HAL open science

Numerical methods for EPRL spin foam transition amplitudes and Lorentzian recoupling theory

Pietro Dona, Giorgio Sarno

To cite this version:

Pietro Dona, Giorgio Sarno. Numerical methods for EPRL spin foam transition amplitudes and Lorentzian recoupling theory. 2024. hal-01846723

HAL Id: hal-01846723

https://hal.science/hal-01846723v1

Preprint submitted on 25 Feb 2024

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.

arXiv:1807.03066v3 [gr-qc] 18 Sep 2018

Numerical methods for EPRL spin foam transition amplitudes and Lorentzian recoupling theory

Pietro Donà[a*], Giorgio Sarno[b †]

[a] Institute for Gravitation and the Cosmos & Physics Department,
Penn State, University Park, PA 16802, USA

[b] Aix Marseille Univ, Université de Toulon, CNRS, CPT, Marseille, France

September 20, 2018

Abstract

The intricated combinatorial structure and the non-compactness of the Lorentz group have always made the computation of $SL(2,\mathbb{C})$ EPRL spin foam transition amplitudes a very hard and resource demanding task. With `sl2cfoam` we provide a C-coded library for the evaluation of the Lorentzian EPRL vertex amplitude. We provide a tool to compute the Lorentzian EPRL 4-simplex vertex amplitude in the intertwiner basis and some utilities to evaluate SU(2) invariants, booster functions and $SL(2,\mathbb{C})$ Clebsch-Gordan coefficients. We discuss the data storage, parallelizations, time, and memory performances and possible future developments.

Contents

1 Introduction **1**

2 The EPRL spin foam transition amplitude **3**

3 The `sl2cfoam` library **5**
3.1 General strategy . . . 5
3.2 SU(2) invariants . . . 6
3.3 Booster functions . . . 9
3.4 Four simplex amplitude . . . 12

4 Utilities **15**
4.1 Coherent states . . . 15
4.2 $SL(2,\mathbb{C})$ symbols . . . 15

5 What comes next? **16**

A SU(2) Symbols **18**

1 Introduction

The spin foam framework provides an approach to the dynamics of loop quantum gravity that is both background independent and Lorentz covariant. The most successful model is the one proposed by Engle, Pereira, Rovelli and Livine (EPRL) [1, 2, 3] and independently by Freidel and Krasnov (-FK) [4]. The model

*pxd81@psu.edu
†sarno@cpt.univ-mrs.fr

is formulated as a sum over simplicial triangulations with given boundary providing transition amplitudes for (projected) spin network states of the canonical theory restricted to four valent nodes. This procedure is equivalent to a sum over histories of quantum geometries providing in this way a regularised version of the quantum gravity path integral.

A generalization to arbitrary complexes and spin-networks has been developed [5]. Nevertheless, in this work, we focus primarily on the simplicial formulation, and we provide all the tools for more general computations.

Computations in this framework are particularly complex. Two of the most successful analytical calculations are: On the one hand, the presence of the discrete Regge action in the spin foam amplitude of a semiclassical state [6, 7]. Regge calculus provides a discretization of general relativity based on a triangulation. [1] On the other hand, the graviton n-point function reproduces at the leading order the one of Regge calculus in the semi-classical limit [8, 9, 10, 11]. This result is of particular historical importance since it revealed the shortcomings of the Barrett-Crane model and led to the development of the EPRL-FK model. In particular, it has been shown that within the Barrett-Crane model it was not possible to reproduce the correct long-distance scaling of the graviton two-point function [12, 13, 14]. Therefore, the model is unable to implement the simplicity constraints correctly. These are responsible for reducing the topological BF theory, the starting point of any spin foam model, to gravity.

Evaluating spin foam transition amplitudes for assigned boundary data is an arduous task. All the extraordinary results we mentioned are asymptotic computations performed in the large spin limit. At the present time, numerical calculations within the theory are very rare. Spin correlations have been computed with the old Barrett-Crane model [15, 16] or with the Lorentzian EPRL model but restricted to simple generalized spin foams [17].

In this work, we introduce the library `sl2cfoam` where we provide the tools to evaluate the EPRL spin foam vertex amplitude numerically. It is based on the decomposition of the EPRL vertex amplitude in terms of booster functions introduced in [18]. These utilities are indicated for the study of vertex amplitudes with coherent boundary data and non, opening new windows on the covariant formulation of loop quantum gravity.

Some preliminary results have already been achieved. Using a primitive version of this code, it was possible to verify, for the first time, the asymptotic behavior of the topological BF SU(2) spin foam amplitude [19], but also to find the semiclassical region at relatively small values of the spins. The transition amplitudes and spin correlations between two dipole spin networks, in the Lorentzian EPRL model, with up to two (non-simplicial) vertices has been studied numerically in [17]. Even if restricted to simple vertices this was the first full evaluation of an EPRL transition amplitude. The numerical computation of the non-isotropic scaling of the booster functions, performed with a piece of this software, is a central ingredient in the recent estimation of the degree of divergence of the EPRL model [20].

Moreover, we are finishing the numerical exploration of the Euclidean sector of the asymptotic of the Lorentzian EPRL vertex amplitude, reproducing with great accuracy the analytic asymptotic formula [21].

With the access to a reliable and systematic way to perform numerical computation within the Lorentzian spin foam framework, in particular with more than a single vertex (or equivalently with many simplices), we are opening a window on a plethora of open questions. One question we plan to address soon is the so-called *flatness problem*, firstly mentioned by Freidel and Conrady [22], later by Bonzom [23] and Hellmann and Kaminski [24]. They argue that the classical geometries reproduced by the EPRL model in the semiclassical limit are restricted to be flat and, as a result, the model itself is put in question, since GR in four dimensions admits curved spacetime solutions. A preliminary answer, in the Euclidean setting, has been given [25, 26], showing that curved geometric configurations are asymptotically allowed. We think that a numerical exploration is viable in Lorentzian setting and could help us shedding light on this issue.

The realm of applications of `sl2cfoam` is not limited to technical results. The computation of physical observables requires the ability to evaluate spin foam transition amplitude with multiple vertices. The study of the black hole to white hole tunneling processes [27, 28] and quantum cosmological models [29, 30, 17] is the beginning of a long term project that aims to connect the quantum theory of gravity with observation. We hope that the release of the tools in `sl2cfoam` will help these computations to flourish.

The purpose of the present work goes beyond giving the tools for the computation of Lorentzian EPRL

[1] The variables of Regge calculus are the simplex lengths (encoding the intrinsic curvature). The extrinsic curvature is encoded in the dihedral angles hinging on the triangles of the triangulation.

transition amplitudes. In particular, we believe that the techniques we introduced and the algorithms we implemented can be used by the loop quantum gravity community on problems outside the spin foam context. For example, the numerical evaluation of $SL(2,\mathbb{C})$ Clebsch Gordan coefficients is a challenging and interesting mathematical problem on its own and the very efficient way we introduced to deal with SU(2) invariant can play a key role in exploring numerically the action of the hamiltonian operator in the canonical approach [31].

We assume that the reader is familiar with the EPRL model, and refer to the original literature [1, 2, 3, 4] and existing reviews (e.g. [32, 33]) for motivations, details and its relation to loop quantum gravity. We give a short review of the transition amplitude in Section 2.

In Section 3 we present in details how the library `sl2cfoam` is structured. In Section 3.1 we give a general overview of the problems we faced and the solutions we propose, while in Sections 3.2, 3.3 and 3.4 we discuss its three main components, respectively the evaluation of SU(2) invariants, booster functions and finally the complete vertex amplitude.

In the last two sections we present two additional utilities that computes the Livine-Speziale coherent states in Section 4.1 and the $SL(2,\mathbb{C})$ Clebsch-Gordan coefficients in Section 4.2.

All the tests shown in this paper have been performed on Intel(R) Xeon(R) E5-2687W v2 @ 3.40GHz CPUs provided by CPT servers in Marseille. If not stated otherwise the tests are performed on a single core.

The library `sl2cfoam` can be found at: `https://bitbucket.org/giorgiosarno/sl2cfoam-1.0/`. We provide a `README` file in which we explain how to use the code. We include the test programs we used to perform the benchmarks in this paper and can be seen as examples. The computation of a vertex amplitude for given data has to be computed only once. We include a compressed table of pre-calculated symbols, building blocks, and full amplitudes that we already computed in the repository. We will continuously update it in the future.

2 The EPRL spin foam transition amplitude

The spin foam formalism introduces a partition function of a given closed 2-complex $\mathcal{C}$ as a state sum over $SU(2)$ spins j_f on the faces and intertwiners i_e on the edges:

$$Z_{\mathcal{C}} = \sum_{j_f, i_e} \prod_f A_f(j_f) \prod_e A_e(i_e) \prod_v A_v\left(j_f,\ i_e\right) \ . \tag{1}$$

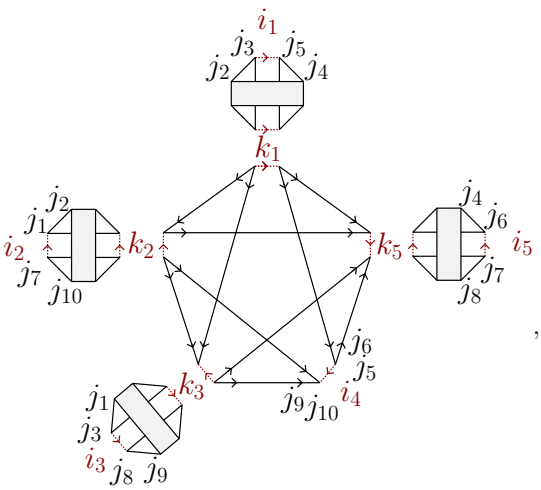
defined in terms of the face amplitude A_f, and the edge amplitude A_e and the vertex amplitude A_v. Requiring the correct convolution property of the path integral at fixed boundary, the form of the face amplitude $A_f(j_f) := 2j_f + 1$ and the edge amplitude $A_e(i_e) = 2i_e + 1$ are fixed [34]. In this paper, we will focus on the numerical evaluation of the vertex amplitude A_v defined by the EPRL model.

The vertex amplitude is built from the topological $SL(2,\mathbb{C})$ BF spin foam vertex amplitude by imposing, weakly, the simplicity constraints resulting into a restriction of the unitary irreducible representations in the principal series [1, 35]. To evaluate it in its original form one should perform four group integrals (one of the original five is redundant and has to be removed to guaratee finiteness [36]). Each group integral is, in general, a six dimensional unbounded highly oscillating integral for which numerical integration methods are very inefficient. To solve this issue, a more computational friendly form for the amplitude, has been

proposed [18]:

$$A_v\left(j_f,\, i_e\right) = \sum_{l_f,k_e} \left(\prod_e \left(2k_e+1\right) B_4(j_f, l_f; i_e, k_e) \right) \{15j\}_{j_f, i_4}(l_f, k_e) \tag{2}$$

$=\sum_{l_f,k_e}$



,

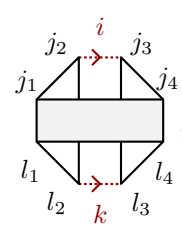
resulting in a superposition of $SU(2)$ $\{15j\}$ symbols weighted by one booster functions B_4 per edge e in the considered vertex. To obtain this form each $SL(2,\mathbb{C})$ integral is decomposed in a SU(2) "rotation", a boost along a fiducial axes and another SU(2) "rotation". The compact integrals are then evaluated exactly composing the SU(2) invatiants in (2).

