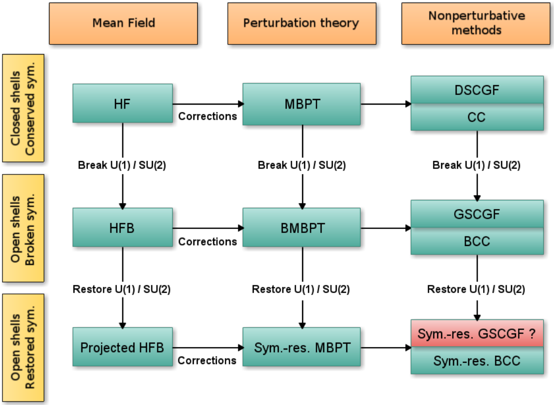
Spherical reduction of tensor networks in nuclear many-body theory: Automated symbolic angular-momentum algebra

Julien Ripoche
CEA, DAM, DIF, F-91297 Arpajon, France

with A. Tichai (ESNT, CEA Saclay) and T. Duguet (CEA Saclay).

Progress in Ab Initio Techniques in Nuclear Physics
TRIUMF Vancouver - February 28th 2019

Expansion methods



Courtesy of P. Arthuis, T. Duguet

New methods recently proposed and implemented

- BMBPT [Tichai *et al.* 2018]
- GSCGF, BCC [Somà *et al.* 2011, Signoracci *et al.* 2014]
- Sym.-res. BCC & sym.-res. BMBPT [Duguet 2015, Duguet *et al.* 2017, Qiu *et al.* 2017]

Push ab initio many-body calculations

Many-body calculations employ **mode-n tensors** and compute **tensor networks**

Input: Mode-n tensors

- Mode-2 tensor: t_{pq}, kinetic energy (Storage N^2)
- Mode-4 tensor: v_{pqrs}, two-body force (Storage N^4)
- Mode-6 tensor: w_{pqrstu}, three-body force (Storage N^6)

where $pq\ldots \in \mathcal{H}^{(1)}$ of dimension N.

Output: Tensor networks

$$E_0 = E_0^{HF} + \frac{1}{2}\sum_{ijab} \bar{v}_{ijab} t_i^a t_j^b + \frac{1}{4}\sum_{ijab} \bar{v}_{ijab} t_{ij}^{ab} \text{ (Evaluation } N^4\text{)}$$

Example from BCC at second order

Second-order triple amplitude in BCC theory

$$t^{60(2)}_{k_1 k_2 k_3 k_4 k_5 k_6} = P(k_1 k_2 k_3 / k_4 k_5 k_6) \sum_{k_7} \frac{\Omega^{31}_{k_1 k_2 k_3 k_7} t^{40(1)}_{k_7 k_4 k_5 k_6}}{E_{k_1 k_2 k_3 k_4 k_5 k_6}} \qquad \text{(20 terms)}$$

State-of-the-art calculations !

- Storage/handling of full **mode-6 tensors** impossible in $\mathcal{H}^{(1)}$ (N^6)
- Evaluation of **tensor networks** CPU intensive (N^7)

Need to **reduce the dimensionality** of the problem !

- Use of symmetry in **spherical basis**

From MScheme to JScheme

MScheme: Indices of tensors from spherical basis, $k = (n_k l_k j_k t_k m_k)$
(In $e_{max} = 12$, dimension $N = 1820$)

JScheme: Indices of tensors from coupled basis, $\tilde{k} = (n_k l_k j_k t_k)$
(In $e_{max} = 12$, effective dimension $\tilde{N} = 182 \ll N$)

MScheme tensors are expressed as a function of their **JScheme** equivalent:

$$t^{40(1)}_{k_1 k_2 k_3 k_4} = (-1)^{j_{k_3}+m_{k_3}+j_{k_4}+m_{k_4}} \sum_{JM} {}^{J}\tilde{t}^{40(1)}_{\tilde{k}_1 \tilde{k}_2 \tilde{k}_3 \tilde{k}_4} \begin{pmatrix} j_{k_1} & j_{k_2} & \big| & J \\ m_{k_1} & m_{k_2} & \big| & M \end{pmatrix} \begin{pmatrix} j_{k_3} & j_{k_4} & \big| & J \\ -m_{k_3} & -m_{k_4} & \big| & M \end{pmatrix}$$

