Title: Advances in numerical techniques for spinfoam amplitudes with applications

Speakers: Francesco Gozzini

Series: Quantum Gravity

Date: September 10, 2020 - 2:30 PM

URL: http://pirsa.org/20090003

Abstract: The computation of transition amplitudes in Loop Quantum Gravity is still a hard task, especially without resorting to large-spins approximations. In Marseille we are actively developing a C library (sl2cfoam) to compute Lorentzian EPRL amplitudes with many vertices. We have already applied this tool to obtain interesting results in spinfoam cosmology and on the so-called flatness problem of spinfoam models. I will highlight the main aspects of our software library, describe the current applications and also introduce undergoing developments (Montecarlo over spin space, tensor contractions, GPU offloading) that will greatly enhance the library's speed and capabilities.


Advances in numerical techniques for spinfoam amplitudes with applications
Francesco Gozzini, CPT, Aix-Marseille University
Diapositiva 1
Google Slides

> plan of the talk

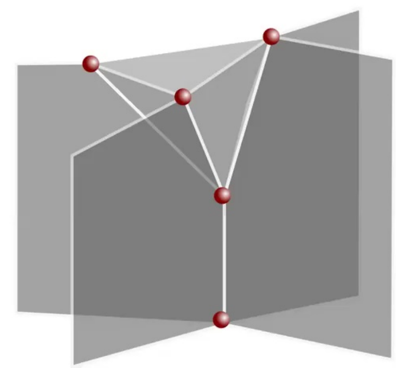
- **Spinfoams and the EPRL model**
  - Definitions and geometry
  - Factorization of EPRL model for computation
- **The library `sl2cfoam`**
  - Introduction and examples
- **Application: spinfoam cosmology**
  - Single-vertex model
  - Many-vertices models
- **Application: flatness problem**
  - Are spinfoams flat?
  - Numerical insights into the resolution
- **Ongoing developments: `sl2cfoam-next`**
  - New tools, new techniques, new performance

> spinfoams, simplices and all that

Covariant LQG defines transition amplitudes between quantum states of geometry defined on discrete abstract graphs. It helps to consider the graphs as duals to **triangulations** made by 4-simplices, tetrahedra, triangles and segments.

A **spinfoam** is a graph (2-complex) made by vertices, edges and faces. Given a gauge group G, faces are colored by representations of G, edges are colored by intertwiners (invariant tensors in tensor products of G). Edges are associated to the matrix of parallel transport between two vertices (modulo gauge-invariance), faces are associated to the smearing of the frame field along the dual hyper-surface.

Clear **geometrical interpretation in 4d**: if the graph is dual to a triangulation by 4-simplices, then a vertex is dual to a 4-simplex, an edge is dual to tetrahedron, a face is dual to a triangle.

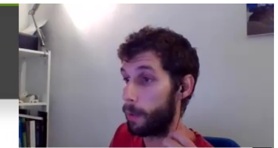


The boundary of a n-dimensional spinfoam defines a (n-1)-dimensional **spin network** state. A spinfoam represents one term of the expansion of the discrete gravitational path-integral with prescribed boundary.
The corresp. partition function is the sum over all **bulk degrees of freedom**:

$$\int e^{\, i \int \sqrt{-g} R} \approx$$

$$\sum_{(j_f, i_e)} \prod_f d_{j_f} \prod_e d_{i_e} \prod_v A_v(j_f, i_e)$$

> the EPRL model in a nutshell



The **vertex amplitude** contains the dynamics of the model. Classically for Lagrangian $\mathcal{L} = B \wedge F$ we have a topological BF theory of the gauge group G. The quantum discretized version for compact gauge group G = SU(2) is well-understood in 3d (Ponzano-Regge model, $A_v \sim \{6j\}$) and 4d (Ooguri model, $A_v \sim \{15j\}$).

To get the non-topological physically relevant 4d theory there are two steps:

- replace G = SU(2) with non-compact **G = SL(2,C)**
- introduce **simplicity constraints** to impose metricity of the G-connection and break topological invariance

The **EPRL model** implements the constraints at quantum level in a weak sense. In a nutshell, this amounts to insert the so called *Y-map* at the interface between the vertices (on the edges), mapping SU(2) irreps to SL(2,C) **γ-simple irreps**.

$$Y_\gamma : D^{(j)}_{mn}(g) \mapsto D^{(\gamma j, j)}_{jmjn}(h)$$

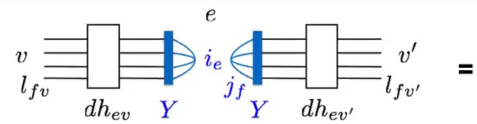
Still **open questions**: *Are the constraints imposed correctly? What is the classical limit of the theory? And how to approach this limit?* There are (positive?) results with the large-spin limit of a single vertex[1] and many vertices[2] but the questions are not settled in full generality.

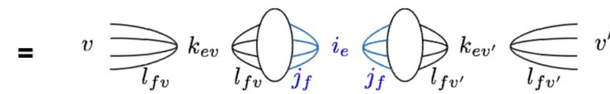
⟹ Here I outline a numerical path to study this questions and to apply the model to physically interesting settings.

[1] Barrett et al. (J.Math.Phys., 2009) 0902.1170
[2] Han and Zhang (Class. Quantum Grav., 2013) 1109.0499

> how to *compute* EPRL amplitudes

It is convenient[1] to compute the EPRL vertex as the **convergent sum over "shells"** of the composition of the BF vertex ({ 15j }) with *booster coefficients* that are inserted along the edges.