The summation is over a set of auxiliary spins l_f[2] for each face involving the vertex but excluding the gauge fixed edge for a total of 6 distinct l_f, with lower bound $l_f \geq j_f$, and a set of auxiliary intertwiners k_e for each edge in the vertex excluding the gauge fixed one for a total of 4 that can assume all the values compatible with triangular inequalities.

The expression (2) is far more convenient than the original one for numerical evaluation. We decomposed the amplitude into smaller building blocks, which are easier to compute. We trade four six dimensional unbounded integrals with a summation of SU(2) invariants, which are widely studied, and a class of one-dimensional integrals that we need to perform and we can characterize independently.

These formulas can be generalized to arbirtary valency, for more details see [17]. In the library `sl2cfoam` , in addition to the simplicial amplitude, we also provide the tools to evaluate amplitudes of four valent vertices. The reader can be unfamiliar with the B_4 functions, these are the non compact remants of the $SL(2,\mathbb{C})$ integrals and they encode all the details of the EPRL model. We named them booster functions and are defined in the following way:

$$B_4(j_f, l_f; i, k) = \frac{1}{4\pi} \sum_{p_a} \begin{pmatrix} j_f \\ p_f \end{pmatrix}^{(i)} \left(\int_0^\infty \mathrm{d}r \sinh^2 r \prod_{a=1}^{n} d^{(\gamma j_f, j_f)}_{j_f l_f p_f}(r) \right) \begin{pmatrix} l_f \\ p_f \end{pmatrix}^{(k)} = \tag{3}$$



,

where $d^{(\rho,k)}(r)$ are the boost matrix elements for γ-simple irreducible representations of $SL(2,\mathbb{C})$ in the principal series, γ is the Immirzi parameter and the $(4jm)$ symbols are reported in Appendix A. All the test and computations in this paper are performed with a conventional value of $\gamma = 1.2$. The explicit form of the boost matrix elements can be found in the literature in its general form [37, 38, 39, 18], here we just report

[2] that are effectively magnetic indices respect the group $SL(2,\mathbb{C})$ and are originated by the splitting of two representation matrices

them in the case of simple irreducible representation:

$$d^{(\gamma j,j)}_{jlp}(r) =(-1)^{\frac{j-l}{2}} \frac{\Gamma\left(j+i\gamma j+1\right)}{|\Gamma\left(j+i\gamma j+1\right)|} \frac{\Gamma\left(l-i\gamma j+1\right)}{|\Gamma\left(l-i\gamma j+1\right)|} \frac{\sqrt{2j+1}\sqrt{2l+1}}{(j+l+1)!} \left[(2j)!(l+j)!(l-j)!\frac{(l+p)!(l-p)!}{(j+p)!(j-p)!}\right]^{1/2}$$
$$\times e^{-(j-i\gamma j+p+1)r} \sum_s \frac{(-1)^s\, e^{-2sr}}{s!(l-j-s)!}\, {}_2F_1[l+1-i\gamma j, j+p+1+s, j+l+2, 1-e^{-2r}]\ . \tag{4}$$

The study of the booster function is still in the initial stage, more work is necessary to fully understand all their properties. For example, while the symmetries are known, the recursion relations still need to be explored. The study of their semiclassical limit is in a preliminary stage [40] but an appealing geometrical picture seems to emerge.

On one hand, the introduction of booster functions brings a drastic simplification to the computation of the vertex amplitude because it reduces the problem of dealing with many high oscillatory integrals to the study a family of one dimensional integrals, which are easier to handle and manipulate. On the other hand, we introduce a large amount of convergent summation in the computation which need to be dealt with. The net gain of this operation is evident once we realize that each element of the sum can be computed in a fast way and with high precision.

3 The `sl2cfoam` library

3.1 General strategy

We can summarize the evaluation of the vertex amplitude (2) in three main ingredients that we need to mix in the right way:

- all the necessary $\{15j\}$ symbols,
- all the required booster functions,
- the sum over the auxiliary spins and intertwiners.

In the next sections we will discuss in great detail the problems we encounter in these three steps and their possible solutions. Here we want to give a general overview. The first valuable lesson we learned is to not waste time computing vanishing contributions. Many SU(2) invariants and booster functions are zero because of symmetries (triangular inequalities, odd symmetries under permutations), we avoid computing objects that are part of amplitudes that we can identify as vanishing a priori.

Symmetries also play a secondary but equally important role. The objects in the summation repeat themselves, instead of wasting time doing the same computation over and over again, we store them in the RAM and, at the end, we dump them to disk. To make it possible and not saturate the memory in a couple of iterations, we optimize the computation factoring the symmetries and consider only one representative per equivalence class. The bottleneck of the computation while dealing with SU(2) invariants is not the evaluation time but the memory needed. In an equivalence class of $\{6j\}$ symbols there are approximately a hundred elements, we store only one representative per class reducing by two orders of magnitude the amount of necessary memory.

When dealing with summations with a huge amount of terms we need to be extremely careful. In our calculation, apart from the summation over auxiliary spins and intertwiners itself, these sums appear also in the evaluation of the booster functions. In particular, to deal effectively with interference effects we need to resort to arbitrary precision libraries [41, 42, 43]. How to sum a large number of finite precision floating point numbers is a well-studied problem in numerical analysis. The bound of the worst-case error on the sum of N numbers with floating point precision grows proportional to N. By using a compensated summation algorithm [44] we can make this error independent of N, and having it depend only on the floating-point precision of the addends. Moreover, we employ compensated summation algorithms to significantly reduce the numerical error in the addition of finite precision floating point numbers.

The full computation is divided into a large number of small tasks and we can efficiently parallelize it with `OpenMP`. We parallelize the computation of the booster functions and the summation over the auxiliary

spins but not the evaluation of the SU(2) invariants. The computational time needed for the evaluation of the SU(2) invariants is negligible respect to all the other steps.

The sums over the six auxiliary spins are unbounded and convergent, therefore to extract a number from them we need to truncate. We introduce a homogeneous cutoff on all the spins $j_f \leq l_f \leq j_f + \Delta$ and we sum over all compatible intertwiners k_e. The details of the convergence of this summation strongly depend on the value of the face spins and edge intertwiners, hindering our ability to provide a consistent estimate of the systematical error due to the truncation. We are then forced to perform an analysis a posteriori: we report some examples in Figure 8 where it is possible to estimate it between 1 and 5 percent for $\Delta \approx 5$.

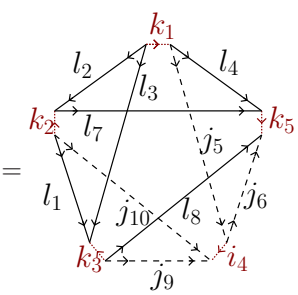
We mention here that we studied the possibility to develop an algorithm to guess an appropriate value for the cutoff as a function of the desired precision. However, we decided not to implement it in the current version of the library because we are lawing a way to deal with non monotonic convergences.

The error on the booster functions, due to the discretization of the integral, and the one on the SU(2) invariants are completely irrelevant with respect to the one due to the truncation. We will discuss them in more detail in the following sections.

For a single 4-simplex with fixed intertwiners, the bottleneck of the computation is the evaluation of the booster functions. If we need to compute the amplitude for all the possible intertwiners, for example, because we are contracting more than a single vertex, most of the computational time is used in performing the summation. Having in mind the goal of performing the computation of spin foam diagrams with many vertices, since the number of elements grows exponentially with it, we are studying selection rules based on the asymptotic of the booster functions or with the machine learning techniques.

3.2 SU(2) invariants

Evaluating a significant amount of SU(2) invariants can be a resources draining task. An SU(2) invariant is defined as a summation over the magnetic indices of intertwiners (and them as a sum of $(3jm)$ symbols). Their evaluation, directly from the definition, is not convenient, as it will exhaust too many computational resources, both in time and memory. To compute an amplitude we need to evaluate a large number of invariant, requiring a technique that is both fast and memory lightweight. We can solve the time issue by reducing to one the number of summation. We can reduce every SU(2) invariant in a sum of $\{6j\}$ and $\{9j\}$ symbols that can be directly computed using prebuilt libraries [46]. We tackled this problem for the first time in [19] where the asymptotic of a general SU(2) invariant has been studied analytically and also numerically in the simplicial case. We employ a similar technique in `sl2cfoam` . Following the notation of the monography [45] we identify five different kinds of $\{15j\}$ symbols. For example, we can write the $\{15j\}$ symbols of the first kind as a sum of the product of five $\{6j\}$s symbols:

$\{15j\}_I :=$ 

$$= \sum_x (2x+1)(-1)^{\sum_i l_i + j_i + k_i + i_4} \begin{Bmatrix} k_1 & l_7 & x \\ k_5 & j_5 & l_4 \end{Bmatrix} \begin{Bmatrix} j_5 & k_5 & x \\ l_8 & i_4 & l_6 \end{Bmatrix} \tag{5}$$

$$\times \begin{Bmatrix} i_4 & l_8 & x \\ k_3 & j_{10} & j_9 \end{Bmatrix} \begin{Bmatrix} j_{10} & k_3 & x \\ l_3 & k_2 & l_1 \end{Bmatrix} \begin{Bmatrix} k_2 & l_3 & x \\ k_1 & l_7 & l_2 \end{Bmatrix}.$$

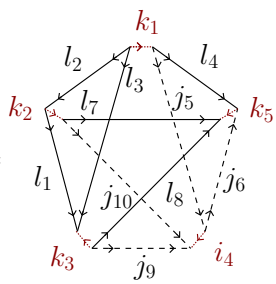
To compute a single amplitude at a fixed cutoff Δ, we need to evaluate an enormous amount of $\{15j\}$ symbols, and consequently of $\{6j\}$ (or $\{9j\}$) symbols. We recall that all the spins l_f vary in the range between the face spin j_f and the cutoff $j_f + \Delta$, while the intertwiners k_e assume all the values allowed by the triangular inequalities.

The development of algorithms that allow for a fast and accurate computation of $\{6j\}$ symbol is a fascinating and challenging topic studied by mathematicians and computer scientist, and a significant amount of libraries for the job are available. In this work, we adapt the C library `wigxjpf` developed in [46] that can

be used to compute $(3jm)$, $\{6j\}$ and $\{9j\}$ symbols. We refer to their original paper for a detailed discussion of the library's performances, accuracy and memory management.

We also solved the memory usage issue by efficiently storing $\{6j\}$ symbols taking into account the various symmetries of the symbols. The right data structure for this job is the hash table, a structure that efficiently maps keys to values. We associate to each $\{6j\}$ symbol a key that is automatically shared by the whole class of symbols with the same value (because of symmetries). Every $\{6j\}$ symbols share its value with other 143 [48] symbols due to permutation symmetries. Notice that an extension of the library `wigxjpf` named `fastwigxj` provides a hash table implementation for $(3jm)$, $\{6j\}$ and $\{9j\}$ symbols but, since it is designed for more general applications, it stores all possible combinations of the desired invariants with *all* spins smaller than a cutoff. This amount of data is way more of what we need in an ordinary EPRL four simplex computation resulting in a huge memory consumption, and consequently being an obstacle when exploring larger values of the spins. To circumvent this limitation, we implement our own hash tables based on `khash` provided in the library `Klib` [47]. We generate the keys of the hash table with the hash functions discussed in [48], and we store the $\{6j\}$ symbols assigning a unique six integers key to each symmetry class, saving one representative value.

