Tensor models II

Subject: Graphs, tensor models, combinatorics, recursion, backtracking, symbolic evaluation.

Introduction

In a previous problem we discussed the construction of interaction vertices for rank-r fermionic tensor models. Once a particular model has been chosen, one may investigate its correlation functions, free energy and other quantities of interest as usual, *i.e.* by doing perturbation theory and expanding them in Feynman diagrams.

For example, the expansion of the free energy involves the computation of connected Feynman diagrams with no open legs or self-contractions, whose number of vertices is related to the order in perturbation theory we are computing. In a rank-r model with a single MST complete interaction bubble of $n = r + 1$ fields, these diagrams are connected n-regular graphs.[1]

While the number of graphs involved in the perturbative expansions of generic models grows exponentially, making explicit computations prohibitively hard, in some cases one may define a suitable large-N limit in which only particular classes of Feynman diagrams dominate. These can sometimes be resummed in exact Schwinger-Dyson equations, which then lead to the non-perturbative solution of the model under study.

To explore the possibility of defining a suitable large-N limit, we can compute the large-N scaling of some Feynman diagrams at low orders. This is done in a manner entirely similar to how large-N power-counting is performed for matrix models using double-line diagrams *à la* 't Hooft. Indeed,

[1] An n-regular graph is a graph where every node has n edges connected to it.

all we have to do is count loops in the *ribbon graphs* associated to the Feynman diagrams in question. The ribbon graphs themselves are constructed from Feynman diagrams by transforming each propagator into a multi-line propagator with r colored strands, and each vertex into a "fat" vertex having n open legs, each also with r strands; see Fig. 1.

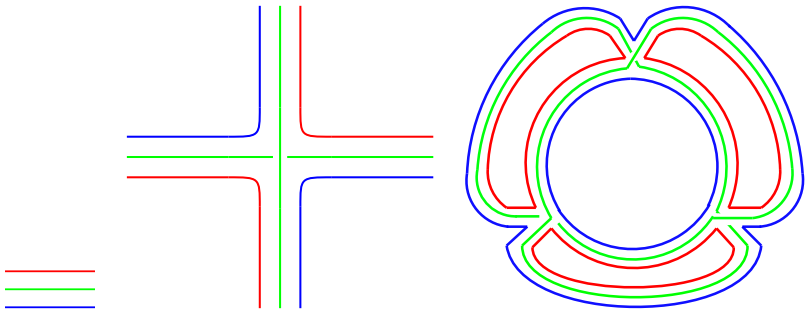

Figure 1: Ribbon graph representation of Feynman diagrams in a rank-3 tensor model: (a) the propagator; (b) the tetrahedral vertex (1); (c) an order-3 diagram scaling as $g^3(N^6 + 3N^5 + 3N^4 + 2N^3) \sim \mathcal{O}(g^3 N^6)$.

The whole process therefore typically involves three steps:

1. We construct all Feynman diagrams of interest up to some maximal order K.

2. We compute the large-N scaling for each of the diagrams produced in step (1).

3. We identify in the results of step (2) a power α such that we can take the limits $g \to 0, N \to \infty$ with $\lambda = gN^\alpha$ fixed, and only some diagrams dominate in the λ expansion.[2]

If we can characterize in general the particular sub-class of Feynmann diagrams that dominate in the large-N limit thus defined, we might hope to resum the expansion at leading order in λ to obtain non-perturbative results in the coupling constant g.

[2]We assume for simplicity the theory has a single interaction vertex, the coupling constant appearing in the action in front of it being g.

In this problem, we will proceed with step (1) above in general but then restrict our attention to the rank-3 model with the tetrahedral interaction of [1],

$$g\,\psi_{i_1 j_1 k_1}\psi_{i_1 j_2 k_2}\psi_{i_2 j_1 k_2}\psi_{i_2 j_2 k_1}\,. \tag{1}$$

Problem statement

Generating all non-isomorphic n-regular connected graphs

In order to study the free energy of fermionic rank-r tensor models with a single MST complete interaction bubble, we should be able to write down all the relevant Feynman diagrams up to some given maximal order K in perturbation theory. As explained earlier, these are n-regular graphs of k vertices, where $n = r + 1$ is the number of fields in the interaction term and $k = 1, \ldots, K$ is the corresponding power of the coupling constant g. The diagrams should be connected and have no self-contractions, and moreover since all the vertices correspond to the same interaction we are only interested in the non-isomorphic graphs up to node-relabelling.

Write down a recursive function `GenerateGraphs` to generate all of these graphs. It should work essentially like in the previous problem (alternating generation and filtering) but the various implementation details are left as an excercise. You should find for example that the number of 4-regular graphs with $k = 2, 3, 4, 5, 6, 7, \ldots$ vertices is $1, 1, 3, 6, 19, 50, \ldots$. Some examples of these are shown in Fig. 2.

