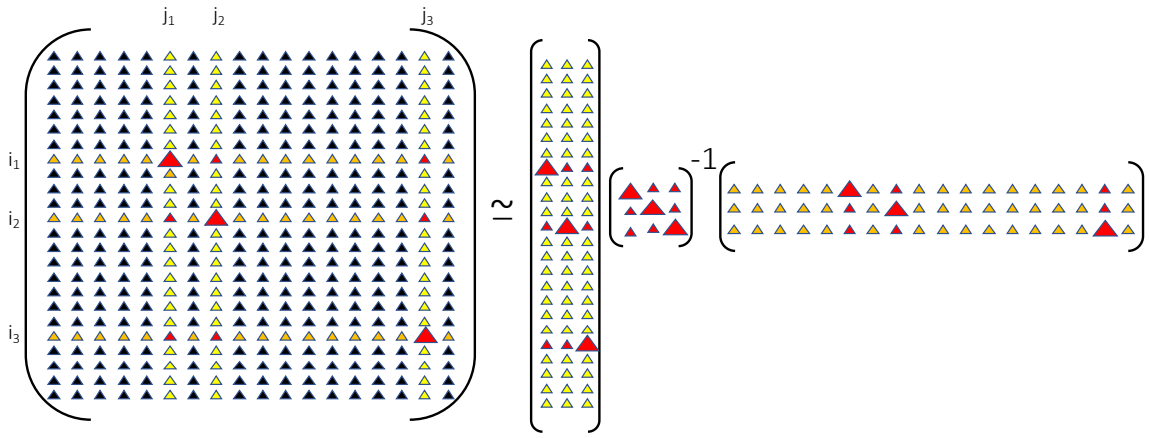


Fig. 4: *Illustration of the cross interpolation (CI) of a matrix. The large red triangles indicate real pivots and the smaller red triangles indicate automatically generated pivots. The right-hand side only contains small subparts of the matrix. (Adapted from [8])*

4.1.2 Cross Interpolation

The cross interpolation formula approximates $A \approx A_{\text{CI}}$, where A_{CI} is defined as

$$A_{\text{CI}} = \begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} (A_{11})^{-1} \begin{pmatrix} A_{11} & A_{12} \end{pmatrix}. \tag{31}$$

In other words, the Schur complement is the *error* of the cross interpolation,

$$A = A_{\text{CI}} + \begin{pmatrix} 0 & 0 \\ 0 & [A/A_{11}] \end{pmatrix}. \tag{32}$$

An important remark is that to construct A_{CI}, one does not need to know anything about A_{22}. Indeed, when a matrix is of (low) rank χ, we only need χ independent vectors (the first matrix in the definition of A_{CI}) and χ rows (which tells us how the other vectors decompose in terms of the independent ones). The cross interpolation formula has two important properties: (*i*) first it is exact when evaluated on the blocks that have been used to construct it (A_{11}, A_{12} and A_{12}) as evident in the above equation. We refer to this as the interpolation property. (*ii*) Second it is exact if A_{11} is a $\chi \times \chi$ matrix and A is exactly of rank χ. To prove this second assertion, we construct the sub-matrix of A that contains the 11 block plus a single extra row i_0 and a single extra column j_0. Using the Schur complement, we have:

$$\left| \det \begin{pmatrix} A_{11} & A_{1j_0} \\ A_{i_0 1} & A_{i_0 j_0} \end{pmatrix} \right| = \left| \det A_{11} \right| \times \left| A_{i_0 j_0} - A_{i_0 1} A_{11}^{-1} A_{1 j_0} \right|. \tag{33}$$

(with a slight abuse of notations that mixes indexing with block indexing). The left hand side is zero by definition of A being of rank χ (it is a $(\chi{+}1){\times}(\chi{+}1)$ matrix) hence $A_{i_0 j_0} - A_{i_0 1} A_{11}^{-1} A_{1 j_0} = 0$, i.e., the cross interpolation is exact.

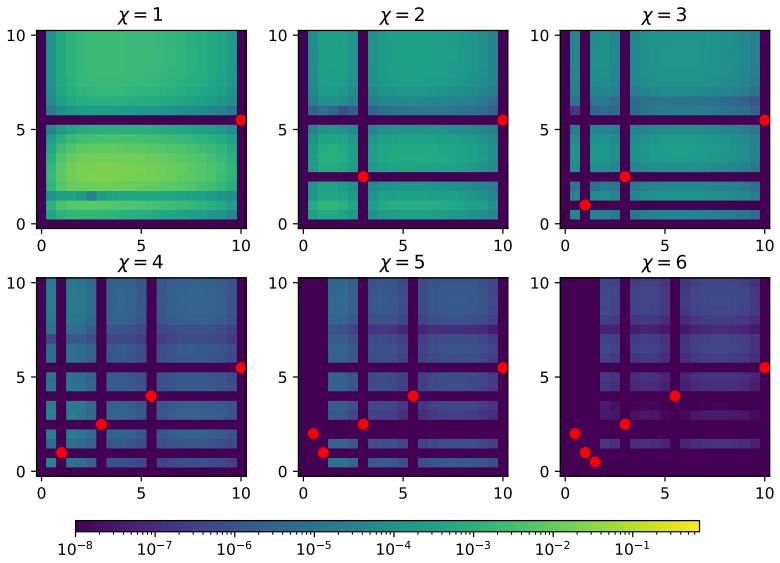


Fig. 5: *Error* $\left|A_{ij}-[A_{\rm CI}]_{ij}\right|$ *versus* i *and* j *at different stages of the Cross-Interpolation for an* $M\times M$ *matrix with* $M=20$. *In this toy example,* $A_{ij} = \left(\frac{i/M}{i/M+1}\right)^4 \left(1+e^{-(j/M)^2}\right)\left[1+(j/M)\cos(j/M)e^{-(j/M)\frac{i/M}{(i/M)+1}}\right]$. *The red dots indicate the pivots. The* x *and* y *axis have been rescaled to be in* $[0,10]$. *(Adapted from Jeannin et al. [10])*