$$\Omega^{31}_{k_1 k_2 k_3 k_4} = (-1)^{j_{k_3}+m_{k_3}} \sum_{JM} {}^{J}\tilde{\Omega}^{31}_{\tilde{k}_1 \tilde{k}_2 \tilde{k}_3 \tilde{k}_4} \begin{pmatrix} j_{k_1} & j_{k_2} & \big| & J \\ m_{k_1} & m_{k_2} & \big| & M \end{pmatrix} \begin{pmatrix} j_{k_3} & j_{k_4} & \big| & J \\ -m_{k_3} & m_{k_4} & \big| & M \end{pmatrix}$$

$$\begin{aligned} {}^{j}\tilde{t}^{60(2)}_{\tilde{k}_1 \tilde{k}_2 J_{12} \tilde{k}_3 ; \tilde{k}_4 \tilde{k}_5 J_{45} \tilde{k}_6} = & \sum_{m_{k_1} \dots m_{k_6} m M_{12} M_{45}} (-1)^{j_{k_4}+m_{k_4}+j_{k_5}+m_{k_5}+j_{k_6}+m_{k_6}} t^{60(2)}_{k_1 k_2 k_3 k_4 k_5 k_6} \\ & \times \begin{pmatrix} j_{k_1} & j_{k_2} & \big| & J_{12} \\ m_{k_1} & m_{k_2} & \big| & M_{12} \end{pmatrix} \begin{pmatrix} J_{12} & j_{k_3} & \big| & j \\ M_{12} & m_{k_3} & \big| & m \end{pmatrix} \begin{pmatrix} j_{k_4} & j_{k_5} & \big| & J_{45} \\ -m_{k_4} & -m_{k_5} & \big| & M_{45} \end{pmatrix} \begin{pmatrix} J_{45} & j_{k_6} & \big| & j \\ M_{45} & -m_{k_6} & \big| & m \end{pmatrix} \end{aligned}$$

Example from BCC at second order

Second-order triple amplitude in BCC theory (Third term)

$$\left(t^{60(2)}_{k_1 k_2 k_3 k_4 k_5 k_6}\right)_3 = -\sum_{k_7} \frac{\Omega^{31}_{k_3 k_1 k_4 k_7} t^{40(1)}_{k_2 k_7 k_6 k_5}}{E_{k_1 k_2 k_3 k_4 k_5 k_6}}$$

Resulting **summation over magnetic quantum numbers**:

$$\sum_{m_1 m_2 m_3 m_4 m_5 m_6 m_7 m M_{12} M_{45} M^{(1)} M^{(2)}} \hat{J_{12}}\, \hat{J_{45}}\, \hat{j}^2\, \hat{J^{(1)}}^2\, \hat{J^{(2)}}^2$$

$$\begin{pmatrix} j_1 & j_2 & J_{12} \\ m_1 & m_2 & -M_{12} \end{pmatrix} \begin{pmatrix} J_{12} & j_3 & j \\ M_{12} & m_3 & -m \end{pmatrix} \begin{pmatrix} j_4 & j_5 & J_{45} \\ m_4 & m_5 & M_{45} \end{pmatrix} \begin{pmatrix} J_{45} & j_6 & j \\ -M_{45} & m_6 & m \end{pmatrix}$$

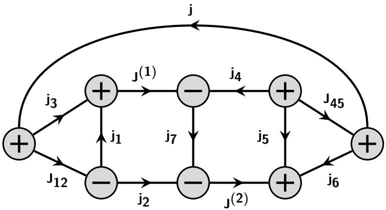
$$\begin{pmatrix} j_3 & j_1 & J^{(1)} \\ -m_3 & -m_1 & M^{(1)} \end{pmatrix} \begin{pmatrix} j_4 & j_7 & J^{(1)} \\ -m_4 & m_7 & -M^{(1)} \end{pmatrix} \begin{pmatrix} j_2 & j_7 & J^{(2)} \\ -m_2 & -m_7 & M^{(2)} \end{pmatrix} \begin{pmatrix} j_6 & j_5 & J^{(2)} \\ -m_6 & -m_5 & -M^{(2)} \end{pmatrix}$$

Representation of CG networks as Yutsis Graph

Definition of a Yutsis Graph

- It is a **cubic** graph, i.e. each **node** has three **edges**.
- Each **node** representes a **3-j symbol**.
- Each **edge** representes **angular-momentum indices** (j,m).
- Node **sign** is $+$ if indices are in **trigonometric** order else $-$.
- Edge **orientation** goes from **positive** projection to **negative** one.