The computation of a **single-vertex** amplitude consists in the following steps:

1. fix a boundary state (spin network, coherent, ...) and number of shells Δs
2. compute all the { 15j } symbols and booster coefficients for each shell
3. multiply and sum these over all shells
4. contract with boundary state
5. study the convergence in the Δs shell parameter

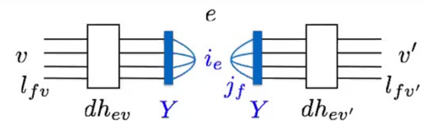
$$B(j_l,l_f;i_n,k_e)=\sum_{p_i}\begin{pmatrix} j_l\\ p_i\end{pmatrix}^{(i_n)}\begin{pmatrix} l_f\\ p_i\end{pmatrix}^{(k_e)} \times\left(\int_0^\infty \mathrm{d}r\, \frac{\sinh^2 r}{4\pi}\prod_{i=1}^{4} d^{(\gamma j_i,j_i)}_{j_i l_i p_i}(r)\right)$$

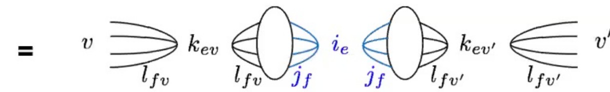
For a **many-vertices** amplitude:

1. draw the spinfoam graph and write down all the summations over bulk intertwiners and bulk faces (possibly infinite and divergent without renormalization, cutoffs needed)
2. compute the single-vertex amplitudes for all combinations of bulk intertwiners and faces
3. multiply and sum all combinations of single-vertex amplitudes
4. contract with boundary state
5. study the convergence in all the cutoff parameters

[1] Speziale (J.Math.Phys., 2017) 1609.01632

> how to *compute* EPRL amplitudes





It is convenient[1] to compute the EPRL vertex as the **convergent sum over "shells"** of the composition of the BF vertex ({ 15j }) with *booster coefficients* that are inserted along the edges.

$$B(j_l, l_f; i_n, k_e) = \sum_{p_i} \begin{pmatrix} j_l \\ p_i \end{pmatrix}^{(i_n)} \begin{pmatrix} l_f \\ p_i \end{pmatrix}^{(k_e)} \times \left(\int_0^\infty \mathrm{d}r \, \frac{\sinh^2 r}{4\pi} \prod_{i=1}^{4} d^{(\gamma j_i, j_i)}_{j_i l_i p_i}(r) \right)$$

The computation of a **single-vertex** amplitude consists in the following steps:

1. fix a boundary state (spin network, coherent, ...) and number of shells Δs
2. compute all the { 15j } symbols and booster coefficients for each shell
3. multiply and sum these over all shells
4. contract with boundary state
5. study the convergence in the Δs shell parameter

For a **many-vertices** amplitude:

1. draw the spinfoam graph and write down all the summations over bulk intertwiners and bulk faces (possibly infinite and divergent without renormalization, cutoffs needed)
2. compute the single-vertex amplitudes for all combinations of bulk intertwiners and faces
3. multiply and sum all combinations of single-vertex amplitudes
4. contract with boundary state
5. study the convergence in all the cutoff parameters

Better said than done...

[1] Speziale (J.Math.Phys., 2017) 1609.01632

> sl2cfoam

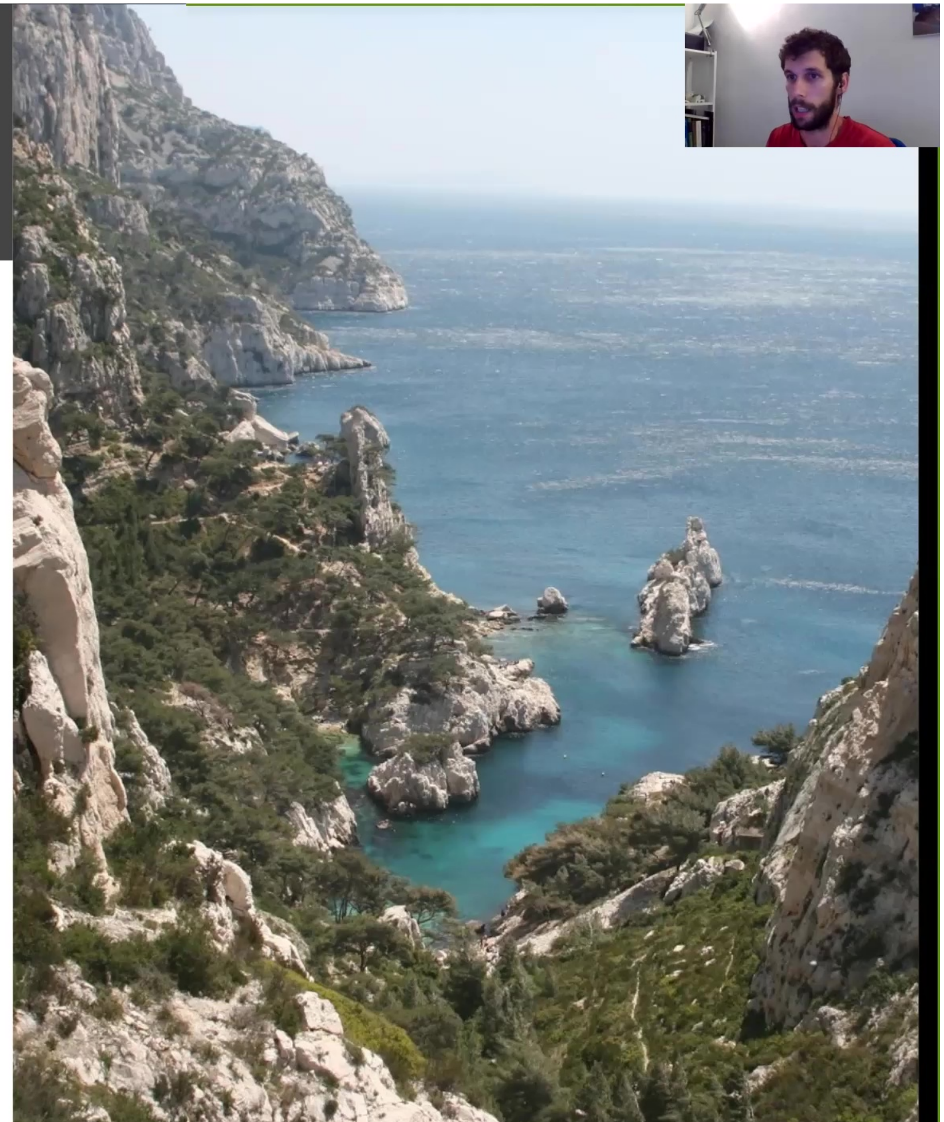
A library for computing EPRL transition amplitudes [1, 2]

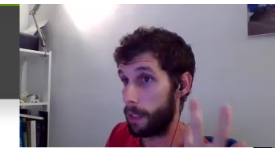
Developed in Marseille by
- me 👋
- Giorgio Sarno
- Pietro Donà

(with contributions from François Collet).