We can introduce an additional simplification to improve the performance, as already used in [19]. We can use the freedom of choosing the intertwiners recoupling basis of the spin foam edges to rewrite the $\{15j\}$ symbols as:

$\{15j\}_R :=$ 

$$= (-1)^{2(l_4+l_3)+2k_1-(k_2+k_3+i_4+k_5)} \times \begin{Bmatrix} k_1 & k_3 & k_2 \\ l_1 & l_2 & l_3 \end{Bmatrix} \begin{Bmatrix} k_1 & i_4 & k_5 \\ j_6 & l_4 & j_5 \end{Bmatrix} \begin{Bmatrix} k_2 & k_3 & k_1 \\ j_{10} & j_9 & i_4 \\ l_7 & l_8 & k_4 \end{Bmatrix}, \tag{6}$$

and then rewrite the $\{9j\}$ symbol as a sum of three $\{6j\}$ symbols:

$$\{15j\}_R = (-1)^{2(l_4+l_3)+2k_1-(k_2+k_3+i_4+k_5)} \begin{Bmatrix} k_1 & k_3 & k_2 \\ l_1 & l_2 & l_3 \end{Bmatrix} \begin{Bmatrix} k_1 & i_4 & k_5 \\ j_6 & l_4 & j_5 \end{Bmatrix} \times \sum_x (2x+1)(-1)^{2x} \begin{Bmatrix} k_2 & j_{10} & l_7 \\ l_8 & k_5 & x \end{Bmatrix} \begin{Bmatrix} k_3 & j_9 & l_8 \\ j_{10} & x & i_4 \end{Bmatrix} \begin{Bmatrix} k_1 & i_4 & k_5 \\ x & k_2 & k_3 \end{Bmatrix}. \tag{7}$$

Comparing with (5) we are still using five $\{6j\}$ symbols but we are summing only over three of them, resulting in a relevant save of time and memory. This simplification is quite in general, (is free if the edges are not boundary ones), but we need to be careful in computations of spin foams transition amplitudes with many vertices, since the base choice on one edge affects both vertices it connects, and the split (7) needs to be done consistently in every vertex. For an explicit example of a five vertex transition amplitude where this splitting is performed see Section 4.4 of [20].

The function where we implemented the evaluation and the storing of the $SU(2)$ invariants is called `J15Symbol_Hash`. We sketch its working principles in the flowchart in Figure 1. The function prepares all the $\{6j\}$ symbols in (7) needed for the computation of a vertex amplitude given the 10 spins j_f $f = 1, \ldots, 10$ and a cutoff Δ. Following the conventions of (7) the gauge fixed edge is identified by the four spins j_5, j_6, j_9 and j_{10} in the recoupling basis (j_5, j_6) with intertwiner i_4. After the evaluation, the hash tables are stored in a file indexed with the name of the gauge fixed edge and the cutoff Δ, with the naming convention `2j5.2j6.2j9.2j10_2i4_2`Δ. The factor of two multiplying the spins deals with the possible half-integer values. We decided not to use all boundary variables in the indexing to avoid files proliferation when computing amplitudes with unbounded internal faces. With our convention, all the needed symbols will be stored in the same file. In this way, we can reach a compromise between hash tables' memory consumption

and time necessary to read and write the hash tables on disk. We strongly suggest selecting the gauge fixed edge as part of a boundary face to minimize the number of files created. We report a typical time and memory consumption of `J15Symbol_Hash` in Figure 2 and 3.

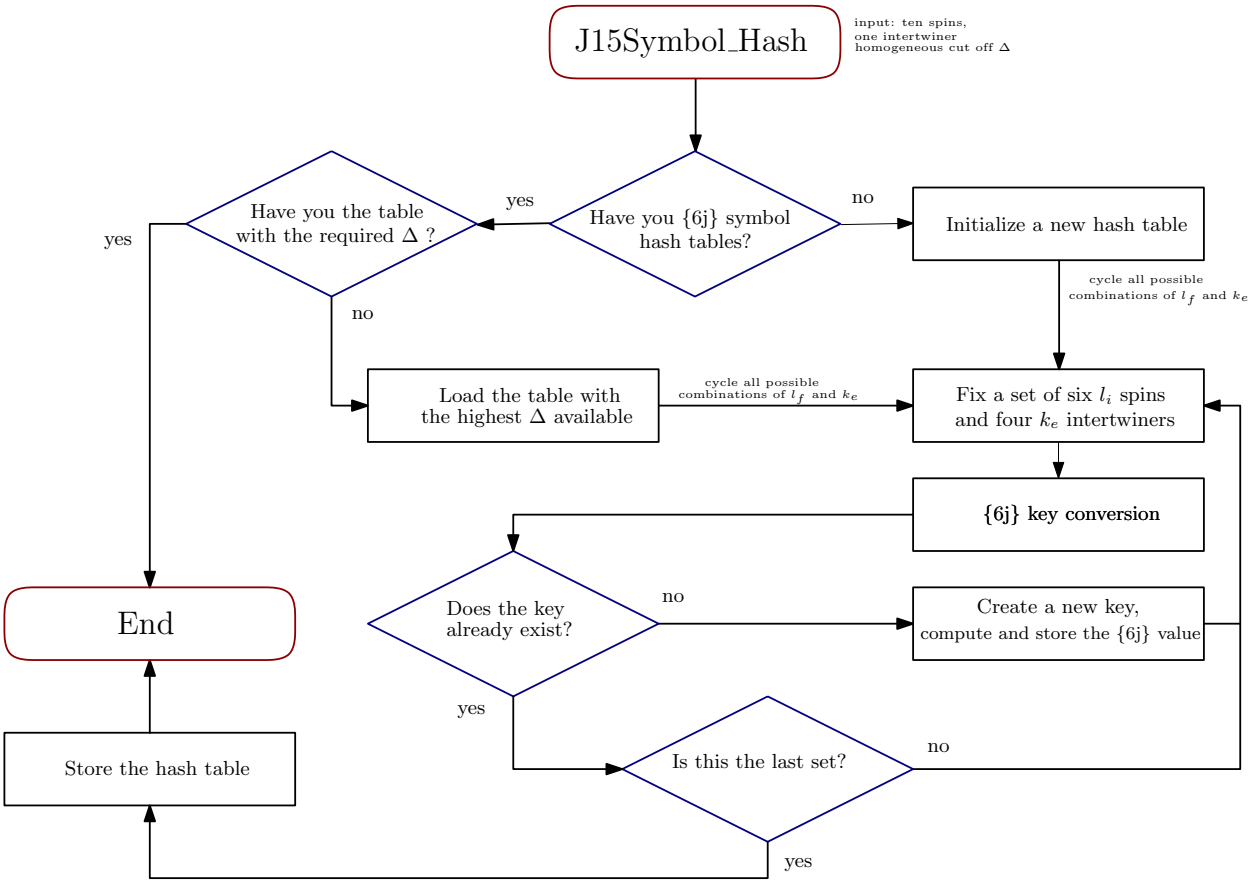


Figure 1: `J15Symbol_Hash` is the main function to store in an efficient way $\{6j\}$ symbols.

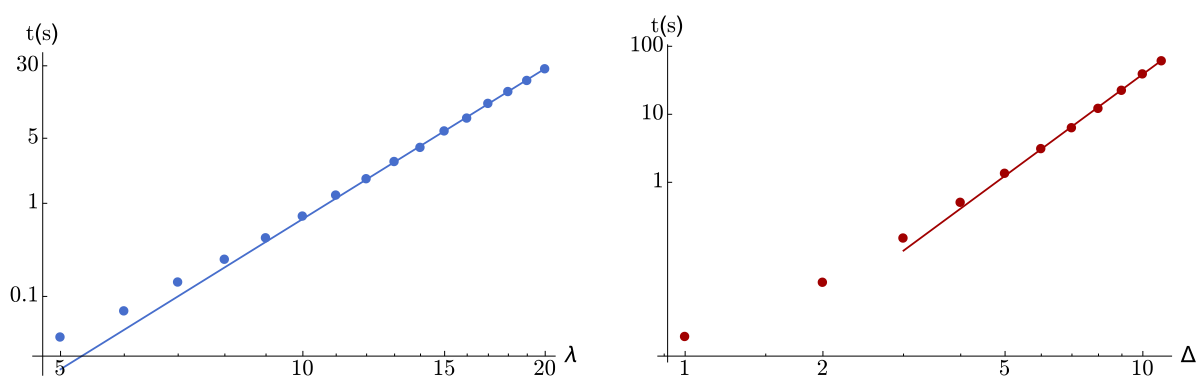


Figure 2: *Evaluation time of the function* `J15Symbol_Hash`. Left panel: *Scaling of the configuration with all the spins and the gauge fixed intertwiner equal* $j_f = i_4 = \lambda$ *and a vanishing cutoff* $\Delta = 0$*, we compute the* $\{15j\}$ *symbols for all the values of the intertwiners* k_e*. We see a power law trend* $a \times j^b$ *with* $a = 2.9 \times 10^{-6} s$ *and* $b = 5.3$. Right panel: *Scaling in the cutoff of the configuration with all the spins and the gauge fixed intertwiner equal to* $j_f = i_4 = 1$*. We see a power law trend* $a \times j^b$ *with* $a = 4 \times 10^{-4} s$ *and* $b = 5.0$.

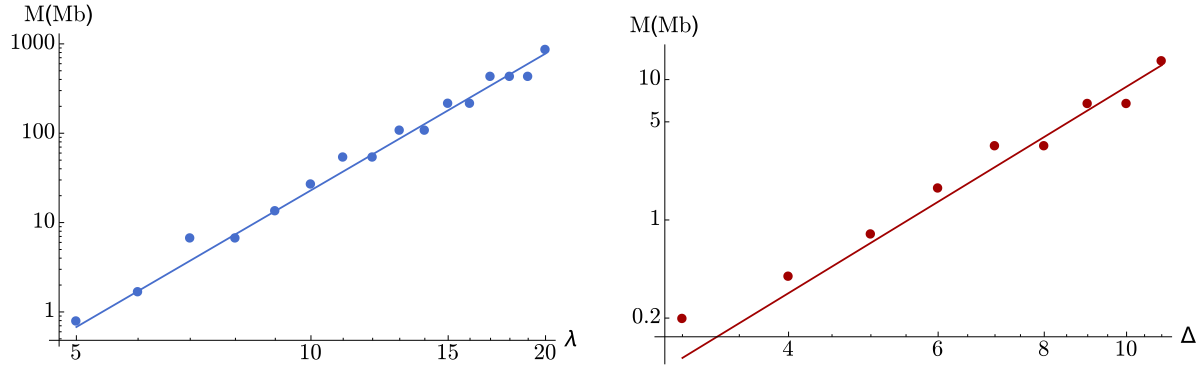


Figure 3: *Memory usage of the function* `J15Symbol_Hash`. Left panel: *Scaling of the configuration with all the spins and the gauge fixed intertwiner equal* $j_f = i_4 = \lambda$ *and a vanishing cutoff* $\Delta = 0$*, we compute the* $\{15j\}$ *symbols for all the values of the intertwiners* k_e*. We see a power law trend* $a \times j^b$ *with* $a = 1.8 \times 10^{-4}$`Mb` *and* $b = 5.1$*.* Right panel: *Scaling in the cutoff of the configuration with all the spins and the gauge fixed intertwiner equal to* $j_f = i_4 = 1$*. We see a power law trend* $a \times j^b$ *with* $a = 1.8 \times 10^{-3}$`Mb` *and* $b = 3.7$*.*

As a possible upgrade for the future, we are planning to avoid the splitting of the $\{9j\}$ symbol as a sum of three $\{6j\}$ symbols in (7) and evaluate it directly with `wigxjpf` with consequent hashing and storing. This upgrade hasn't been implemented yet since it requires an efficient way to assign a unique key to each symmetry class of the $\{9j\}$ symbol, that we are still studying. Anyway, while this upgrade will significantly improve the speed of low spin computations, it will not be applicable in regimes with larger spins because it would use too much memory.