4.1.3 Practical cross interpolation

In practice to build up $A_{\rm CI}$ we need to choose the A_{11} block properly. Let us introduce the notation that we will use to design the chosen rows and columns. Let $\mathcal{I} = \{i_1, i_2, \ldots, i_\chi\}$ (respectively $\mathcal{J} = \{j_1, j_2, \ldots, j_\chi\}$) denote a list of the rows (columns) of A (that will form the A_{11} block). Indexing these sets gives the corresponding index: $\mathcal{I}_a \equiv i_a$ is its a$^{\rm th}$ element. The list of the indices of all rows (columns) is denoted $\mathbb{I} = \{1, 2, \ldots, M\}$ ($\mathbb{J} = \{1, 2, \ldots, N\}$). Following usual programming convention (as in Python/MATLAB/Julia), we denote by $A(\mathcal{I}, \mathcal{J})$ the submatrix of A comprised of the rows $\mathcal{I}$ and columns $\mathcal{J}$; $A(\mathcal{I}, \mathcal{J})_{ab} \equiv A_{\mathcal{I}_a, \mathcal{J}_b}$. We have

$$A = A(\mathbb{I}, \mathbb{J}) \tag{34}$$

$$A_{\rm CI} = A(\mathbb{I}, \mathcal{J})A(\mathcal{I}, \mathcal{J})^{-1}A(\mathcal{I}, \mathbb{J}) \tag{35}$$

Equation (35) is illustrated graphically in Fig. 4. The rows and columns of $A(\mathcal{I}, \mathcal{J})$ are called the *pivots* and $A(\mathcal{I}, \mathcal{J})$ is the *pivot matrix*. The pivots are chosen one by one iteratively in such a way as to maximize the determinant of the matrix $A(\mathcal{I}, \mathcal{J}) = A_{11}$ in order to guarantee that the chosen vectors are truly independent. This is known as the maximum volume (maxvol) principle. Another way to look at the maxvol principle is that each new pivot is chosen to be the one where the current error of $A_{\rm CI}$ is maximum (maxerror) so that adding this pivot brings the largest amount of new information into the approximation. The proof of the equivalence between maxvol and maxerror is in Eq. (33). A practical example of how the error decreases for the cross extrapolation of a (toy) matrix is shown in Fig. 5.

The important thing to remember about cross interpolation is that it is given in terms of slices of the matrix A: it is entirely defined in terms of the two lists $\mathcal{I}$ and $\mathcal{J}$ of the rows and columns of the pivot matrices.

4.1.4 Stable evaluation of the cross interpolation

We need a last ingredient to be able to use cross interpolation in practice. Indeed, as one adds more pivots, the A_{11} matrix becomes increasingly singular so we do not want to calculate A_{11}^{-1} explicitly as it becomes numerically unstable (even for moderate values of χ). There are several ways to stabilize the evaluation of

$$\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} (A_{11})^{-1}. \tag{36}$$

The first is to perform a QR factorization of the first matrix, writing

$$\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} = \begin{pmatrix} Q_{11} \\ Q_{21} \end{pmatrix} R\,. \tag{37}$$

The Q matrix being an isometry, it is well conditioned. All the (possibly very small) singular values of A_{11} are in the triangular matrix R which disappears from the calculation. Indeed, we have

$$\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} (A_{11})^{-1} = \begin{pmatrix} 1 \\ Q_{21}Q_{11}^{-1} \end{pmatrix}. \tag{38}$$

The second way to stabilize this calculation (now our preferred way) is to realize that Eq. (29) can be rewritten as

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{21}A_{11}^{-1}A_{12} \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & [A/A_{11}] \end{pmatrix} \tag{39}$$

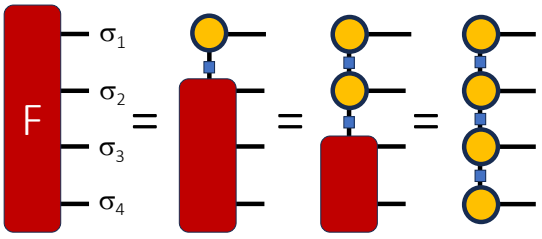
in other words, if one ignores the Schur complement one is left with the cross interpolation, Eq. (32). We can use Eq. (29) iteratively, performing the decomposition pivot after pivot (as stated above, this is legit, the proof can be found in [15]) building a decomposition in the form $A_{\mathrm{CI}} = LDU$ where L is lower triangular, D diagonal and U upper triangular. This is nothing but the celebrated LU decomposition used for, e.g., inverting matrices. The only caveat is that it is partial (we stop it after getting the χ pivots, we do not go all the way through) and it is rank revealing (we use the maxvol criteria to select the pivots). It is the prrLU (partial rank revealing LU) decomposition. But again, this is just a neat way to obtain the cross interpolation in a stable way.