The library is written in C and consists of ~12000 lines of code. It uses external libraries for various tasks, in particular:
- WIGXJPF[3] for { 3j, 6j, 9j } symbols
- GMP, MPFR, MPC for arbitrary precision arithmetic
- OpenMP for parallelization

[1] *github.com/qg-cpt-marseille/sl2cfoam*
[2] Donà, Sarno (GRG, 2018) 1807.03066
[3] Johansson, Forssén (SIAM Journal on Scientific Computing, 2015) 1504.08329





> #include “sl2cfoam.h”; sl2cfoam_four_simplex(j1, j2, …)

Computations included in the library:

- EPRL 4d and 3d vertex amplitudes
- BF 4d and 3d vertex amplitudes
- booster coefficients
- SL(2, C) recoupling symbols using recursion relations
- Livine-Speziale coherent states coefficients

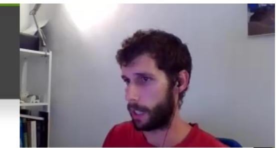
The main task is the computation of a single 4d Lorentzian EPRL vertex amplitude. Inputs are 15 boundary spins, γ (Barbero-Immirzi constant) and Δs (number of shells). Then:

- all the necessary Wigner symbols and booster coefficients are computed
- these are composed and summed over virtual intertwiners and spins to obtain the amplitude

```
GNU nano 4.8                                          ampl.c
#include <stdio.h>
#include <stdlib.h>

#include "sl2cfoam.h"

int main(int arg, char** argv) {

        sl2cfoam_init();


        int two_J = (int)atoi(argv[1]);
        double ampl = sl2cfoam_four_simplex(two_J, two_J, two_J, two_J,two_J, two_J, two_J, two_J,two_J, two_J,
                                0, 0, 0, 0, 0, 2, 0.2);

        printf("amplitude = %.6g\n", ampl);

        sl2cfoam_free();
}
```

Simple!

> time, memory, single-vertex asymptotics



- For very low boundary spins (j ~ 2) and very low number of shells (Δs ~ 2) the amplitudes are computed in **seconds** on a normal laptop.
- All the Wigner symbols ({6j}s) and all the boosters are computed just once and stored into hashtables on disk for reuse.
- The computed amplitudes are stored, too (hence 2nd call is instantaneous).

The time grows with a power law as boundary spins and/or number of shell increase. If the avg. boundary spin is *j* an estimate of the growth factor is $(2j+1)^4(\Delta s+1)^6$.
This comes from cycling over all combinations of 4 virtual intertwiners and 6 virtual spins.

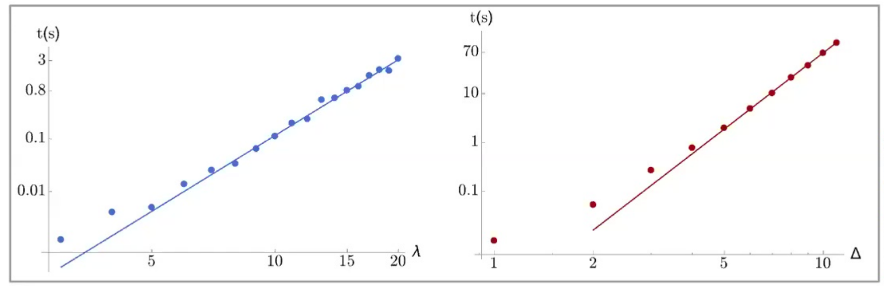


Image taken from [3]

The library has been initially applied to test the large-spin asymptotics of the single vertex, both with the BF and the Lorentzian EPRL amplitudes and with Euclidean and Lorentzian boundary conditions[1, 2].

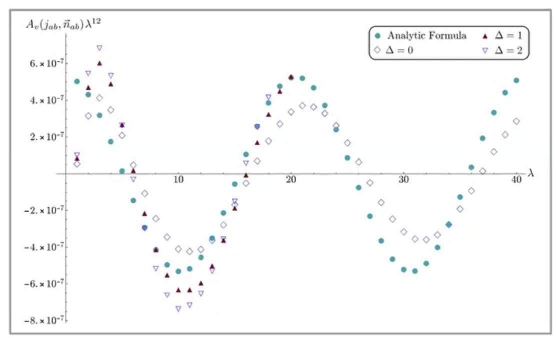
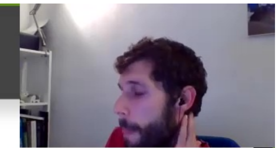


Image taken from [2]

[1] Donà et al. (Class. Quantum Grav., 2018) 1708.01727
[2] Donà et al. (PRD, 2019) 1903.12624
[3] Donà, Sarno (GRG, 2018) 1807.03066

> application: spinfoam cosmology



A first, physically interesting application[1]. Model a closed universe as a topological 3-sphere. Its simplest regular triangulation is the 4-simplex, which is dual to a single spinfoam vertex.
Set boundary spins $(j, j, \dots, j; i_1, \dots, i_5)$ for the vertex and define the state $|\psi_0\rangle$ as

$$\langle j; i_n|\psi_0\rangle = A_v(j, i_n)$$

The 4-simplex boundary has a unique connected component. If we interpret it as *the future one* then:

- ➔ $A_v(j, i_n)$ is the amplitude for the transition *nothing-to-*$|j; i_n\rangle$ (to lowest order in spinfoam expansion)
- ➔ the vertex amplitude describes **the amplitude of the transition from nothing to the 3-geometry** of the boundary
- ➔ the state $|\psi_0\rangle$ is the analogue in Lorentzian LQG of the Hawking "no-boundary" wavefunction defined with a (Euclidean) path-integral over compact geometries. Explicitly:

$$|\psi_0\rangle = \sum_{(i_1,\dots,i_5)} A_v(j, i_a)\, |j, i_1\rangle \otimes \dots \otimes |j, i_5\rangle$$

The boundary spins *j* are taken to be equal and fixed, they determine the areas of the boundary triangles.
The boundary intertwiners are quantum degrees of freedom and describe the **"shapes"** of the boundary tetrahedra.
We can define **geometrical observables** on the state $|\psi_0\rangle$ and study them as functions of the boundary spins *j*.

[1] Gozzini, Vidotto 1906.02211

> application: spinfoam cosmology

There is no “bulk” so we define operators on the 3d boundary. Examples:

- intertwiners (3d dihedral angles)
- volume
- length[1]
- “3d scalar curvature” [2]

We can also look at how the entanglement entropy between different (groups of) tetrahedra scales and compare with other states[3].