3.3 Booster functions

The expression for the booster functions (3) requires the calculation of the integral over the rapidity of four boost matrix elements (4) and a sum over magnetic indices. The latter is not a problem in this case since the number of terms to be summed over is relatively small (for all equal spins j is less than $(2j)^3$ terms). However, the numerical integration is still very challenging. The integrand is a superposition of highly oscillating hypergeometric functions that, when multiplied together, interfere resulting in a localized, smooth and regular function. Moreover, for specific values, the hypergeometric functions are evaluated near one of their poles. These apparent divergences will cancel with other terms in the superposition but would require extremely high precision to control the accuracy of the result. We solved these issues by attacking them from a different angle. A direct computation is very slow, and with this motivation an alternative formula was proposed in [18] and numerically explored by [49]. Nevertheless, the situation improves significantly using a different representation for the boost matrix elements (4). This problem has been explored in [50] where each boost matrix elements (4) is rewritten as a finite sum of exponentials with complex coefficients. For simple irreducible representations and $\gamma \neq 0$, the result is given by:

$$d^{(\gamma j,j)}_{jlp}(r) = \frac{1}{(e^r - e^{-r})^{j+l+1}} \times \left[\sum_{m=0}^{j+l-|j-p|} Y^{\gamma j,j;j,l;p}_m e^{(j+l-|j-p|-2m-i\gamma j)r} + (-1)^{l-j} \sum_{n=0}^{j+l-|j-p|} \overline{Y}^{\gamma j,j;j,l;-p}_n e^{(j+l-|j-p|-2n+i\gamma j)r} \right], \tag{8}$$

where Y are complex coefficients [3]. In the same work they provide a C++ implementation of the booster functions based on this formula. We integrated this code in `sl2cfoam` porting it in C, we added the evaluation of the $(4jm)$ symbols in (3) with `wigxjpf` and we provided an hash table data structure similar to the one we introduced for the SU(2) invariants in the previous section. The evaluation of these functions requires arbitrary precision floating point numbers, implemented with `MPFR` and `MPC`. These tools allow us to compute the booster functions in an accurate way and to test some of theirs properties.

[3] We refer to [50] for their full expression.

The integral is performed using the trapezoidal rule with 3000 points as a default number. We explored this choice in Figure 4. The convergence looks very fast in the number of points; we observe a slight worsening with the increase of the l_f. The number of integration points can be increased arbitrarily to reduce even further the admissible error on the single booster function.

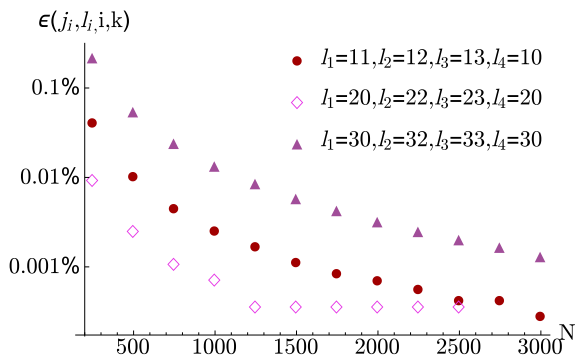


Figure 4: *Dependence of the booster function on the number of points in the rapidity integral. We plot the relative difference ϵ with respect to the value with 10^4 points. All three booster functions reported here have all equal spins $j_f = 10$ and intertwiner $i = 0$. The four spins l_f and intertwiner k are respectively: $l_f = 11, 12, 13, 10$ and $k = 1$, $l_f = 20, 22, 23, 20$ and $k = 2$, $l_f = 30, 32, 33, 30$ and $k = 2$.*

The sampling of the integration interval is parallelized with `OpenMP`. The library `sl2cfoam` implement the evaluation of all the booster functions needed in the computation of a vertex amplitude with the function `B4_Hash`. We illustrate its structure in the diagram in Figure 5.

The values of the booster functions are stored in hash tables and then in a file indexed with the name of the gauge fixed edge and the cutoff Δ, using the same naming conventions as the $\{6j\}$ symbols `2j5.2j6.2j9.2j10_2`Δ. Furthermore, we divide the files into subfolders depending on the Immirzi parameter γ to avoid mixing of incompatible evaluations in case one is interested in working with different values for γ. We store in each file the booster functions with all the possible combinations of the six l_f spins appearing in a vertex amplitude and every compatible intertwiner. We report a typical time consumption of `B4_Hash` in Figure 6 and the scaling of the time as a function of the number of cores used in the computation in Figure 7. The memory's use of booster functions is negligible compared to $\{6j\}$ symbols' one.

For completeness, we provide also a function, called `B4Function`, that computes a single booster, in case one is interested in them outside the context of vertex amplitudes.

When computing the booster functions, one notices that, for a fixed set of spins j_f and l_f, not all combinations of intertwiners contributes with values of similar order of magnitude. An example of this can be seen in [18] and in [51], for the case $l_f = j_f$ (called simplified model), where the main contribution is given by the same intertwiners $i_e = k_e$. The generalization of this result for any set of l_f is unfortunately still unknown. This result, or an equivalent method to localize the dominant contribution from the booster functions, will immediately translate into a significant save of time in the evaluation of the amplitude, especially in the case of multiple vertices.

We also plan to explore the possibility of completely changing the way in which we compute the booster functions and use their expression in terms of the Clebsch-Gordan coefficients for $SL(2, \mathbb{C})$ introduced in [18]. Using the Clebsch-Gordan formulation of the booster functions, we trade the integrals over the rapidity and the sum over the magnetic indices, with a single integral (and a finite sum) over a unitary $SL(2, \mathbb{C})$ virtual irrep. To do so, one has to compute Clebsch-Gordan coefficients fast and with high accuracy. The speed of the functions we included in `sl2cfoam` that do this task is too low to compete with the actual implementation of the booster functions. For this reason, we have not implemented it yet. The possibility of using recursion relations fro the Clebsch-Gordan coefficients also makes this option very appealing. A first attempt was made in [49], we think this is a very promising route to improve `sl2cfoam` .

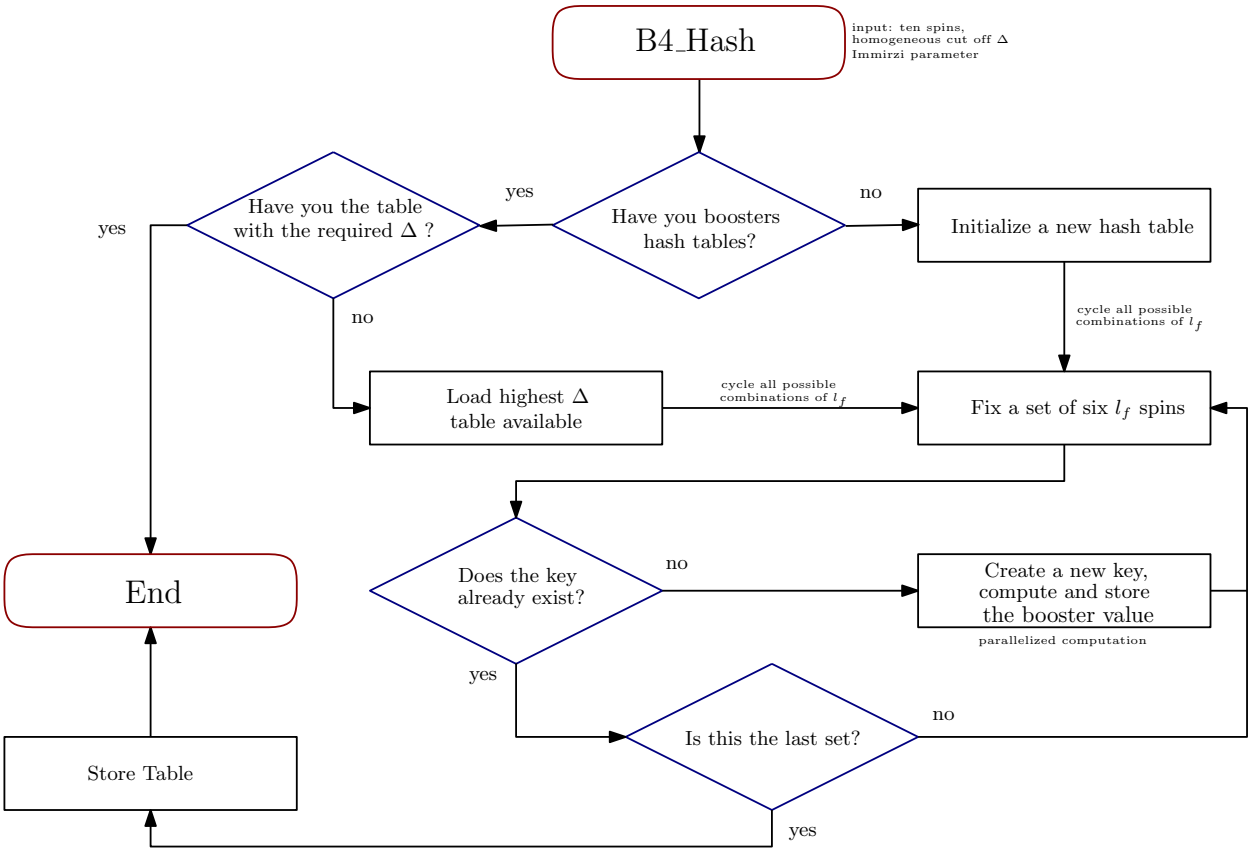


Figure 5: `B4_Hash` compute and store efficiently the booster functions.