4.2 TCI: Extension of CI to n-dimensional tensors

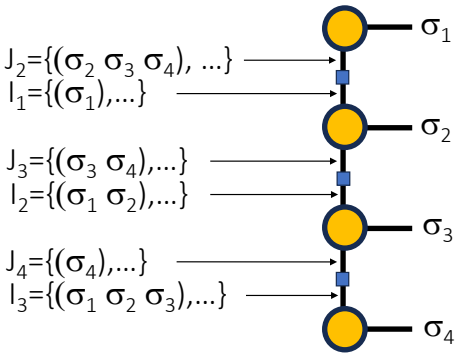
We now have everything we need to factorize matrices, we need to extend cross interpolation to tensors. This is what TCI does.

4.2.1 TCI: naive approach

We can decompose any tensor using cross interpolation iteratively. We first group together all indices except σ_1, apply cross interpolation on the resulting matrix and repeat the procedure until the tensor has been entirely factorized. Graphically this (very naive) algorithm has the following form:



where the small blue squares stand for the inverse of the pivot matrices. This algorithm is not practical since applying cross interpolation on an exponentially large matrix requires an exponentially large amount of memory and computing time. However, it has the merit of showing that such a decomposition exists. More interestingly, it shows the structure of the "pivots" of a TCI representation. Indeed, the cross interpolation is defined in terms of the lists $\mathcal{I}$ and $\mathcal{J}$. Now we have one of such list on each side of the pivot matrices (the blue squares above). Each element of this list is now a list itself that contains the value of the corresponding indices. We call such a list a *multi-index*. More explicitly, we have for our example,



The most tricky thing about writing a TCI code is to correctly do the book keeping of these lists of lists.

4.2.2 TCI: formal form

Let us introduce our notations a bit more formally. A TCI representation is essentially a MPS but we keep the pivot matrices explicit so that the TCI is entirely made of "slices" of the original tensor. For any α such that $1 \leq \alpha \leq N$, we consider "row" multi-indices $(\sigma_1, \sigma_2, \ldots, \sigma_\alpha)$ and "column" multi-indices $(\sigma_\alpha, \sigma_{\alpha+1}, \ldots, \sigma_N)$. The pivot lists are defined as $\mathcal{I}_\alpha = \{i_1, i_2, \ldots, i_\chi\}$ for the "rows" (the multi-indices have size α) and $\mathcal{J}_\alpha = \{j_1, j_2, \ldots, j_\chi\}$ for the "columns" (the multi-indices have size $N-\alpha+1$). For notational convenience, we define $\mathcal{I}_0$ and $\mathcal{J}_{N+1}$ as

singleton sets each comprised of an empty multi-index. Last, we use the symbol $\oplus$ to denote the concatenation of multi-indices

$$(\sigma_1, \sigma_2, \ldots, \sigma_{\alpha-1}) \oplus (\sigma_\alpha) \oplus (\sigma_{\alpha+1}, \ldots, \sigma_N) \equiv (\sigma_1, \ldots, \sigma_N). \tag{40}$$

We are now ready to define the TCI representation formally. The definitions are a bit scary looking but they are nothing else than what we obtained above using the naive algorithm. The blue squares are the pivot matrices P_α defined as,

$$[P_\alpha]_{ij} \equiv F_{[\mathcal{I}_\alpha]_i \oplus [\mathcal{J}_{\alpha+1}]_j} \tag{41}$$

(with $P_N = 1$ for notation convenience). Likewise the orange three leg tensors T_α are defined as,

$$[T_\alpha]_{i\sigma j} \equiv F_{[\mathcal{I}_{\alpha-1}]_i \oplus \sigma \oplus [\mathcal{J}_{\alpha+1}]_j}. \tag{42}$$

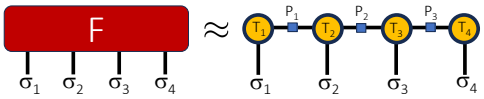
We also introduce the matrix $T_\alpha(\sigma)$ defined as

$$T_\alpha(\sigma)_{ij} \equiv [T_\alpha]_{i\sigma j} \tag{43}$$

to make contact with the standard MPS form. Using these notations, we have,

$$F_{\vec{\sigma}} \approx [F_{\text{TCI}}]_{\vec{\sigma}} \equiv \prod_{\alpha=1}^{N} T_\alpha(\sigma_\alpha) P_\alpha^{-1}. \tag{44}$$

or graphically



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 $F_{\vec{\sigma}}$ amounts to optimizing the selection of $\mathcal{I}_\alpha$ and $\mathcal{J}_\alpha$ for $1 \leq \alpha \leq N$. Only $O(Nd\chi^2) \ll d^N$ entries of $F_{\vec{\sigma}}$ are used in the approximation.