The amplitudes are computed for all boundary values. Results:

- all the **expectation values are compatible with regular tetrahedra** on the boundary of a regular geometrical 4-simplex
- **large fluctuations** (spread of operators)
- correlations do **not** vanish as $j \to \infty$
- results qualitatively similar if using BF or Lorentzian EPRL vertex (except for entanglement entropy)

[1] Bianchi (Nuclear Physics B, 2009) 0806.4710
[2] Alesci et al. (PRD, 2014) 1403.3190
[3] Bianchi et al. (PRD, 2019) 1812.10996

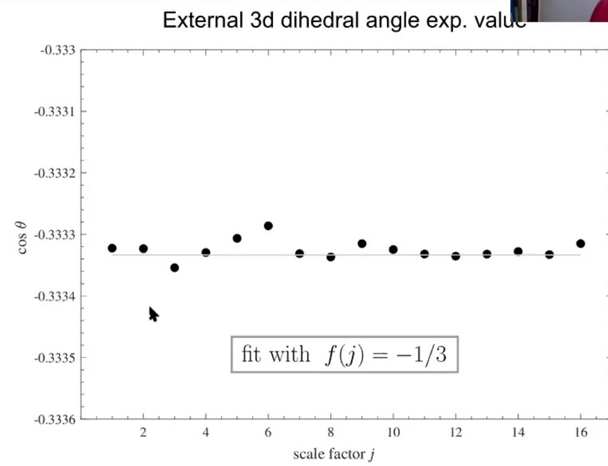
> application: spinfoam cosmology

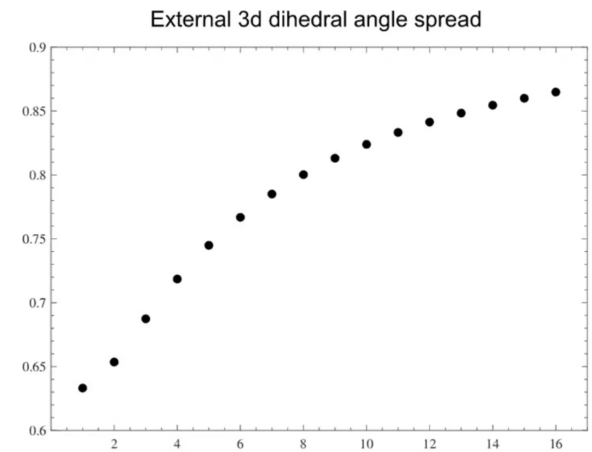
There is no "bulk" so we define operators on the 3d boundary. Examples:
- intertwiners (3d dihedral angles)
- volume
- length[1]
- "3d scalar curvature" [2]

We can also look at how the entanglement entropy between different (groups of) tetrahedra scales and compare with other states[3].

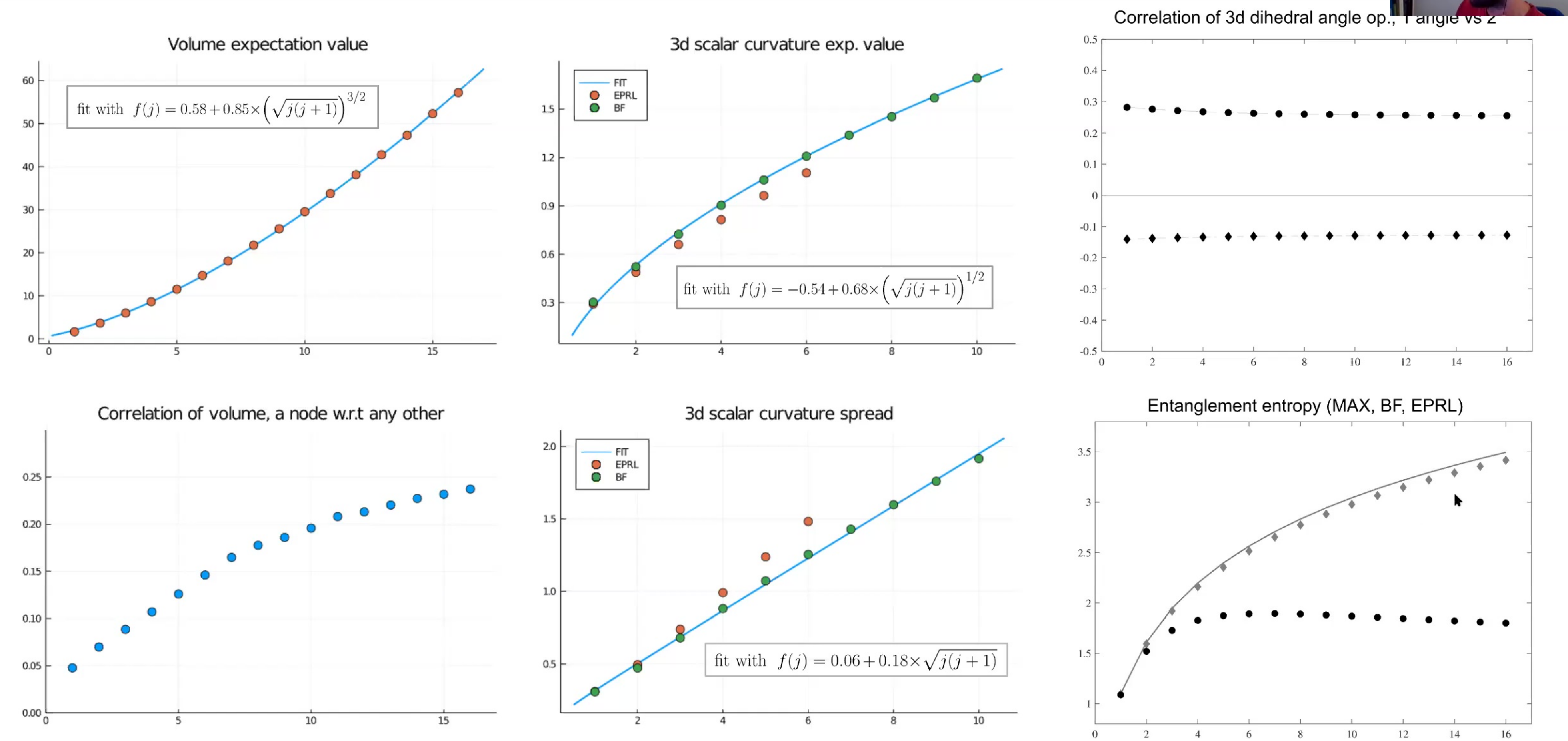
The amplitudes are computed for all boundary values. Results:
- all the **expectation values are compatible with regular tetrahedra** on the boundary of a regular geometrical 4-simplex
- **large fluctuations** (spread of operators)
- correlations do **not** vanish as $j \to \infty$
- results qualitatively similar if using BF or Lorentzian EPRL vertex (except for entanglement entropy)