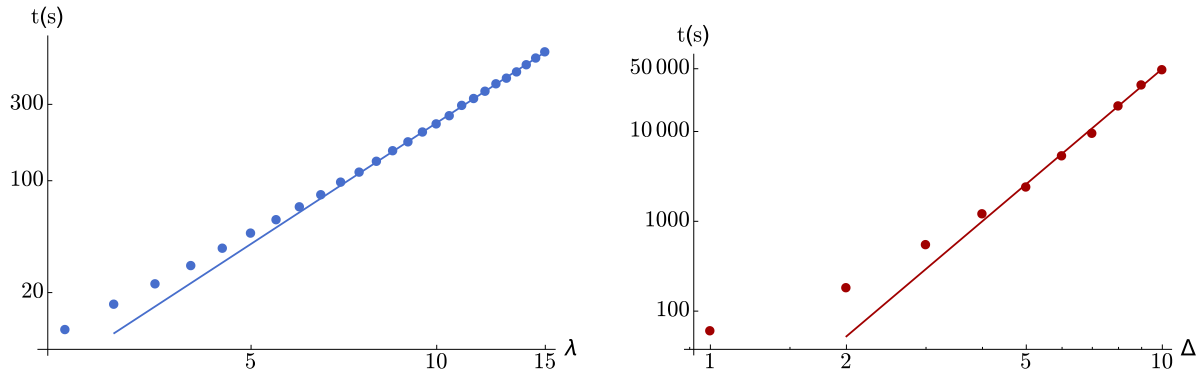


Figure 6: *Evaluation time of the function* `B4_Hash`. Left panel: *Scaling of the configuration with all the spins and the gauge fixed intertwiner equal* $j_f = i_4 = \lambda$ *and a vanishing cutoff* $\Delta = 0$. *We compute all the booster functions for all the values of the intertwiners* k_e. *We see a power law trend* $a \times j^b$ *with* $a = 0.7s$ *and* $b = 2.5$. Right panel: *Scaling in the cutoff of the configuration with all the spins and the gauge fixed intertwiner equal to* $j_f = i_4 = 1$. *We see a power law trend* $a \times j^b$ *with* $a = 2.7s$ *and* $b = 4.3$.

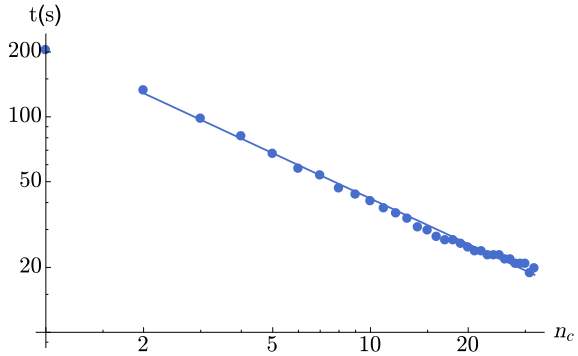


Figure 7: *Parallelization at fixed workload: scaling of the time required by the computation of booster functions for a 4-simplex with $j_f = 10$ and $\Delta = 0$ while increasing the number of cores. The power law $a \times j^b$ reads $a = 209s$ and $b = -0.7$.*

3.4 Four simplex amplitude

In the previous two sections we illustrated the evaluation of the constituents of the vertex amplitude (2). In this section we will discuss how all the contributions are summed together. The summation is performed over six auxiliary spins l_f and four k_e intertwiners, whose ranges depends on the l_f but are generally bounded. The main problem in this step is the extension of these summations. They are, generally, unbounded and numerous. As we already stressed multiple times, the full amplitude is well defined [36], implying that the summations over the l_f are convergent, thus being unbounded is not a problem. Nevertheless, to be able to do a numerical evaluation of the sum we need to introduce a homogeneous cut-off Δ. In the following, we will denote as the Nth *shell* the set of factors contributing to the sum with maximum spin difference $\mathrm{Max}\{l_f - j_f\} = N$. To be concrete we studied the convergence of the amplitude as a function of the cutoff, generally, it is quite rapid. We report some explicit examples in some particular cases in Figure 8. It is not possible to have a unique prescription to set the optimal Δ, since the convergence of the amplitude depends on the details of the data like the face spins j_f and the Immirzi parameter. We explored other configurations and the convergence is qualitatively the same. However, the ratio between the amplitude value with a different number of shells varies.

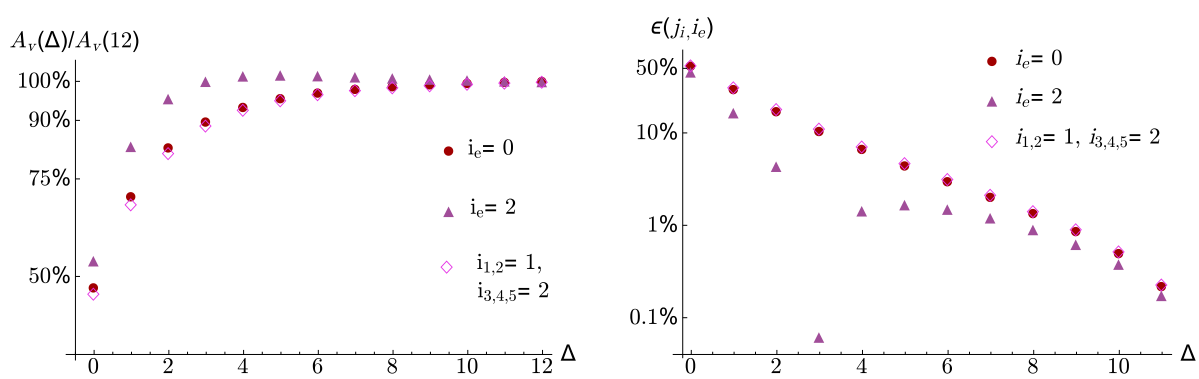


Figure 8: *Convergence of the vertex amplitude.* Left panel: *Convergence of the vertex amplitude while increasing the cutoff Δ for a configuration with all $j_f = 1$ and three different set of boundary intertwiners.* Right panel: *The relative error ϵ between the shell Δ and the shell $\Delta = 12$, that has been taken as the correct value of the amplitude. The error for the simplified model ($\Delta = 0$) is $\sim 50\%$ while from $\Delta = 5$ it decreases to $1\% - 5\%$. The convergence of the vertex amplitude depends on many factors including the chosen boundary configuration and the Immirzi parameter.*

In the library `sl2cfoam` the function `FourSimplex` performs the summation of the 4-simplex vertex amplitude. The algorithm at the base of the implementation is described in Figure 12. This function is the main result of this work and the big star of the library. Given the ten face spins, the five edge intertwiners, the cutoff Δ and the Immirzi parameter, `FourSimplex` uses the functions `J15Symbol_Hash` and `B4_Hash`

described in the previous sections to precompute all the needed SU(2) invariants and booster functions and then proceed to the summations.

This last step needs particular care. Given the vast number of factors to be summed over we need to employ a compensated summation algorithm to reduce the numerical error caused by the sum of finite precision floating point numbers. Furthermore, to minimize the error, we convert and multiply the components of the factors to arbitrary precision floating point numbers (using the library `MPFR`). Parallelization is needed at this step but, while we have implemented it for the summation over the face spins l_f, it was counterproductive to do it also for the four intertwiner sums k_e because of triangular inequalities: when we fix a set of l_f we set the bounds over the k_e and the remaining four sums depend on them. In Figure 9 we show a parallelization test at fixed workload while increasing the number of cores.

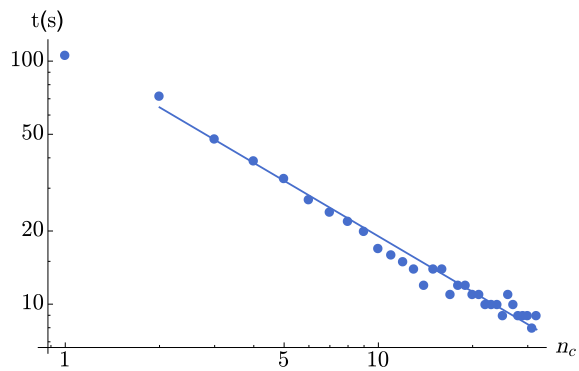


Figure 9: *Parallelization at fixed workload: Scaling of the time required by the computation of a 4-simplex configuration with $j_f = 1$ and $\Delta = 11$ while increasing the number of cores. The power law $a \times j^b$ is given by $a = 109s$ and $b = -0.76$. Since the parallelization acts only on the l_f summations and not on the k_e ones the code performs better, in this regard, when a large Δ is considered.*

The data are saved using the same convention as the boosters functions, we index them as `2j5.2j6.2j9.2j10_2`Δ. We stress that selection rules, to find the important terms within the sums, would be an essential step forward. At the present stage, we can set an empirical cut-off over boosters to reduce the number of terms to be summed over but it would be exciting to study, with new methods such as machine learning's techniques, selection rules for the vertex amplitude.

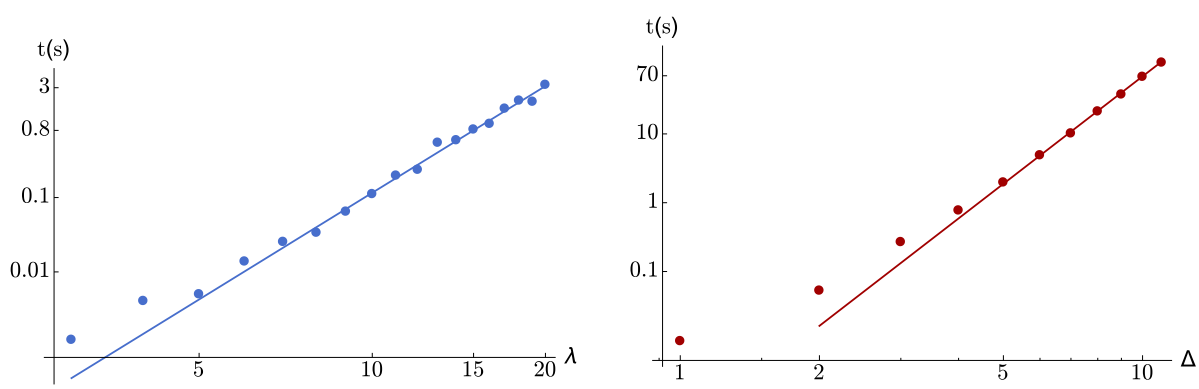


Figure 10: *Evaluation time of the function* `FourSimplex`. Left panel: *Scaling of the configuration with all the spins and the gauge fixed intertwiner equal $j_f = i_4 = \lambda$ and $\Delta = 0$. We see a power law trend $a \times j^b$ with $a = 2.0 \times 10^{-6}s$ and $b = 4.8$.* Right panel: *Scaling in the cutoff of the configuration with all the spins and the gauge fixed intertwiner equal to $j_f = i_4 = 1$. We see a power law trend $a \times j^b$ with $a = 4 \times 10^{-4}s$ and $b = 5.2$.*