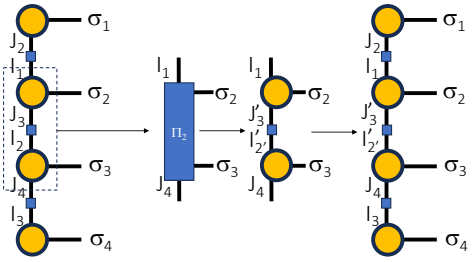
4.2.3 Practical TCI algorithm

We start with an initial point $(\sigma_1, \ldots, \sigma_n)$ which we split in $N-1$ different ways $(\sigma_1, \ldots, \sigma_N) = (\sigma_1, \ldots, \sigma_\alpha) \oplus (\sigma_{\alpha+1}, \ldots, \sigma_N)$ 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 tensor $F_{\vec{\sigma}}$ factorizes as a product of tensors of one variable.

To improve on this TCI, we are going to sweep over pairs of tensors $(T_\alpha, T_{\alpha+1})$ as is done in two-site DMRG. The sweeping is performed until convergence. For each pair, we use the following procedure: First, we introduce yet another tensor, $\mathit{\Pi}_\alpha$ as

$$[\mathit{\Pi}_\alpha]_{i\sigma\sigma' j} \equiv F_{[\mathcal{I}_{\alpha-1}]_i \oplus \sigma \oplus \sigma' \oplus [\mathcal{J}_{\alpha+2}]_j}. \tag{45}$$

Second, we replace $\left[T_\alpha(\sigma_\alpha) P_\alpha^{-1} T_{\alpha+1}(\sigma_{\alpha+1})\right]_{ij}$ inside the TCI by $[\mathit{\Pi}_\alpha]_{i\sigma_\alpha\sigma_{\alpha+1}j}$ because the former is a cross interpolation of the latter, hence we might as well use the more precise form. Next, we continue the cross interpolation of $\mathit{\Pi}_\alpha$ (seen as a matrix $[\mathit{\Pi}_\alpha]_{i\otimes\sigma,\sigma'\otimes j}$) by adding a new pivot, i.e., one new entry to the list $\mathcal{I}_\alpha$ and $\mathcal{J}_{\alpha+1}$. Graphically, we have



and that is it; this is a fully functional TCI algorithm (although there are versions that are more suitable for certain purposes). During the sweeping, we monitor the so-called pivot error between the $\mathit{\Pi}_\alpha$ tensor and its cross interpolation. We stop the iteration when this error is below a certain threshold during an entire sweep

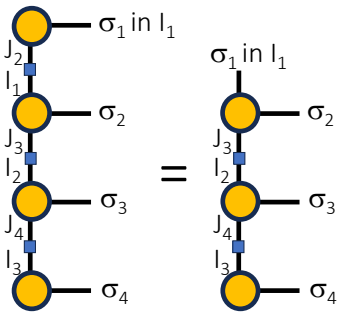
$$\epsilon_{\mathit{\Pi}} = \max_{i\sigma\sigma' j} \left| [\mathit{\Pi}_\alpha]_{i\sigma\sigma' j} - \left[T_\alpha(\sigma) P_\alpha^{-1} T_{\alpha+1}(\sigma')\right]_{ij} \right| . \tag{46}$$

Now, there is a subtle point that we have swept under the rug: the fact that the error $\epsilon_{\mathit{\Pi}}$ is *actually* the error of the TCI approximation for the corresponding pivots,

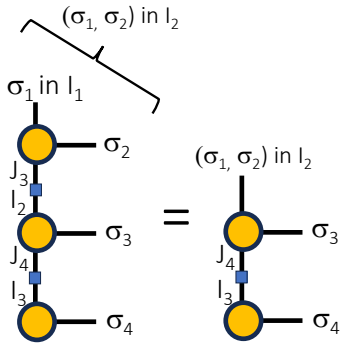
$$\epsilon_{\mathit{\Pi}} = \max_{i\sigma\sigma' j} \left| F_{[\mathcal{I}_{\alpha-1}]_i \oplus \sigma \oplus \sigma' \oplus [\mathcal{J}_{\alpha+2}]_j} - [F_{\mathrm{TCI}}]_{[\mathcal{I}_{\alpha-1}]_i \oplus \sigma \oplus \sigma' \oplus [\mathcal{J}_{\alpha+2}]_j} \right| . \tag{47}$$

Therefore improving the cross interpolation of $\mathit{\Pi}_\alpha$ does indeed improve the TCI approximation itself (at least for these pivots). To prove this point, we need to remember that the cross interpolation is exact on the pivots. We also need to realize that there is a form of "nesting condition" that connects the different pivot lists: a pivot $i_\alpha \in \mathcal{I}_\alpha$ takes the form $i_\alpha = i_{\alpha-1} \oplus \sigma_\alpha$ with $i_{\alpha-1} \in \mathcal{I}_{\alpha-1}$ (and a similar condition for the $\mathcal{J}_\alpha$). Using these two ingredients, one easily sees that there is a telescopic condition for the restriction of the TCI on these pivots.

Let us see how this works concretely. We start by restricting σ_1 to values that belong to $\mathcal{I}_1$. For these values, T_1 and P_1 cancel due to the interpolation property. Schematically, it reads



We continue by requesting that $\sigma_1 \oplus \sigma_2 \in \mathcal{I}_2$ which we can do because of the nested condition. The interpolation property implies that



and we can continue like that down the TCI representation. Since the same thing can be done with the $\mathcal{J}_\alpha$, we can also go up from the bottom of the TCI. See [8] or [15] for a more formal proof of the statement.