[1] Bianchi (Nuclear Physics B, 2009) 0806.4710
[2] Alesci et al. (PRD, 2014) 1403.3190
[3] Bianchi et al. (PRD, 2019) 1812.10996





> application: spin cosmology



F. Gozzini - Advances in numerical techniques for spinfoam amplitudes with applications - 10/09/2020

> application: spinfoam cosmology

It is important to go beyond the single-vertex approximation:

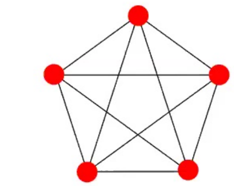
1. study correlations between **non-neighboring** tetrahedra
2. understand the **“flow”** of local observables
3. compute **bulk** observables

“Curse of dimensionality” problem: the values to be summed increase exponentially in the number of boundary tetrahedra. For *N* tetrahedra there are $(2j+1)^N$ possible intertwiners!

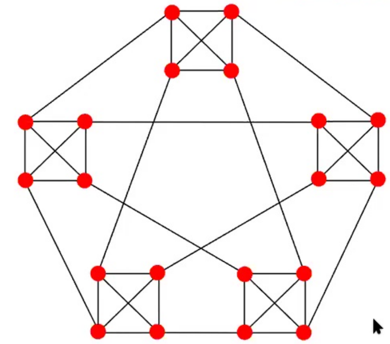
Solution: explore the intertwiner space using **Montecarlo sampling**. The density function is $\langle \psi_0 | \psi_0 \rangle$ (normalized) as in standard Quantum MC with the wavefunction.

$$\langle Q \rangle = \sum_{(i_a)} A^2_\Gamma(j, i_a) \langle i_a | Q | i_a \rangle \approx \sum_{\{i_a\}} \langle i_a | Q | i_a \rangle$$

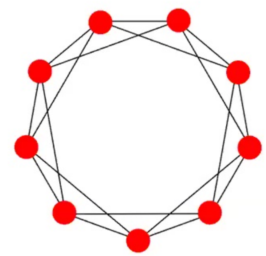
So far we considered graphs with 5, 9, 16, 20 boundary tetrahedra and up to 1 internal face.



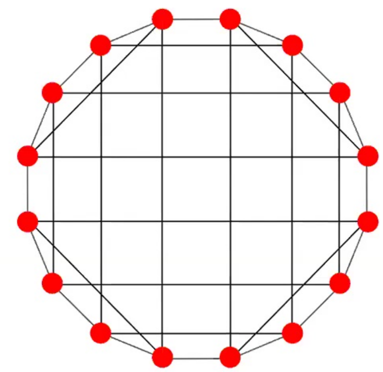
Single vertex



STAR (5 vertices)



Δ3
(3 vertices, 1 bulk face)

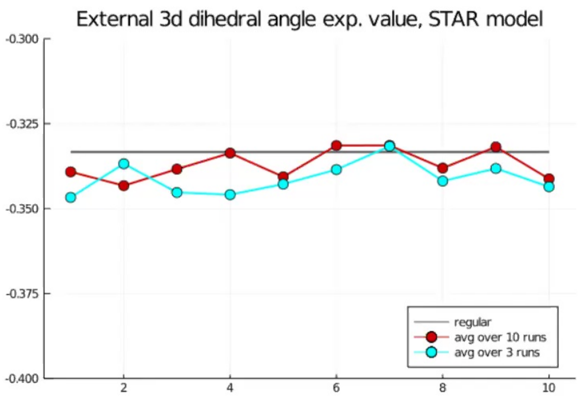


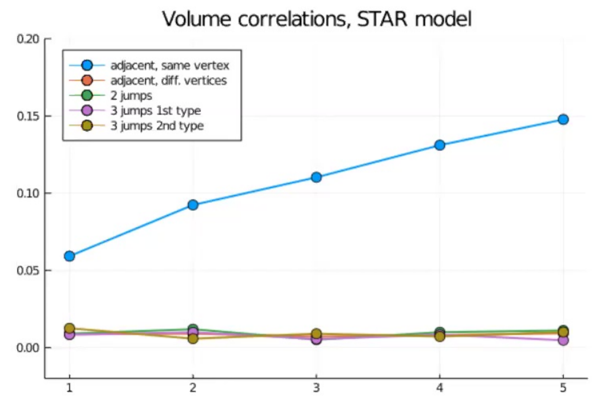
16-CELL (BF, no bulk)

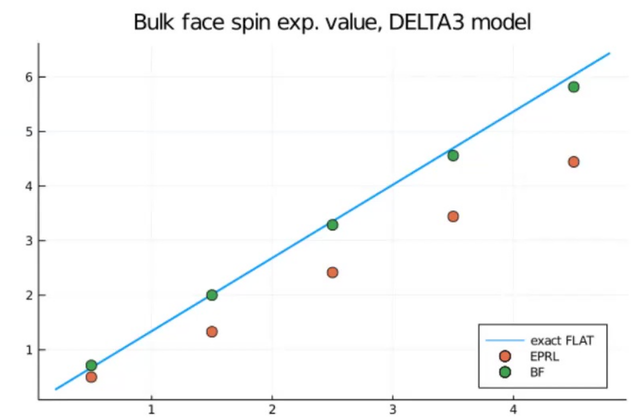
> application: spinfoam cosmology

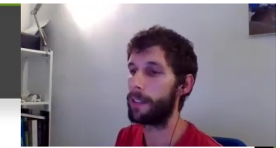
Results (*preliminary*):

- all the expectation values compatible with the classical corresponding geometry
- fluctuations are large
- correlations decrease sharply as “vertex-distance” increases
- boundary observables similar between BF and EPRL cases
- bulk observables depend crucially on the model: BF strongly differs from EPRL