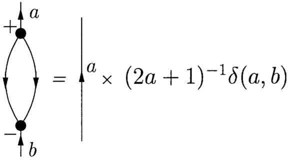
$$\sum_{m_1 m_2 m_3 m_4 m_5 m_6 m_7 m M_{12} M_{45} M^{(1)} M^{(2)}} \begin{pmatrix} j_1 & j_2 & J_{12} \\ m_1 & m_2 & -M_{12} \end{pmatrix} \begin{pmatrix} J_{12} & j_3 & j \\ M_{12} & m_3 & -m \end{pmatrix} \begin{pmatrix} j_4 & j_5 & J_{45} \\ m_4 & m_5 & M_{45} \end{pmatrix} \begin{pmatrix} J_{45} & j_6 & j \\ -M_{45} & m_6 & m \end{pmatrix} \begin{pmatrix} j_3 & j_1 & J^{(1)} \\ -m_3 & -m_1 & M^{(1)} \end{pmatrix} \begin{pmatrix} j_4 & j_7 & J^{(1)} \\ -m_4 & m_7 & -M^{(1)} \end{pmatrix} \begin{pmatrix} j_2 & j_7 & J^{(2)} \\ -m_2 & -m_7 & M^{(2)} \end{pmatrix} \begin{pmatrix} j_6 & j_5 & J^{(2)} \\ -m_6 & -m_5 & -M^{(2)} \end{pmatrix} \longrightarrow$$



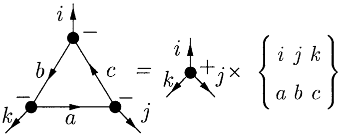
[JR, A. Tichai, in prep.]

Reduction rules

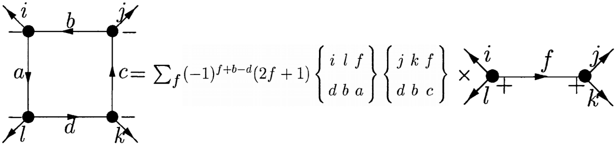
Reduction of a 2-cycle



Reduction of a 3-cycle



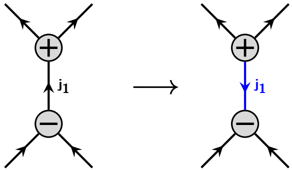
Reduction of a 4-cycle



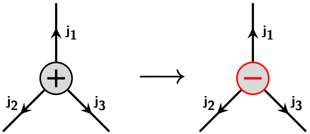
[D. Van Dyck, V. Fack, Discrete Mathematics]

Additional rules

Line reversal of internal line



Direct change of Node Sign



Phase associated to **Line reversal** of an internal line:

- $\phi_{lr} = (-1)^{2j}$

Phase associated to direct change of **Node Sign**:

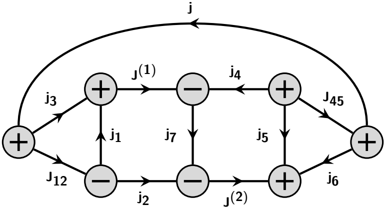
- $\phi_{ns} = (-1)^{j_1+j_2+j_3}$

Algorithm for Yutsis Graph reduction

ReduceYutsisGraph(**YutsisGraph** Y)

```
while Y != triangular delta do
  if Y has bubble then
    Format and remove arbitrary bubble
  else if Y has triangle then
    Format and remove arbitrary triangle
  else if Y has square then
    Format and remove arbitrary square
  else
    //Implement higher-order rules
  end if
end while
return formula
```

Reduction of a Yutsis Graph



$$\longrightarrow \quad -(-1)^{j_1+j_2+J_{12}} \begin{Bmatrix} j & J^{(1)} & j_2 \\ j_1 & J_{12} & j_3 \end{Bmatrix} \begin{Bmatrix} j & j_4 & J^{(2)} \\ j_7 & j_2 & J^{(1)} \end{Bmatrix} \begin{Bmatrix} j & J_{45} & j_6 \\ j_5 & J^{(2)} & j_4 \end{Bmatrix}$$