4.2.4 Application to integrals

There are many things that one may do with a TCI representation. As stated, it is an entry point to be able to use the many algorithms that have been developed for many-body physics. For the purpose of these notes, we are interested in multi-dimensional integration. It is an alternative to the Monte Carlo approach. When it works (the convergence with χ will depend on the integrand), it compares very favorably to Monte Carlo in two aspects: (*i*) the convergence is much faster than what is allowed by the law of large numbers; (*ii*) it is immune to the sign problem that plagues Monte Carlo whenever the integrand has an oscillatory behavior.
In its plainest version, multi-dimensional integration is quite straightforward. Let us consider a function $f(u_1, \ldots, u_n)$ (our integrand). We discretize it using a plain quadrature rule with d points per dimension $a_1, \ldots, a_d$ and the corresponding weight $w_1, \ldots, w_d$. For instance, we could use the Gauss-Kronrod-21 rule (with $d = 21$) or even the trapezoidal rule. We write

$$\int du_1 \cdots du_n f(u_1, \ldots, u_n) \approx \sum_{\sigma_1 \cdots \sigma_n} F_{\vec{\sigma}}\, w_{\sigma_1} \cdots w_{\sigma_N} \tag{48}$$

with

$$F_{\vec{\sigma}} \equiv f(a_{\sigma_1}, \ldots, a_{\sigma_n}). \tag{49}$$

The problem, of course, is that the sum runs over d^n different configurations which is impractical. This is known as the curse of dimensionality. If, however, we can factorize $F_{\vec{\sigma}}$ using TCI, then calculating this sum reduces to n matrix-vector multiplication. It becomes essentially trivial

$$\sum_{\sigma_1 \cdots \sigma_n} F_{\vec{\sigma}}\, w_{\sigma_1} \cdots w_{\sigma_n} \approx \prod_{\alpha=1}^{n} \sum_{\sigma} T_\alpha(\sigma) P_\alpha^{-1}. \tag{50}$$

This is essentially what we do to calculate our coefficients Q_n. There is a small difficulty that we will not discuss in details though: we do not perform TCI $f(u_1, \ldots, u_n)$ (whose rank is very high) but rather $f(u_1, u_2{-}u_1, u_3{-}u_2, \ldots, u_n{-}u_{n-1})$ (whose rank is much smaller). The reason is that the later function automatically enforces time-ordering – or, more precisely, works in a single sector of the ordering (see the corresponding discussion in [8]).

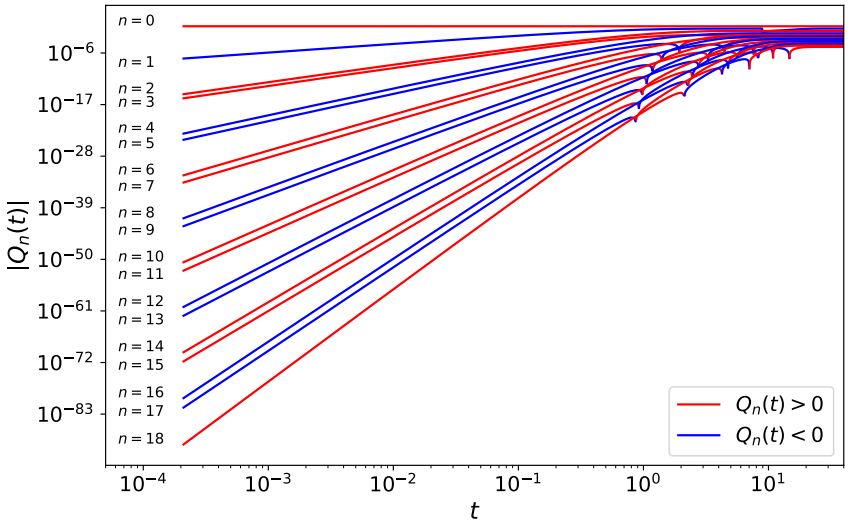


Fig. 6: *Coefficients of the expansion. Absolute value of the coefficients Q_n as a function of time for the charge ($\varepsilon_d = 0$). Red (blue) portion of curve corresponds to positive (negative) values of Q_n. At large time $Q_n(t)$ reaches its known exact value (calculated with Bethe ansatz) with high precision. (Adapted from [10])*

5 Summing up the series (problem C) and results

We are now in position to actually calculate these coefficients $Q_n(t)$. An example of the typical raw data coming out of such a calculation is shown in Fig. 6 up to order eighteen. Please note the span of the x and y axis: times span four orders of magnitude while the coefficients themselves span almost ninety orders of magnitude. The final result (at infinite time) is correct in this instance with close to eight digits precision. These results are orders of magnitude more precise and faster than what was obtained with previous techniques. We have a last problem to solve (problem C) before we can turn to do a little physics.

5.1 Cross extrapolation

Due to the convergence properties of the series of $Q(U,t)$ we can only calculate it accurately in two regimes: small times (but up to large U) and small U (but any t). We would like to extrapolate our results up to the interesting regime where both t and U are large. Let us first discretize U and t to get a matrix $Q_{ij} = Q(U_i, t_j)$. Only a few first rows and columns of this matrix are known.