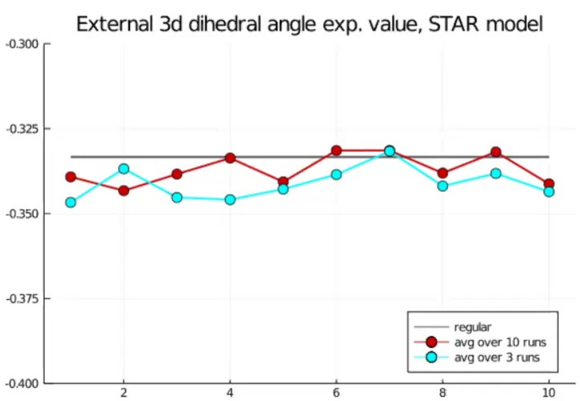


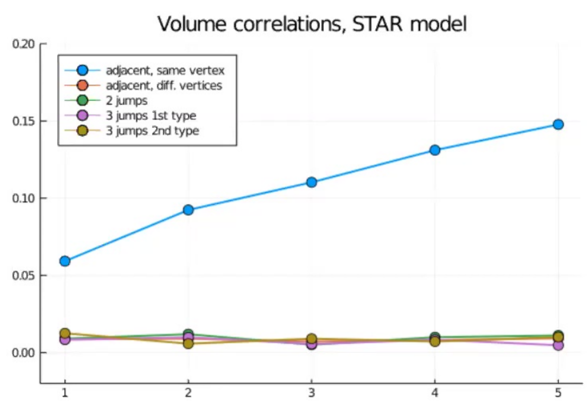


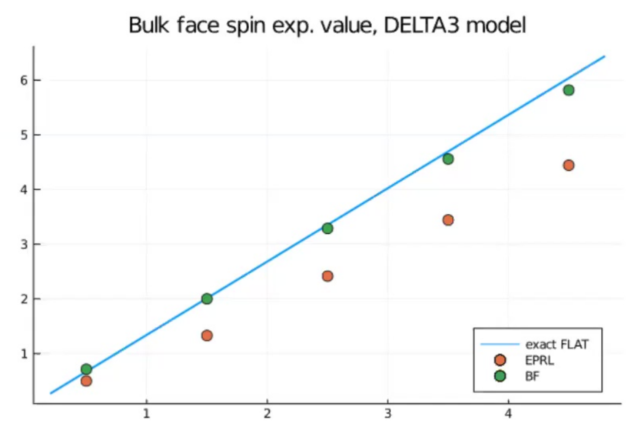
> application: spinfoam cosmology

Results (*preliminary*):

- all the expectation values compatible with the classical corresponding geometry
- fluctuations are large
- correlations decrease sharply as "vertex-distance" increases
- boundary observables similar between BF and EPRL cases
- bulk observables depend crucially on the model: BF strongly differs from EPRL







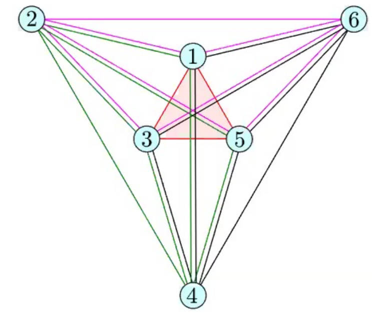
➔ Hints at the semiclassical regime with many vertices, locality, non-trivial dynamics...

➔ Can we deduce some experimentally observables consequences?

work in progress!

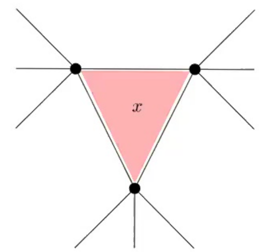
> application: flatness problem

In a recent work[1] we applied the library to study the **flatness problem** of spinfoam theories. This states that the semiclassical (large-j) limit of EPRL model is dominated by flat geometries, i.e. the solutions to the semiclassical e.o.m. that emerge from EPRL dynamics are all flat.



The simplest argument leading to this claim is based on the known single-vertex asymptotics (with coherent intertwiners) $A_v(j_f, \vec{n}_{fa}) \sim \cos \sum_f j_f \, \theta_f(\vec{n}_{fa})$.

Simple variation w.r.t the spins sets the deficit angle $\theta = 0$. Of course there are problems with this approach, in particular the spins are not independent on-shell of the constraints for which the asymptotic formula holds. There are other, different arguments but due to various technicalities the answer is not yet clear.



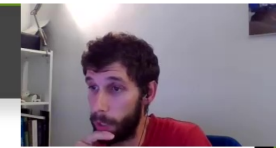
Crucially these arguments apply not only to the EPRL model but also to the simpler BF model.

➔ Can we **test these claims** with an *exact* calculation, without approximations?

We need a spinfoam graph with at least one internal (dynamical) face. The simplest one is the **$\Delta 3$** graph with 3 vertices, 15 boundary links, 9 boundary intertwiners.

[1] Donà, Gozzini, Sarno 2004.12911 (accepted for publication in PRD)

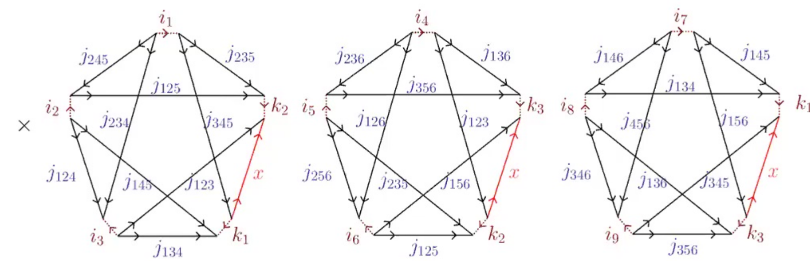
> application: flatness problem

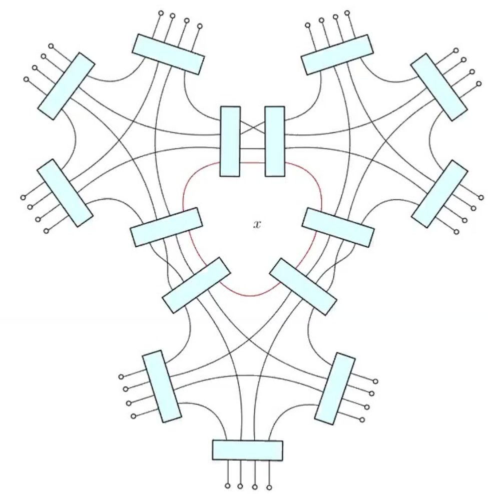


- We defined 3 different corresponding geometries, one with vanishing deficit angle and two with non-vanishing one.
- For each geometry we computed the coherent state coeff..
- We used the library to compute the amplitude → for each different boundary configuration.
  For each bulk face spin x :
  - compute all bulk intertwiner k_a
  - compute all { 15j }s for all k_a and boundary i_n
  - multiply them and sum over k_a
  - multiply with coherent states and some over all i_n