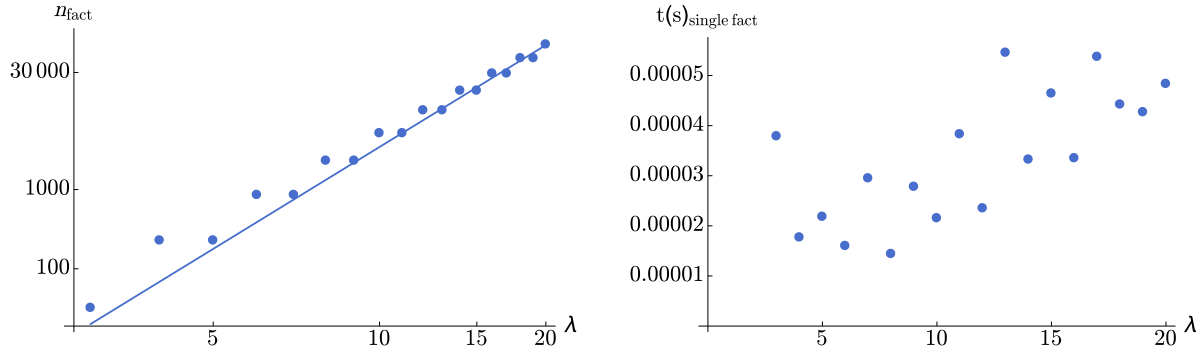


Figure 11: Left panel: *We show the number of factors* $\#_{fact}$ *to be cycled over to compute a single 4-simplex with boundary spins, intertwiners equal to* λ*, and* $\Delta = 0$*. The power law is given by* $a = 0.2$ *and* $b = 4.3$ Right panel: *Evaluation time of a single factor in a computation with all boundary spins and intertwiners equal to* λ*. Each factor takes* $\sim 10^{-5}$ *s. Memory data access using* `khash` *works at* $\sim 10^{-7}$*s: we have to retrieve data for four B4, two "fixed"* $\{6j\}$ *and for a sum of three* $\{6j\}$*, converting data to* `MPFR` *variables and then assembly the amplitude.This leads us to* $\sim 10^{-5}$*s of evaluation time for a single factor of the* l_f, k_e *sum. If we would be able in the future to directly store* $\{9j\}$ *symbols we could save one order of magnitude in time.*

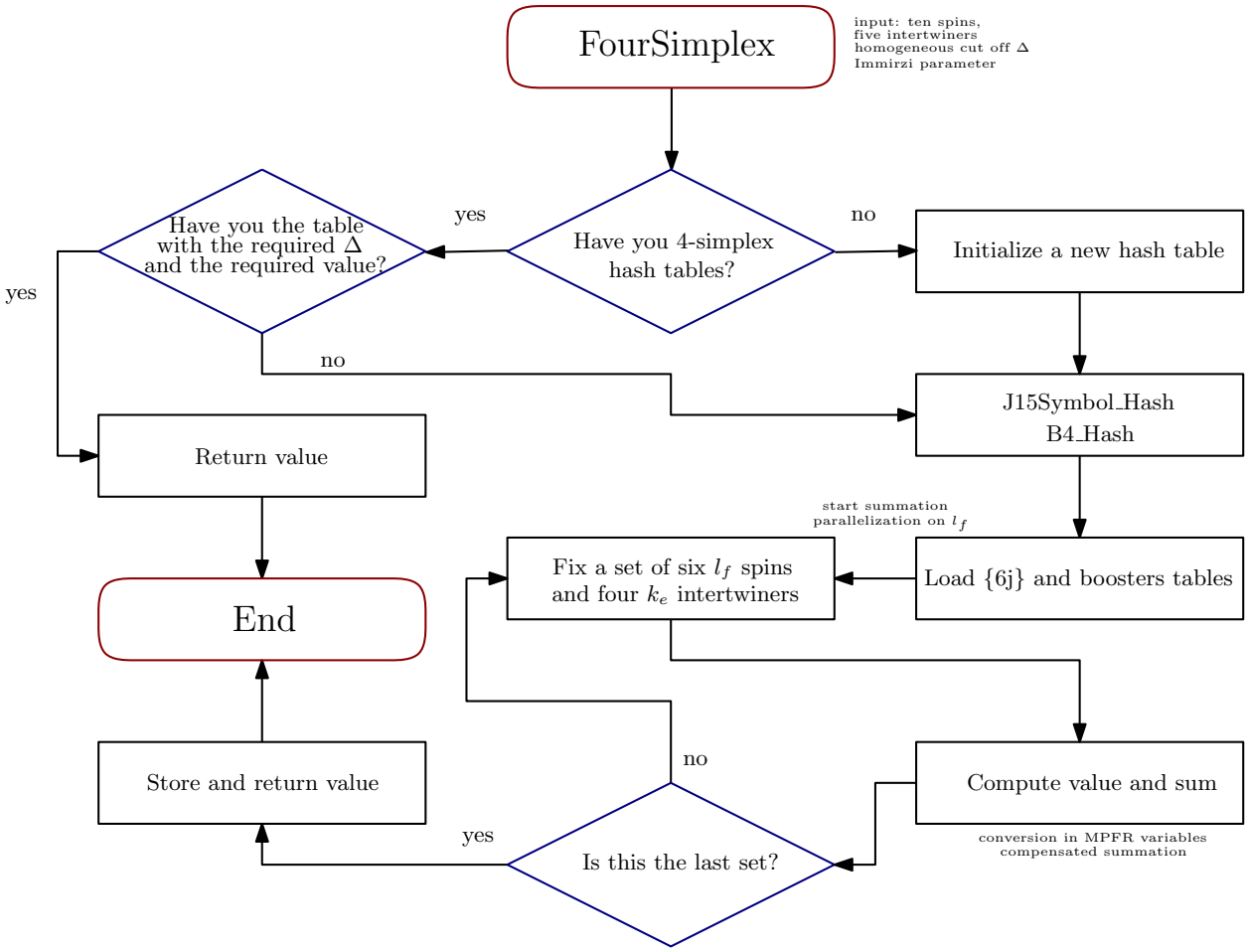


Figure 12: `FourSimplex` is the main function to compute and store EPRL vertex amplitudes

4 Utilities

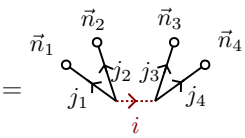
4.1 Coherent states

The study of the semiclassical behavior of a spin foam transition amplitude relies on the relation between polyhedra and SU(2) invariants [6, 19] and coherent intertwiners [2]. Thus, in `sl2cfoam` , we provide a tool to compute Livine-Speziale coherent intertwiners as arbitrary precision floating point using `MPFR`. We refer to [19] for the conventions and background material used in the present work, while we report here the definitions to fix the notation. We recall that an n-valent coherent intertwiner [2] is defined by the group averaging:

$$|\,|\{j_a, \vec{n}_a\}\rangle := \int dg \otimes_{a=1}^{n} g\,|j_a, n_a\rangle \;\in\; \mathrm{Inv} \otimes_{a=1}^{n} V^{(j_a)} \;, \tag{9}$$

where V^{j_a} are SU(2) irreducible representation, $\mathrm{Inv} \otimes_{a=1}^{n} V^{(j_a)}$ is a singlet vector space and the vectors n_a can be parametrized as $\vec{n_a} := (\sin\Theta\cos\Phi, \sin\Theta\sin\Phi, \cos\Theta)$. The library provides a function to compute them for $n = 4$. In this case the decomposition of a coherent intertwiner, in the recoupling channel i with outgoing links, reads

$$c_i(\vec{n}_a) := \langle j_a, i|\, j_a, \vec{n}_a\rangle = \sum_{m_a} \begin{pmatrix} j_1 & j_2 & j_3 & j_4 \\ m_1 & m_2 & m_3 & m_4 \end{pmatrix}^{(i)} \prod_{a=1}^{4} D^{j_a}_{m_a, j_a}(\vec{n_a}) = \tag{10}$$



,

where $D^j_{m,j}(\vec{n}) = D^j_{m,j}(\phi, \Theta, -\phi)$ is the SU(2) Wigner matrix parametrized as:

$$D^j_{m,j}(\phi, \Theta, -\phi) = e^{-im\phi} d^j_{m,j}(\Theta) e^{ij\phi} \;,$$

with $d^j_{m,j}$ as a small Wigner matrix [52].

`CoherentStateI` computes a coherent state coefficient $c_i(\vec{n}_a)$ for a given intertwiner i. It takes as input the four spins, one intertwiner, the four set of angles (Θ_a, ϕ_a) (parametrizing the four normals $\vec{n}_a$) and also four signs related to the orientation of the links. For instance, to compute the coherent intertwiner in (10) one needs to set:

`CoherentStateI` $(2i, 2j_1, 2j_2, 2j_3, 2j_4,\ \phi_1,\ \Theta_1,\ \phi_2,\ \Theta_2,\ \phi_3,\ \Theta_3,\ \phi_4,\ \Theta_4,\ +1,\ +1,\ +1,\ +1)$.

The last four arguments represent the "outgoing" orientation of the four links in the same order the spins are inputted. To represent "ingoing" links is sufficient to flip the sign to -1 of the corresponding entries.

4.2 $SL(2,\mathbb{C})$ symbols

In this section we review the basic formulas of Lorentzian recoupling theory and introduce the code of `sl2cfoam` to compute them. We will follow closely the notation in [18]. The $SL(2,\mathbb{C})$ Clebsch-Gordan coefficients can be factorized in an SU(2) one and a boost one:

$$C^{\rho_3 k_3 j_3 m_3}_{\rho_1 k_1 j_1 m_1 \rho_2 k_2 j_2 m_2} = \chi(\rho_1, \rho_2, \rho_3, k_1, k_2, k_3; j_1, j_2, j_3)\, C^{j_3 m_3}_{j_1 m_1 j_2 m_2} \;. \tag{11}$$

We will use a simplified notation indicating with $\chi(j_1, j_2, j_3) = \chi(\rho_1, \rho_2, \rho_3, k_1, k_2, k_3; j_1, j_2, j_3)$. Denoting with

$$J = \sum_i j_i, \qquad K = \sum_i k_i, \qquad P = \sum_i \rho_i, \tag{12}$$

we can write the boost part of the $SL(2,\mathbb{C})$ Clebsch-Gordan coefficients as a finite sum of ratios of complex gamma functions as:

$$\chi(j_1, j_2, j_3) = \frac{(-1)^{\frac{K+J}{2}}}{4\sqrt{2\pi}} x(\rho_i, k_i) \Gamma(\tfrac{1-iP+K}{2}) \Gamma(\tfrac{1-iP-K}{2}) \sqrt{d_{j_1} d_{j_2} d_{j_3}}\, \kappa(\rho_i, k_i; j_i), \tag{13}$$

with

$$\kappa(\rho_i,k_i;j_i) = (-1)^{-k_1-k_2} e^{-i\left(\Phi^{\rho_1}_{j_1}+\Phi^{\rho_2}_{j_2}-\Phi^{\rho_3}_{j_3}\right)} \frac{(-1)^{j_1-j_2+j_3}}{\sqrt{d_{j_3}}} \left[\frac{(j_1-k_1)!(j_2+k_2)!}{(j_1+k_1)!(j_2-k_2)!}\right]^{1/2} \tag{14}$$
$$\times \sum_{n=-j_1}^{\min\{j_1,k_3+j_2\}} \left[\frac{(j_1-n)!(j_2+k_3-n)!}{(j_1+n)!(j_2-k_3+n)!}\right]^{1/2} \begin{pmatrix} j_1 & j_2 & j_3 \\ n & k_3-n & -k_3 \end{pmatrix}$$
$$\times \sum_{s_1=\max\{k_1,n\}}^{j_1} \sum_{s_2=\max\{-k_2,n-k_3\}}^{j_2} \frac{(-1)^{s_1+s_2-k_1+k_2}\,(j_1+s_1)!(j_2+s_2)!}{(j_1-s_1)!(s_1-k_1)!(s_1-n)!(j_2-s_2)!(s_2+k_2)!(k_3-n+s_2)!}$$
$$\times \frac{\Gamma(\frac{1-i(\rho_1-\rho_2-\rho_3)-K+2s_1}{2})\Gamma(\frac{1+i(\rho_1-\rho_2+\rho_3)+K+2s_2}{2})\Gamma(\frac{1-i(\rho_1+\rho_2-\rho_3)-k_1+k_2+k_3-2n+2s_1+2s_2}{2})}{\Gamma(1-i\rho_1+s_1)\Gamma(1-i\rho_2+s_2)\Gamma(1+i\rho_3+s_1+s_2)\Gamma(\frac{1-iP-k_1+k_2+k_3-2n}{2})},$$