The cross-extrapolation algorithm [9] is a deceptively simple idea: one merely applies the cross-interpolation formula to the matrix Q_{ij} taking advantage of the fact that the formula *does not use* the entire matrix. The only difference with what was done in the context of TCI is that one restricts the choices of the pivots to the known sector of the matrix (the A_{11} block). This is illustrated in Fig. 7 for a toy example. This is the same matrix as in Fig. 5, the only difference is that we supposed that the data inside the white square was inaccessible, hence we had to restrict the choice of the pivots to inside the red square. The convergence is not as fast or as stable as in a regular cross interpolation, this is to be expected, but in many instance the method works well

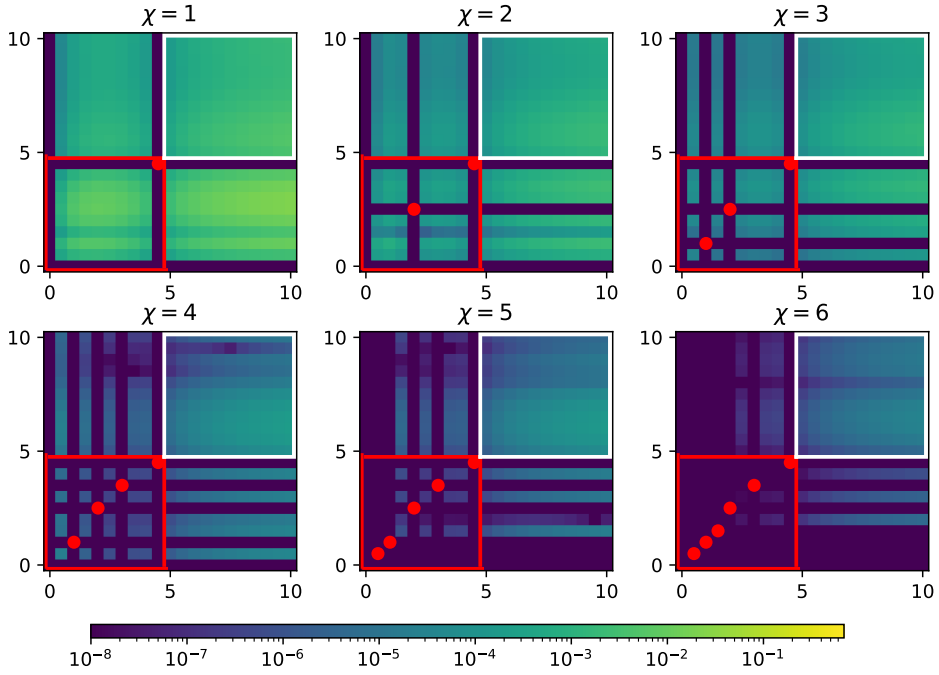


Fig. 7: *Relative error of the cross-extrapolation versus i and j for a toy matrix and different number of pivots $\chi = 1 - 6$. The pivots can only be placed inside the red square in order to extrapolate the results inside the white square. (Adapted from [9])*

and allows one to extrapolate the data further than the initial calculation provides. The error is controlled by varying the rank χ of the extrapolation and the size of the red square region. As in all extrapolation, there is an underlying assumption that must be checked: the fact that $Q(U,t)$ is approximately low rank or in other words that for an error level ϵ, there exist a set of χ one dimensional functions g_k and h_k such that

$$\left|Q(U,t) - \sum_{k=1}^{\chi} g_k(U)h_k(t)\right| < \epsilon. \tag{51}$$

The approach is very general and could in principle be used in many different situations. For instance in Fig. 8, we use it to extrapolate an image whose upper right corner is missing.

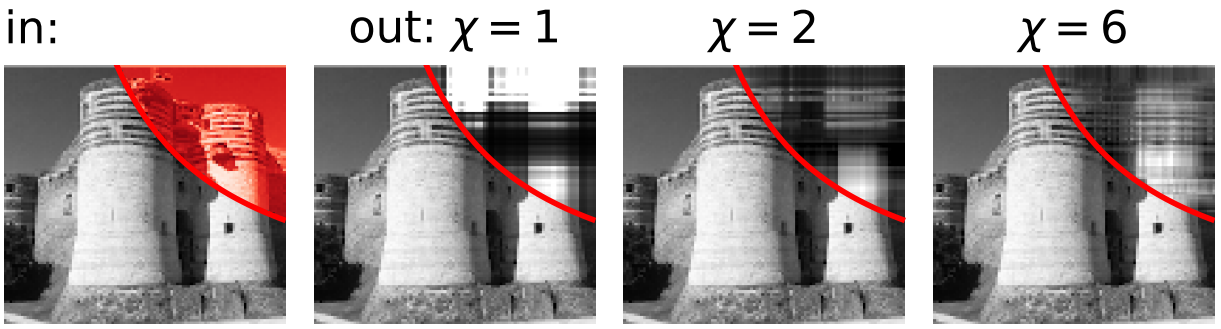


Fig. 8: *Example of cross extrapolation for an image (left) whose upper right corner is missing (reddish zone). In this instance, it does not work so well. (Adapted from [9])*