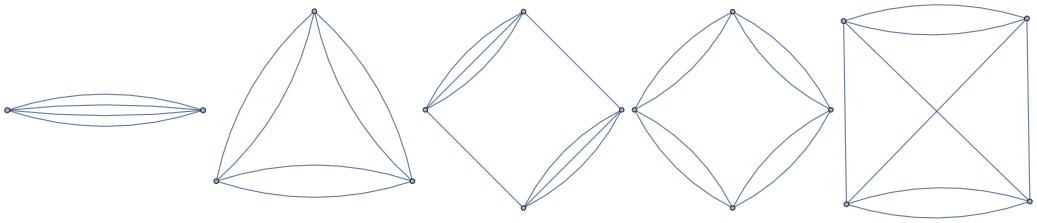

Figure 2: All connected 4-regular graphs with no self-contractions and $k = 2, 3, 4$ vertices.

Comparing with the numbers in http://oeis.org/A129417, you can gauge the efficiency of your code against state-of-the-art implementations such as

[2]. This should make it clear that writing efficient code is no more important than figuring out whether the problem you want to solve has already been addressed by someone more capable. While for pedagogical purposes we will use the `GenerateGraphs` function you have implemented in what follows, in a real-world scenario one would simply interface the implementation of [2] with Mathematica using some standard graph-description file format.

Large-N power-counting

In order to automate the large-N power-counting of ribbon graphs we will represent each of its strands by a Kronecker delta, so that for example the propagator in $r = 3$ models is

$$\langle \psi_{i_1 j_1 k_1} \psi_{i_2 j_2 k_2} \rangle_0 = \delta_{i_1 i_2} \delta_{j_1 j_2} \delta_{k_1 k_2} \,. \tag{2}$$

In turn, the tetrahedral vertex (1) of Fig. 1b is

$$\delta_{i_1 i_2} \delta_{i_3 i_4} \delta_{j_1 j_3} \delta_{j_2 j_4} \delta_{k_1 k_4} \delta_{k_2 k_3} \times g \, \psi_{i_1 j_1 k_1} \psi_{i_2 j_2 k_2} \psi_{i_3 j_3 k_3} \psi_{i_4 j_4 k_4} \,, \tag{3}$$

where of course only the Kronecker deltas are actually relevant.

Define a function that transforms a graph into a product of propagators and vertices in this representation, taking care of properly managing the index labels. Note that while the nodes are indistinguishable, in the perturbative expansion of the free energy the contractions between vertex legs are done in all possible orders, so in practice one should sum over all permutations of the vertex legs.

Once you have implemented the transformation described above, use the `TagSetDelayed` (`/:`) feature of `Mathematica` to evaluate these expressions symbolically by writing simple contraction rules for the Kronecker deltas. Then simply expanding the δ-product expresions you generate should result in their evaluation as polynomials in N.

Identifying melonic diagrams

The evaluation performed in the previous subsection should allow you to extract the maximal power of N appearing in any graph at a given order g^k in perturbation theory. Since ψ_{ijk} has $\mathcal{O}(N^3)$ degrees of freedom, the free energy scales as N^3 times some polynomial in g and N. Evaluating all graphs up to order $K = 8$, it shoud be possible to find the power α such that

keeping $\lambda = gN^\alpha$ fixed the perturbative expansion becomes a λ-power series with each term having an $N^{-1/2}$ expansion.

Once you have identified the proper scaling of the Feynman diagrams dominating the expansion of the free energy, you can provide evidence that the theory is melonic by showing that all the dominating diagrams up to some low order are indeed melonic. Melonic diagrams are constructed from the basic melon of the left of Fig. 3 by repeatedly opening a propagator in two melonic diagrams and joining them.

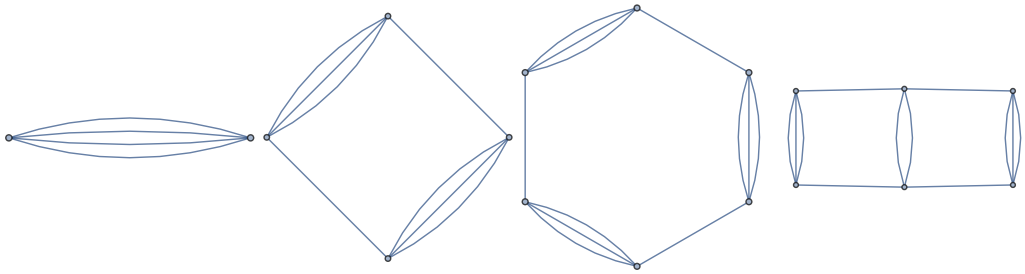

Figure 3: All melonic diagrams of order $k \leq 6$.