where the additional phase

$$x(\rho_i,k_i) = \frac{\Gamma\left(\frac{1+iP-K)}{2}\right)}{|\Gamma\left(\frac{1+iP-K)}{2}\right)|} \tag{15}$$
$$\times \frac{\Gamma\left(\frac{1-i(-\rho_1+\rho_2+\rho_3)-k_1+k_2+k_3}{2}\right)}{|\Gamma\left(\frac{1-i(-\rho_1+\rho_2+\rho_3)-k_1+k_2+k_3}{2}\right)|} \frac{\Gamma\left(\frac{1-i(\rho_1-\rho_2+\rho_3)+k_1-k_2+k_3}{2}\right)}{|\Gamma\left(\frac{1-i(\rho_1-\rho_2+\rho_3)+k_1-k_2+k_3}{2}\right)|} \frac{\Gamma\left(\frac{1-i(-\rho_1-\rho_2+\rho_3)-k_1-k_2+k_3}{2}\right)}{|\Gamma\left(\frac{1-i(-\rho_1-\rho_2+\rho_3)-k_1-k_2+k_3}{2}\right)|},$$

was introduced in [18] to make the Clebsch-Gordan coefficients, and consequently the booster functions, always real. This phase is different from the convention used in [37].

We can write a $SL(2,\mathbb{C})$ version of the $(3jm)$ symbol. It reads

$$\begin{pmatrix} \rho_1,k_1 & \rho_2,k_2 & \rho_3,k_3 \\ j_1 & j_2 & j_3 \end{pmatrix} = (-1)^{j_1-j_2+j_3}\sqrt{d_{j_3}}\,\chi(j_1,j_2,j_3). \tag{16}$$

We provide a function, `wigner_3j_sl2c`, to compute it. Similarly we can also define an analog of the SU(2) $\{6j\}$ symbol for $SL(2,\mathbb{C})$ as:

$$\{6\rho,6k\} = \sum_{j_i\geq k_i} \{6j_i\} \begin{pmatrix} \rho_1,k_1 & \rho_2,k_2 & \rho_3,k_3 \\ j_1 & j_2 & j_3 \end{pmatrix} \begin{pmatrix} \rho_1,k_1 & \rho_5,k_5 & \rho_6,k_6 \\ j_1 & j_5 & j_6 \end{pmatrix} \tag{17}$$
$$\times \begin{pmatrix} \rho_4,k_4 & \rho_2,k_2 & \rho_6,k_6 \\ j_4 & j_2 & j_6 \end{pmatrix} \begin{pmatrix} \rho_4,k_4 & \rho_5,k_5 & \rho_3,k_3 \\ j_4 & j_5 & j_3 \end{pmatrix},$$

To perform a numerical evaluation of this symbol we artifically cutoff the summation to $j_i \leq \Delta$. We provide `wigner_6j_sl2c` to compute it. Since the κ in (14) is given by a sum of ratio of complex gamma functions, an arbitrary precision computation is needed. We implemented it using `MPFR` variables. We tested the commands of this section up to spins ≈ 50, we leave the improvement of this upper limit to future works.

5 What comes next?

In this paper, we described the library `sl2cfoam` that we developed to compute the spin foam vertex amplitude defined by the EPRL model.

Many technical improvements are necessary to unlock the full potential of this code. In analogy to what we do for the $\{6j\}$ symbols, we need to implement hash tables also for $\{9j\}$ symbols. This would speed up the computation of almost an order of magnitude, trading at low spins memory usage with faster access to data. Moreover, we want to explore the possibility of computing booster functions with the Kerimov formula, as explained in [18], for which recursion relations are available, simplifying the summation over the auxiliary spins l_f. To perform the computation of transition amplitudes with many vertices, it is crucial to improve our parallelization process, using `MPI` and `POSIX` threads enabling many servers calculations. A similar improvement will be reached by finding selection rules to restrict the summation to only the

	$j_f = i_4 = 7$		$j_f = i_4 = 15$		$j_f = 1,\ i_4 = 1,\ \Delta = 5$		$j_f = 1,\ i_4 = 1,\ \Delta = 10$	
$15j$	$0.1s$	6.8Mb	$6.1s$	218Mb	$1.4s$	0.8Mb	$62.0s$	6.8Mb
	$a = 2.9 \cdot 10^{-6}s$		$b = 5.4$		$a = 4 \cdot 10^{-4}s$		$b = 5.0$	
B_4	$99.0s$		$642.1s$		$2433s$		$49214s$	
	$a = 0.7s$		$b = 2.5$		$a = 2.7s$		$b = 4.3$	
A_v	$0.03s$		$0.08s$		$2.0s$		$69.7s$	
	$a = 2.0 \cdot 10^{-6}s$		$b = 4.8$		$a = 4 \cdot 10^{-4}s$		$b = 5.2$	

Table 1: *Summary of the computational time of the functions* `J15Symbol_Hash`, `B4_Hash`, *and* `FourSimplex`. *We report some examples of computational time for configuration with all spins equal to* 7 *and* 15 *in the first two columns and the parameters of the power law behavior* aj^b. *In the last two columns we report some examples of computational time for configuration with all spins equal to* 1 *as a function of the cutoff and the parameters of the power law behavior* $a\Delta^b$. *All the computations are performed with a single processor.*

important terms. Our lack of analytical insight into the booster functions will force us to use new techniques like for example machine learning or Monte Carlo techniques. Finally, we want to search for a dynamical way of dealing with the cutoff Δ to be able to perform computations with an error set a priori. We plan to explore these fascinating possibilities soon.
For a quick reference, we report in Table 1 the timing of the various functions in `sl2cfoam` .

We firmly believe that this code could profoundly change the way we understand the dynamics of loop quantum gravity. One of the most successful computations in the theory proved the connection in the semiclassical limit with discrete General Relativity [6, 7]. They showed that a single EPRL vertex amplitude contains the exponential of Regge action for a Lorentzian 4-simplex in the large spin limit. While this calculation is of dramatic importance for the theory, its reign of validity is limited to a single simplex. We still lack a definite result for an extended triangulation made of many simplices glued together. This limitation made the community questions if the simplicity constraints, responsible of constraining the original topological BF theory to General Relativity, are enforced correctly[22, 23, 24]. This translate to an inability of the model in describing amplitudes compatible with curved spacetime. Preliminary results, limited to the Euclidean version of the model, show that curved geometries are indeed allowed in the asymptotic limit of the amplitude. Using `sl2cfoam` we will investigate the large spin limit of the Δ_3 complex, dual to three 4-simplices sharing a bulk triangle, the most straightforward triangulation with an internal face.

We are performing a consistency check of our code by computing the homogeneous scaling of the Lorentzian single vertex amplitude with Euclidean boundary data. To do so, we calculated the vertex amplitude with spins, both integers and half-integers up to order 10, contracted with coherent intertwiner states representing the boundary of a regular Euclidean four simplex. We sum two shells per amplitude with an estimated error of approximately the 20%. We find an amazing agreement with the analytic formula of [6].

We also plan to explore numerically the divergence of the theory recently estimated in [53, 20] by computing the "self-energy" diagram. A large set of power-law divergences is still compatible with both estimates, and the tensorial structure of the diagram is still unknown. To confirm the first and compute the second will allow us to understand how to correctly regularize the theory without introducing a cosmological constant following a prescription similar to the one studied in [54].

The renormalization group flow of the EPRL vertex amplitude is of crucial importance for the future of the theory. Finding a fixed point in the flow and identifying phase transitions would allow us to understand if the diffeomorphism invariance, broken by the discretization, is recovered and what mechanism will restore it. An interesting insight on this topic is provided in [55, 56, 57] and in [58, 59, 60, 61, 62]. In this last series of works they perform a numerical investigation of the flow of the Euclidean EPRL model and on a hypercubic lattice, with a large number of vertices. The calculations are possible thanks to some strong but vital approximations: they work with the Euclidean version of the theory, they restrict to a specific form of the intertwiners, and instead of using the full vertex amplitude they replace it with its asymptotic formula.

At the present `sl2cfoam` cannot directly attack analog computation, the number of vertices needed for a lattice computation is still out of our reach, but we do not exclude to be able to do it in a near future after all the proposed upgrade are implemented and we have access to more processing power (for the moment we are limited to a 32 cores machine).

Nevertheless, if we restrict to a purely quantum region characterized with relatively small spins, probably up to order 10, we can now compute any transition amplitude with a few vertices. We would like to invite the community to look for interesting computations that could be performed within those limitations, and to develop a common framework in which the theory can be used to answer physically meaningful questions.

Acknowledgments

This work was supported in part by the NSF grants PHY-1505411, PHY-1806356 and the Eberly research funds of Penn State. We want to thank François Collet and Francesco Gozzini for providing a basic code for the booster function and the complex gamma function with arbitrary precision floating points numbers and especially for many interesting and helpful discussions. Thanks to NORDITA and to the organizers of the workshop "Quantum Gravity on the Computer", a very inspiring and useful week in Stockholm. Many thanks to Simone Speziale for fruitful discussions and encouragement and we are very happy to welcome Emilio in this puzzling world.

A SU(2) Symbols

We use the definition of the Wigner's $(3jm)$ symbol reported in [52] with the following orthogonality properties

$$\sum_{m_1,m_2} \begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} \begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & n_3 \end{pmatrix} = \frac{\delta_{j_3 l_3}\delta_{m_3 n_3}}{2j_3+1} \ , \tag{18}$$

implying they are normalized to one and that, if triangular inequalities are not satisfied, they vanish. We can define the $(4jm)$ symbol in the following way, as the contraction of two $(3jm)$ symbol using an intertwiner i

$$\begin{pmatrix} j_1 & j_2 & j_3 & j_4 \\ m_1 & m_2 & m_3 & m_4 \end{pmatrix}^{(i)} \equiv \sum_{m_i} (-1)^{i-m_i} \begin{pmatrix} j_1 & j_2 & i \\ m_1 & m_2 & m_i \end{pmatrix} \begin{pmatrix} i & j_3 & j_4 \\ -m_i & m_3 & m_4 \end{pmatrix} \ , \tag{19}$$

with the orthogonality relations

$$\sum_{m_1,m_2,m_3} \begin{pmatrix} j_1 & j_2 & j_3 & j_4 \\ m_1 & m_2 & m_3 & m_4 \end{pmatrix}^{(i_1)} \begin{pmatrix} j_1 & j_2 & j_3 & j_4 \\ m_1 & m_2 & m_3 & m_4 \end{pmatrix}^{(i_2)} = \frac{\delta_{i_1 i_2}}{2i_1+1} \frac{\delta_{j_4 l_4}\delta_{m_4 n_4}}{2j_4+1} \ , \tag{20}$$

where $\frac{\delta_{i_1 i_2}}{2i_1+1}$ is the normalization factor.