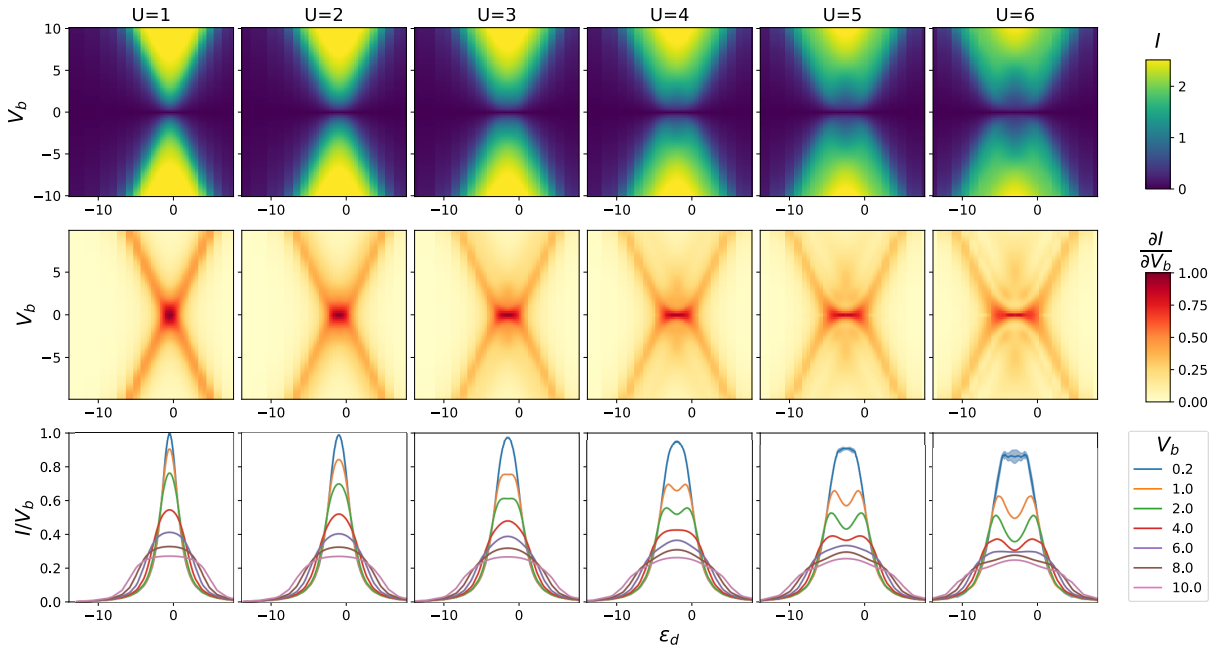


Fig. 9: *Final result of the calculation after cross extrapolation. Upper panels: color plots of the current versus bias voltage V_b and gate voltage ε_d for the SIAM. The diamond like shape shown is known as the "Coulomb diamond" Middle panels: same data but the differential conductance is shown. the horizontal red line around $V_b = 0$ is known as the "Kondo ridge" and its width is the Kondo energy. Bottom panels: a few horizontal cuts of the same data. (Adapted from [10])*

5.2 Main results in the stationary limit

We are now done with the technique itself, it is time to see what it can do in practice and discuss a little the physics of the SIAM. We cannot do justice to the SIAM here. The model is very well understood at equilibrium. It is one of the very few correlated models that is both non-trivial (among other things it features the Kondo effect, hence an emerging energy scale) and very well understood (through Bethe ansatz, the numerical renormalization group, and more). Calculating its properties out-of-equilibrium though is harder and results in this direction are more recent. On the other hand almost all experiments measure current-voltage characteristics, hence take place out-of-equilibrium (see the introduction of [10] for references). So for a long time we were in a semi-comfortable situation where we understood the underlying physics but could not actually calculate the observables that were measured. The present technique contributed to bridge this gap. Fig. 9 shows a snapshot of the results of the calculation featuring both the "Coulomb diamonds" (upper panels) and the "Kondo ridge" (middle panels), two features that have been observed over and over experimentally. Actually, calculating the Coulomb diamond does not require such a sophisticated technique, it can be perfectly well understood at the semi-classical level. Yet, it is nice to obtain the entire colormap observed experimentally with a single technique and in a controlled way.

Let us finish by pushing the technique to its (current) limits which is to calculate $I(V_b)$ in the middle of the Kondo ridge at $U = 12$. We are going to span three orders of magnitude in bias

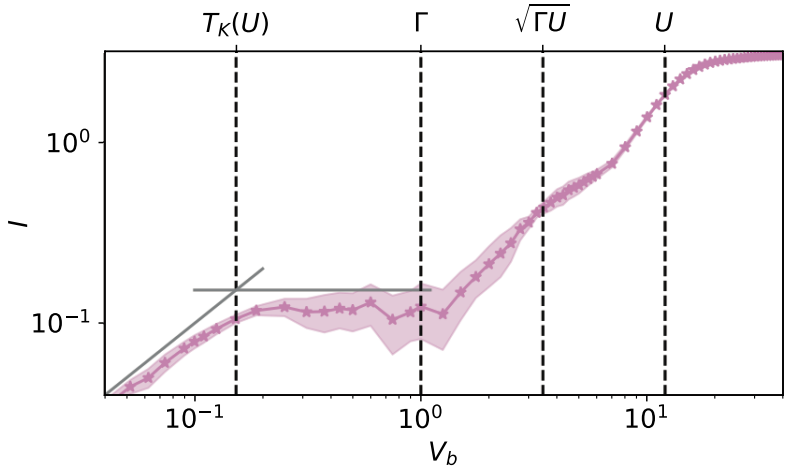


Fig. 10: *Current I as a function of V_b at $U = 12$. The dashed lines indicate the position of the energy scales discussed in the text. The grey lines correspond to perfect transmission $I = V_b$ and a plateau at $I = T_K(U)$. $N = 23$ coefficients were used. The Kondo temperature was extracted from Bethe Ansatz. (Adapted from [10])*

voltage and cross four different energy scales:

- The largest one is the charging energy U. For $V_b > U$ the current must saturate.
- The scale $\sqrt{\Gamma U}$ is associated to the fluctuations of the charge. For energies smaller than this scale, the charge can be considered as "frozen" and the problem reduces to its spin sector (i.e. essentially to the so-called Kondo model).
- The width Γ of the non-interacting resonance.
- The Kondo temperature T_K. This scale is what makes the whole problem interesting. T_K decreases exponentially with U. For $V_b < T_K$ one expects perfect transmission $I = V_b$ (in units where $e^2/h = 1$).