We had to search for the emergence of stationary phase points. Rescaling the boundary spins is too expensive (repeat the above steps for each value of the rescaling parameter).
We used an algorithm[1] that detects stationary phase points by looking at sudden peaks in the rolling average over the bulk variables (after filtering noise, amplifying etc...).

$$(-1)^X \sum_x (-1)^x (2x+1) \sum_{k_b} \left(\prod_b (-1)^{k_b}(2k_b+1)\right) \sum_{i_a} \left(\prod_a (2i_a+1) c_{i_a}(\vec{n}_{fa})\right)$$





[1] Donà, Gozzini, Sarno (Class. Quantum Grav., 2020) 1909.07832

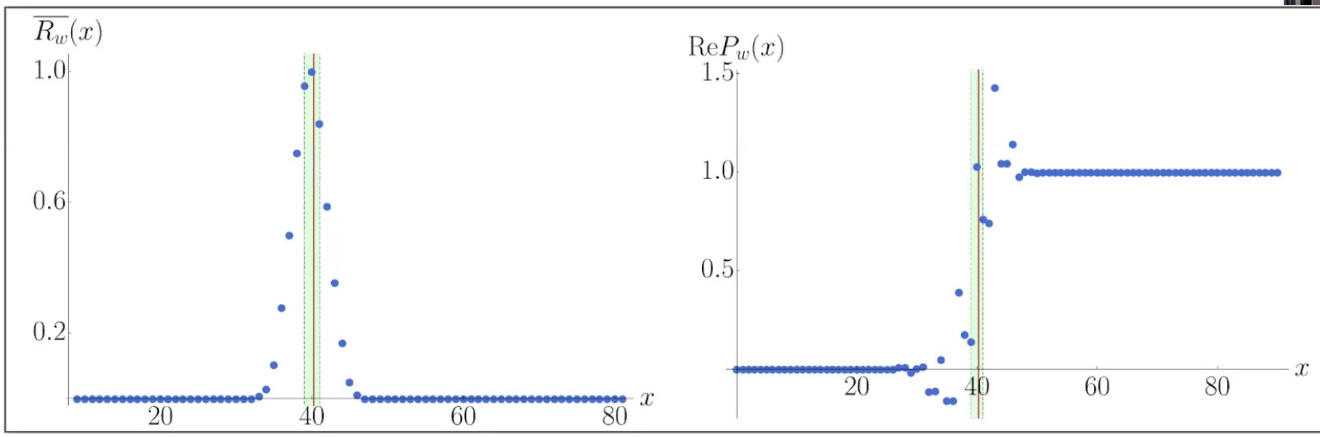
> application: flatness problem

Flat configuration

$\lambda = 30$

$x_{numerical} = 40 \pm 1$

$x_{analytical} = 40.2$

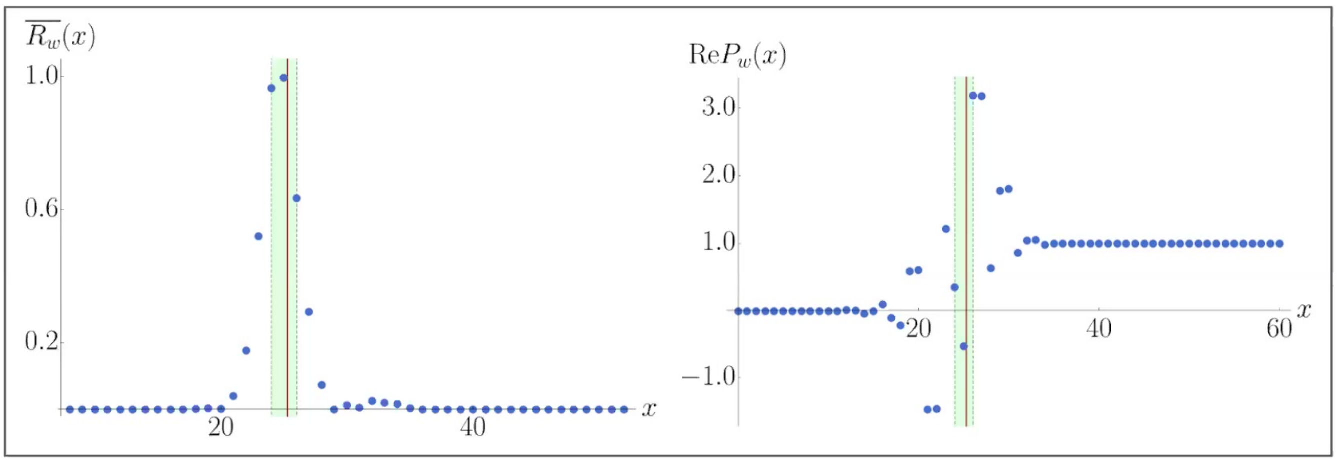


Curved configuration

$\lambda = 20$

$x_{numerical} = 25 \pm 1$

$x_{analytical} = 25.2$

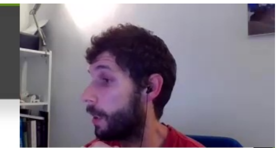


F. Gozzini - Advances in numerical techniques for spinfoam amplitudes with applications - 10/09/2020

Diapositiva 16

Google Slides

> application: flatness problem



Results: not only the flat configurations, but **also the curved configurations** have a stationary phase point, hence are not suppressed by the quantum BF dynamics.

There is a clear **tension** with the flatness claims. We suggest that the tension is solved if the angles that appear in the flatness arguments are **not to be interpreted as deficit angles of the spin connection**, i.e. as deficit angles à la Regge.

The result appears even more puzzling since classical BF theory *is* flat. What the numerics shows in this case is that the BF equations of motion impose flatness of the SU(2)-connection but do not require the deficit angles computed with an associated metric connection to vanish.