Write a function `MelonicDiagramQ` to check whether a graph is melonic or not by recursively undoing the melonic move described in the previous paragraph, and show that indeed the theory we have studied is melonic at least up to order $K = 8$.

The symmetric and antisymmetric $O(N)$ models (optional)

The procedure described above is quite general, and can be applied to other models as well. For example, we could consider the symmetric or antisymmetric $O(N)$ models, where the indices of the field $\psi_{i_1 i_2 \dots i_r}$ are symmetrized or antisymmetrized, respectively. To do this we only have to change the propagator accordingly, so that for example for $r = 3$ we have

$$\begin{aligned}\langle \psi_{i_1 j_1 k_1} \psi_{i_2 j_2 k_2} \rangle_0^{(\mathrm{s})} = \tfrac{1}{6} \, (&\delta_{i_1 i_2}\delta_{j_1 j_2}\delta_{k_1 k_2} + \delta_{i_1 j_2}\delta_{j_1 i_2}\delta_{k_1 k_2} + \delta_{i_1 k_2}\delta_{j_1 j_2}\delta_{k_1 i_2} \\ &+\delta_{i_1 i_2}\delta_{j_1 k_2}\delta_{k_1 j_2} + \delta_{i_1 j_2}\delta_{j_1 k_2}\delta_{k_1 i_2} + \delta_{i_1 k_2}\delta_{j_1 i_2}\delta_{k_1 j_2}) \ , \end{aligned} \tag{4}$$

$$\begin{aligned}\langle \psi_{i_1 j_1 k_1} \psi_{i_2 j_2 k_2} \rangle_0^{(\mathrm{a})} = \tfrac{1}{6} \, (&\delta_{i_1 i_2}\delta_{j_1 j_2}\delta_{k_1 k_2} - \delta_{i_1 j_2}\delta_{j_1 i_2}\delta_{k_1 k_2} - \delta_{i_1 k_2}\delta_{j_1 j_2}\delta_{k_1 i_2} \\ &-\delta_{i_1 i_2}\delta_{j_1 k_2}\delta_{k_1 j_2} + \delta_{i_1 j_2}\delta_{j_1 k_2}\delta_{k_1 i_2} + \delta_{i_1 k_2}\delta_{j_1 i_2}\delta_{k_1 j_2}) \ . \end{aligned} \tag{5}$$

(6)

Evaluation of the large-N scaling of these models proceeds in an analogous fashion, except for the fact that we do not need to consider all permutations of the vertex legs because the propagators already implement the (anti)symmetrization. Try to proceed as before, and analyze whether these two theories are melonic or not.

Details

While the `TagSetDelayed` functionality in `Mathematica` is extremely useful, one should remember that it heavily relies in pattern matching. Therefore, it can lead to reduced performance if many rules are associated to symbols ocurring very frequently. In order to mitigate this limitation, it is often convenient to arbitrarily distinguish otherwise identical entities, which forces `Mathematica` into considering only reduced sets of rules at each step. In the problem we are considering, one may for example distinguish Kronecker deltas coming from vertices and propagators, say $\delta^{(v)}$ and $\delta^{(p)}$, and a mixed version $\delta^{(pv)}$. Then because $\delta^{(v)}$ and $\delta^{(p)}$ never self-contract we only need to include rules such as $\delta^{(v)}\delta^{(p)} \mapsto \delta^{(pv)}$, $\delta^{(pv)}\delta^{(pv)} \mapsto \delta^{(pv)}$ and so on. While it may seem counterintuitive because we have more rules than originally, this approach does pay off in terms of overall efficiency.

Another point to have in mind is that the order in which `Mathematica` expands an expression heavily affects the number of terms in intermediate steps of the computation. Therefore, instead of using `Expand` one could write a custom function that uses the diagram's topology to optimize the δ-contraction process. For example, it is generally convenient to contract at once all propagators connecting two vertices, or break the graph in two pieces to expand separately by cutting the minimal possible number of propagators. These optimizations are not required for this problem, but can be used to go to orders in perturbation theory beyond those studied in [1].

Useful functions

You may find it useful to read the `Mathematica` help pages of the following functions:

- **Graphs:** `Graph`, `ConnectedGraphQ`, `IsomorphicGraphQ`, `VertexAdd`, `EdgeAdd`, `VertexDegree`, ...

- **Others:** `Exponent`, `DeleteDuplicates`, `DeleteDuplicatesBy`, `Subsets`, ...

References

[1] I. R. Klebanov and G. Tarnopolsky, JHEP **1710** (2017) 037 doi:10.1007/JHEP10(2017)037 [arXiv:1706.00839 [hep-th]].

[2] B.D. McKay and A. Piperno, "Practical Graph Isomorphism, II," Journal of Symbolic Computation, **60** (2014), pp. 94-112 doi:10.1016/j.jsc.2013.09.003 [arXiv:1301.1493 [cs-dm]].