The results are shown in Fig. 10. We find the expected regimes at small and large bias voltages. For $T_K < V_b < \Gamma$, we observe a sort of plateau, that still needs to be confirmed given that the error bars are fairly high in this regime.

5.3 What is to take away?

I would like to end with a few comments as to where this is going. We have seen a technique that consists of several subtechniques (non-interacting Green functions, Wick determinants, tensor cross interpolation, series reconstruction) which put together allow to solve a quite challenging problem. None of these techniques are very difficult once someone has taken the trouble to write a proper open source code that properly isolates the corresponding functionality. This is perhaps my first comment: open source code and proper API are key to being able to assemble complex solutions. We need to be able to program at a level of abstraction that is close to the one we use to think about the problem. My second comment refers to TCI. Finding an underlying structure of the problem to be able to solve it has always be central to how physicists work.

What I find amazing with TCI is that this algorithm is able to discover this structure for us. This is generally true of learning algorithms including with deep neural networks and it is a major paradigm shift in the way we think about algorithms. My last comment is about the overall idea of teaching the computer to calculate Feynman diagrams. At the time of this writing it is not clear how far we will be able to go in this direction but I am quite awed by the number of radical speedups and paradigm shifts that we have seen since our first paper on this subject ten years ago. More generally, it seems that the field of computational many-body physics as a whole is moving very quickly these days and I would not be surprised if some of our central problems (say calculating the properties of the 2D Hubbard model) would soon switch from "about to be solved" to "solved actually".

References

[1] N. Prokof'ev, B. Svistunov, and I. Tupitsyn, Pis'ma v Zh.Eks. Teor. Fiz. **64**, 853 (1996) [English translation: cond-mat/9612091]

[2] K. Van Houcke, E. Kozik, N. Prokof'ev, and B. Svistunov: In D. Landau, S. Lewis, and H. Schuttler (Eds.): *Computer Simulation Studies in Condensed Matter Physics XXI* (Springer, Heidelberg, 2008)

[3] P. Werner, T. Oka, and A.J. Millis, Phys. Rev. B **79**, 035320 (2009)

[4] R.E.V. Profumo, C. Groth, L. Messio, O. Parcollet, and X. Waintal, Phys. Rev. B **91**, 245154 (2015)

[5] C. Bertrand, O. Parcollet, A. Maillard, and X. Waintal, Phys. Rev. B **100**, 125129 (2019)

[6] C. Bertrand, O. Parcollet, A. Maillard, and X. Waintal, Phys. Rev. B **100**, 125129 (2019)

[7] M. Maček, P.T. Dumitrescu, C. Bertrand, B. Triggs, O. Parcollet, and X. Waintal, Phys. Rev. Lett. **125**, 047702 (2020)

[8] Y. Núñez Fernández, M. Jeannin, P.T. Dumitrescu, T. Kloss, J. Kaye, O. Parcollet, and X. Waintal, Phys. Rev. X **12** (2022)

[9] M. Jeannin, Y. Núñez Fernández, T. Kloss, O. Parcollet, and X. Waintal, Phys. Rev. B **110**, 035124 (2024)

[10] M. Jeannin, Y. Núñez-Fernández, T. Kloss, O. Parcollet, and X. Waintal: *A comprehensive study of out-of-equilibrium Kondo effect and Coulomb blockade*, arXiv:2502.16306

[11] R. Rossi, Phys. Rev. Lett. **119**, 045701 (2017)

[12] Y. Meir and N.S. Wingreen, Phys. Rev. Lett. **68**, 2512 (1992)

[13] C.W. Groth, M. Wimmer, A.R. Akhmerov, and X. Waintal, New J. Phys. **16**, 063065 (2014)

[14] T. Kloss, J. Weston, B. Gaury, B. Rossignol, C. Groth, and X. Waintal, New J. Phys. **23**, 023025 (2021)

[15] Y.N. Fernández, M.K. Ritter, M. Jeannin, J.-W. Li, T. Kloss, T. Louvet, S. Terasaki, O. Parcollet, J. von Delft, H. Shinaoka, and X. Waintal, SciPost Phys. **18**, 104 (2025)

[16] J. Rammer and H. Smith, Rev. Mod. Phys. **58**, 323 (1986)

[17] X. Waintal, M. Wimmer, A. Akhmerov, C. Groth, B.K. Nikolic, M. Istas, T. Örn Rosdahl, and D. Varjas: *Computational quantum transport* arXiv:2407.16257

[18] B. Gaury, J. Weston, M. Santin, M. Houzet, C. Groth, and X. Waintal, Phys. Rep. **534**, 1 (2014)

[19] U. Schollwöck, Ann. Phys. **326**, 96 (2011)

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