Our result provides an **explicit counterexample** to flatness claims.
But there is more work to be done:

- compute with the **EPRL vertex**
- compute with different boundary data, especially **Lorentzian boundary** (if any) with EPRL vertex
- consider a more complex graph with at least **one bulk segment** to compare to non-trivial Regge dynamics

	numerical	analytical	memory (GB)	time (h)
$\lambda = 10:$	13 ± 1	13.4	2	0.2
$\lambda = 12:$	16 ± 1	16.1	6	0.4
$\lambda = 14:$	18 ± 1	18.8	16	0.9
$\lambda = 16:$	21 ± 1	21.5	33	3
$\lambda = 18:$	23 ± 1	24.1	65	7.8
$\lambda = 20:$	26 ± 1	26.8	121	21
$\lambda = 30:$	40 ± 1	40.2	~ 2500	~ 2700

However, this first computation already required considerable time and resources...

> sl2cfoam-next

Better name needed :-)

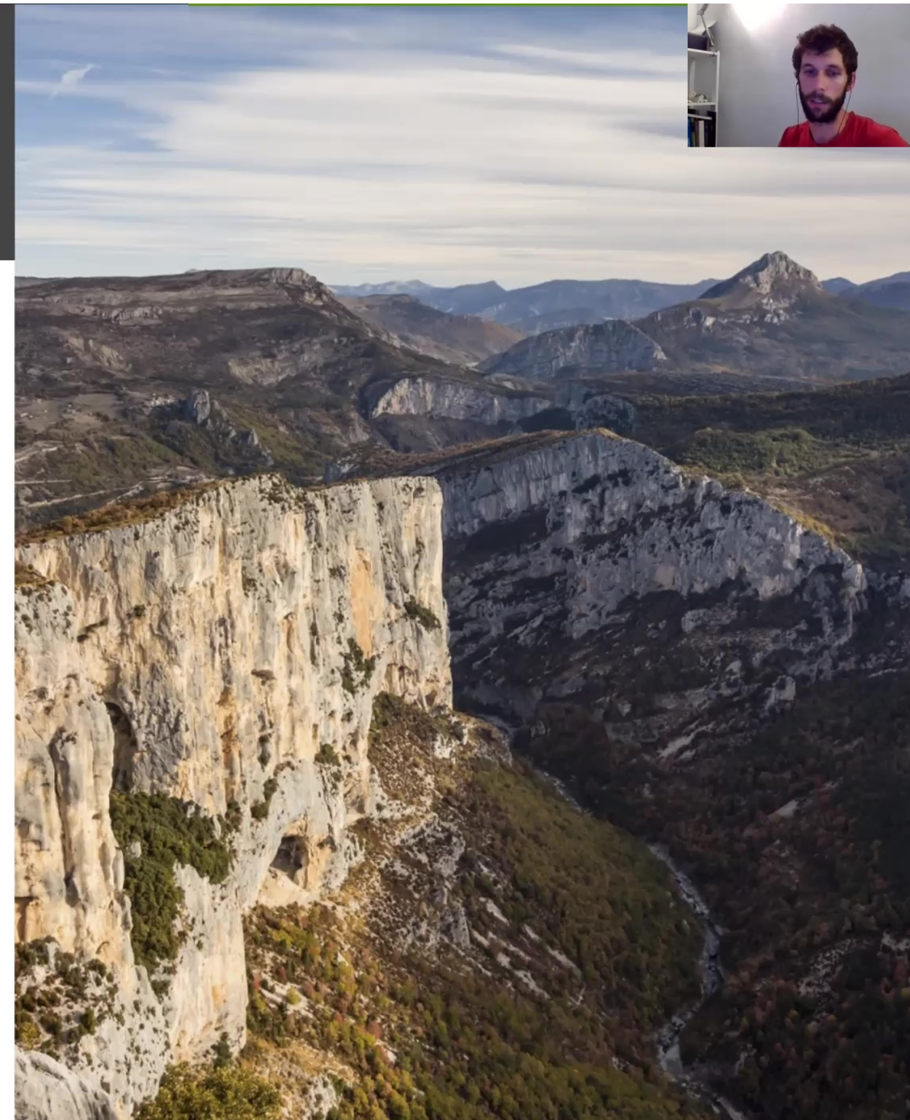
Many interesting computations require higher complexity:

- graphs with bulk segments
- physical models (black hole-to-white hole)
- refining & coarse-graining
- etc ...

It is possible to write much faster code using **high-performance computing** practices and techniques inspired by **tensor networks.**

Currently we are rewriting most of the code from scratch.
The new version will provide:

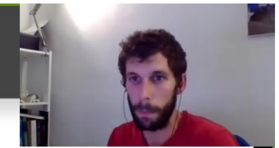
- much faster booster computation
- much faster vertex assembly
- composition of multiple vertices on the GPU



Diapositiva 18

Google Slides

> sl2cfoam-next: a preview



Booster coefficients:
- computation over multiple cluster nodes (MPI)
- more efficient arithmetic and smarter integration

Expected speedup: 100x and more

Single-vertex assembly:
- computation over multiple cluster nodes (MPI)
- sums as **tensor contractions** using loop-over-GEMM

Expected speedup: 1000x and more

$$A(i_1, i_2, i_3) = \sum_{k_1} \sum_{k_2} \sum_{k_3} B(i_1; k_1, k_2) B(i_2; k_2, k_3) B(i_3; k_3, k_1)$$

$$\downarrow$$

$$A^{i_1 i_2 i_3} = \operatorname{tr}\left[\left(B^{[i_1 k_1] k_2} \times (B^t)^{k_2 [i_2 k_3]}\right) \times B^{[i_3 k_3] k_1}\right]$$

Multiple-vertices:
- compose vertices as tensor contractions
- **GPU offloading** of contractions using CUDA
- tensor networks techniques? (SVD)
- Montecarlo sampling?

Expected speedup: 1000x and more

(speedup for *each* step)

```
> m = get_model("bh2wh"); a = amplitude(m); plot(a)
```

- Numerics is starting to play an important role in spinfoam theory
- There are now tools to study non-trivial problems
- Soon we will have more powerful tools to study more complex problems
- Future research directions:
  - settle the flatness problem
  - test the semiclassical dynamics of the EPRL model
  - spinfoam cosmology with many-vertices
  - study renormalization flow with exact vertex
  - think of physically relevant models and observables
  - ...

F. Gozzini - Advances in numerical techniques for spinfoam amplitudes with applications - 10/09/2020