References

[1] Jonathan Engle, Etera Livine, Roberto Pereira, and Carlo Rovelli. LQG vertex with finite Immirzi parameter. *Nucl. Phys.*, B799:136–149, 2008.

[2] Etera R. Livine and Simone Speziale. A New spinfoam vertex for quantum gravity. *Phys. Rev.*, D76:084028, 2007.

[3] Etera R. Livine and Simone Speziale. Consistently Solving the Simplicity Constraints for Spinfoam Quantum Gravity. *EPL*, 81(5):50004, 2008.

[4] Laurent Freidel and Kirill Krasnov. A New Spin Foam Model for 4d Gravity. *Class. Quant. Grav.*, 25:125018, 2008.

[5] Wojciech Kaminski, Marcin Kisielowski, and Jerzy Lewandowski. Spin-Foams for All Loop Quantum Gravity. *Class. Quant. Grav.*, 27:095006, 2010. [Erratum: Class. Quant. Grav.29,049502(2012)].

[6] John W. Barrett, R. J. Dowdall, Winston J. Fairbairn, Henrique Gomes, and Frank Hellmann. Asymptotic analysis of the eprl four-simplex amplitude. *J. Math. Phys.*, 50:112504, 2009.

[7] John W. Barrett, R. J. Dowdall, Winston J. Fairbairn, Frank Hellmann, and Roberto Pereira. Asymptotic analysis of Lorentzian spin foam models. *PoS*, QGQGS2011:009, 2011.

[8] Eugenio Bianchi, Leonardo Modesto, Carlo Rovelli, and Simone Speziale. Graviton propagator in loop quantum gravity. *Class. Quant. Grav.*, 23:6989–7028, 2006.

[9] Simone Speziale. Background-free propagation in loop quantum gravity. *Adv. Sci. Lett.*, 2:280–290, 2009.

[10] Eugenio Bianchi, Elena Magliaro, and Claudio Perini. LQG propagator from the new spin foams. *Nucl. Phys.*, B822:245–269, 2009.

[11] Eugenio Bianchi and You Ding. Lorentzian spinfoam propagator. *Phys. Rev.*, D86:104040, 2012.

[12] Emanuele Alesci and Carlo Rovelli. The Complete LQG propagator. I. Difficulties with the Barrett-Crane vertex. *Phys. Rev.*, D76:104012, 2007.

[13] Emanuele Alesci and Carlo Rovelli. The Complete LQG propagator. II. Asymptotic behavior of the vertex. *Phys. Rev.*, D77:044024, 2008.

[14] Emanuele Alesci, Eugenio Bianchi, and Carlo Rovelli. LQG propagator: III. The New vertex. *Class. Quant. Grav.*, 26:215001, 2009.

[15] J. Daniel Christensen, Etera R. Livine, and Simone Speziale. Numerical evidence of regularized correlations in spin foam gravity. *Phys. Lett.*, B670:403–406, 2009.

[16] J. Daniel Christensen, Igor Khavkine, Etera R. Livine, and Simone Speziale. Sub-leading asymptotic behaviour of area correlations in the Barrett-Crane model. *Class. Quant. Grav.*, 27:035012, 2010.

[17] Giorgio Sarno, Simone Speziale, and Gabriele V. Stagno. 2-vertex Lorentzian Spin Foam Amplitudes for Dipole Transitions. *Gen. Rel. Grav.*, 50(4):43, 2018.

[18] Simone Speziale. Boosting wigner's nj-symbols. *J. Math. Phys.*, 58(3):032501, 2017.

[19] Pietro Dona, Marco Fanizza, Giorgio Sarno, and Simone Speziale. SU(2) graph invariants, Regge actions and polytopes. *Class. Quant. Grav.*, 35(4):045011, 2018.

[20] Pietro Dona. Infrared divergences in the EPRL-FK Spin Foam model. 2018.

[21] Pietro Dona, Marco Fanizza, Giorgio Sarno, and Simone Speziale. Numerical studies of the Lorentzian EPRL vertex amplitude. *In Preparation*.

[22] Florian Conrady and Laurent Freidel. On the semiclassical limit of 4d spin foam models. *Phys. Rev.*, D78:104023, 2008.

[23] Valentin Bonzom. Spin foam models for quantum gravity from lattice path integrals. *Phys. Rev.*, D80:064028, 2009.

[24] Frank Hellmann and Wojciech Kaminski. Holonomy spin foam models: Asymptotic geometry of the partition function. *JHEP*, 10:165, 2013.

[25] José Ricardo Oliveira. EPRL/FK Asymptotics and the Flatness Problem. *Class. Quant. Grav.*, 35(9):095003, 2018.

[26] Vincent Bayle, François Collet, and Carlo Rovelli. Short-scale Emergence of Classical Geometry, in Euclidean Loop Quantum Gravity. 2016.

[27] Marios Christodoulou, Carlo Rovelli, Simone Speziale, and Ilya Vilensky. Planck star tunneling time: An astrophysically relevant observable from background-free quantum gravity. *Phys. Rev.*, D94(8):084035, 2016.

[28] Marios Christodoulou and Fabio D'Ambrosio. Characteristic Time Scales for the Geometry Transition of a Black Hole to a White Hole from Spinfoams. 2018.

[29] Eugenio Bianchi, Carlo Rovelli, and Francesca Vidotto. Towards Spinfoam Cosmology. *Phys. Rev.*, D82:084035, 2010.

[30] Francesca Vidotto. Many-nodes/many-links spinfoam: the homogeneous and isotropic case. *Class. Quant. Grav.*, 28:245005, 2011.

[31] Emanuele Alesci, Klaus Liegener, and Antonia Zipfel. Matrix elements of Lorentzian Hamiltonian constraint in loop quantum gravity. *Phys. Rev.*, D88(8):084043, 2013.

[32] Alejandro Perez. The spin foam approach to quantum gravity. *Living Rev. Rel.*, 16:3, 2013.

[33] Carlo Rovelli and Francesca Vidotto. *Covariant loop quantum gravity: an elementary introduction to quantum gravity and spinfoam theory*. Cambridge University Press, Cambridge (UK), 2015.

[34] Eugenio Bianchi, Daniele Regoli, and Carlo Rovelli. Face amplitude of spinfoam quantum gravity. *Class. Quant. Grav.*, 27:185009, 2010.

[35] Jonathan Engle, Roberto Pereira, and Carlo Rovelli. The Loop-quantum-gravity vertex-amplitude. *Phys. Rev. Lett.*, 99:161301, 2007.

[36] Jonathan Engle and Roberto Pereira. Regularization and finiteness of the Lorentzian LQG vertices. *Phys. Rev.*, D79:084034, 2009.

[37] Werner Ruhl. *The Lorentz group and harmonic analysis*. The Mathematical physics monograph series. New York: W. A. Benjamin, 1970.

[38] M.A. Naimark and H.K. Farahat. *Linear Representations of the Lorentz Group*. International Series in Pure and Applied Mathematics. Elsevier Science, 2014.

[39] M. A. Rashid. Boost Matrix Elements of the Homogeneous Loretz Group. *J. Math. Phys.*, 20:1514–1519, 1979.

[40] Marco Fanizza, Pierre Martin-Dussaud, and Simone Speziale. Asymptotics of SL(2,C) tensor invariants. *In Preparation*.

[41] Torbjorn Granlund and the GMP development team. GNU multiple precision arithmetic library 6.1.2, December 2016. `https://gmplib.org/`.

[42] Laurent Fousse, Guillaume Hanrot, Vincent Lefèvre, Patrick Pélissier, and Paul Zimmermann. MPFR: A Multiple-Precision Binary Floating-Point Library With Correct Rounding, 2018.

[43] Andreas Enge, Patrick Pélissier, and Paul Zimmermann. *MPC: Multiple Precision Complex Library*. INRIA, 1.1.0 edition, 2018. `http://www.multiprecision.org/mpc/`.

[44] William Kahan. *Pracniques: Further Remarks on Reducing Truncation Errors*, *Commun. ACM* 8, (1965) 40,

[45] A. P. Yutsis, I. B. Levinson, and V. V. Vanagas. *Mathematical apparatus of the theory of angular momentum*. 1962.

[46] H. T. Johansson and C. Forssen. Fast and accurate evaluation of Wigner 3j, 6j, and 9j symbols using prime factorisation and multi-word integer arithmetic. *SIAM J. Sci. Statist. Comput.*, 38:A376–A384, 2016.

[47] Open Source. Klib. `https://github.com/attractivechaos/klib/`, 2016.

[48] J. Rasch and A. C. H. Yu. Efficient storage scheme for precalculated wigner 3j, 6j and gaunt coefficients. *SIAM J. Sci. Comput.*, 25(4):1416–1428, April 2003.

[49] Francesco Gozzini. Numerical study of correlations in a Lorentzian spinfoam geometry. *In Preparation*.

[50] François Collet. A (simple) expression of the unitary-irreducible SL(2,C) representations as a finite sum of exponentials. *In Preparation*.

[51] Jacek Puchta. Asymptotic of Lorentzian Polyhedra Propagator. 2013.

[52] D. A. Varshalovich, A. N. Moskalev, and V. K. Khersonskii. *Quantum Theory of Angular Momentum*. World Scientific, Singapore, 1988.

[53] Aldo Riello. Self-energy of the Lorentzian Engle-Pereira-Rovelli-Livine and Freidel-Krasnov model of quantum gravity. *Phys. Rev.*, D88(2):024011, 2013.

[54] L. Freidel and D. Louapre, Diffeomorphisms and spin foam models. *Nucl. Phys.* B **662** (2003) 279 doi:10.1016/S0550-3213(03)00306-7

[55] B. Dittrich, The continuum limit of loop quantum gravity - a framework for solving the theory. Loop Quantum Gravity, Chapter 5, 153-179, World Scientific doi:10.1142/97898132200030006

[56] B. Dittrich, E. Schnetter, C. J. Seth and S. Steinhaus, Coarse graining flow of spin foam intertwiners. *Phys. Rev.* D **94**, no. 12, 124050 (2016) doi:10.1103/PhysRevD.94.124050

[57] C. Delcamp and B. Dittrich, Towards a phase diagram for spin foams. *Class. Quant. Grav.* **34** (2017) no.22, 225006 doi:10.1088/1361-6382/aa8f24

[58] Benjamin Bahr. On background-independent renormalization of spin foam models. *Class. Quant. Grav.*, 34(7):075001, 2017.

[59] Benjamin Bahr and Sebastian Steinhaus. Numerical evidence for a phase transition in 4d spin foam quantum gravity. *Phys. Rev. Lett.*, 117(14):141302, 2016.

[60] Benjamin Bahr and Sebastian Steinhaus. Hypercuboidal renormalization in spin foam quantum gravity. *Phys. Rev.*, D95(12):126006, 2017.

[61] Benjamin Bahr, Sebastian Kloser, and Giovanni Rabuffo. Towards a Cosmological subsector of Spin Foam Quantum Gravity. *Phys. Rev.*, D96(8):086009, 2017.

[62] Benjamin Bahr, Giovanni Rabuffo, and Sebastian Steinhaus. Renormalization in symmetry restricted spin foam models with curvature. 2018.