All in all this gives the **J-coupled** version of second-order triple amplitude:

$$\left({}^{j}\tilde{t}^{60(2)}_{\tilde{k}_1\tilde{k}_2 J_{12}\tilde{k}_3;\tilde{k}_4\tilde{k}_5 J_{45}\tilde{k}_6}\right)_3 = (-1)^{j_1+j_2+J_{12}} \hat{J}_{12}\hat{J}_{45} \sum_{\tilde{k}_7 J^{(1)} J^{(2)}} \frac{{}^{J^{(1)}}\tilde{\Omega}^{31}_{\tilde{k}_3\tilde{k}_1\tilde{k}_4\tilde{k}_7} \; {}^{J^{(2)}}\tilde{t}^{40(1)}_{\tilde{k}_2\tilde{k}_7\tilde{k}_6\tilde{k}_5}}{E_{\tilde{k}_1\tilde{k}_2\tilde{k}_3\tilde{k}_4\tilde{k}_5\tilde{k}_6}}$$

$$\times \hat{J^{(1)}}^2 \hat{J^{(2)}}^2 \begin{Bmatrix} j & J^{(1)} & j_2 \\ j_1 & J_{12} & j_3 \end{Bmatrix} \begin{Bmatrix} j & j_4 & J^{(2)} \\ j_7 & j_2 & J^{(1)} \end{Bmatrix} \begin{Bmatrix} j & J_{45} & j_6 \\ j_5 & J^{(2)} & j_4 \end{Bmatrix}$$

Python implementation of the full process

The full process has been **implemented in Python** !

Firsts results for second-order triple amplitude:

$$
\begin{aligned}
&{}^{J}\tilde{T}^{3}_{\tilde{k}_1\tilde{k}_2 J_{12}\tilde{k}_3;\tilde{k}_4\tilde{k}_5 J_{45}\tilde{k}_6} \\
&= \sum_{n_{k_7}} \hat{j_{k_7}}^{-2}\, \hat{J_{12}^{(0)}}\, \hat{J_{45}^{(0)}}\, {}^{J_{12}^{(0)}}\tilde{H}^{31}_{\tilde{k}_1\tilde{k}_2\tilde{k}_3 n_{k_7}(ljt)_{J^{(0)}}}\, {}^{J_{45}^{(0)}}\tilde{T}^{2}_{\tilde{k}_6 n_{k_7}(ljt)_{J^{(0)}}\tilde{k}_4\tilde{k}_5} \\
&+ \sum_{\tilde{k}_7 J^{(2)}} (-1)^{j_{k_4}+j_{k_5}+J_{45}^{(0)}}\, \hat{J_{12}^{(0)}}\, \hat{J_{45}^{(0)}}\, \hat{J^{(2)}}^{2}\, {}^{J_{12}^{(0)}}\tilde{H}^{31}_{\tilde{k}_1\tilde{k}_2\tilde{k}_5\tilde{k}_7}\, {}^{J^{(2)}}\tilde{T}^{2}_{\tilde{k}_3\tilde{k}_7\tilde{k}_6\tilde{k}_4} \begin{Bmatrix} j_{k_3} & j_{k_7} & J^{(2)} \\ j_{k_5} & J^{(0)} & J_{12}^{(0)} \end{Bmatrix} \begin{Bmatrix} j_{k_6} & j_{k_4} & J^{(2)} \\ j_{k_5} & J^{(0)} & J_{45}^{(0)} \end{Bmatrix} \\
&+ \sum_{\tilde{k}_7 J^{(1)} J^{(2)}} (-1)^{j_{k_1}+j_{k_2}+J_{12}^{(0)}}\, \hat{J_{12}^{(0)}}\, \hat{J_{45}^{(0)}}\, \hat{J^{(1)}}^{2}\, \hat{J^{(2)}}^{2}\, {}^{J^{(1)}}\tilde{H}^{31}_{\tilde{k}_3\tilde{k}_1\tilde{k}_4\tilde{k}_7}\, {}^{J^{(2)}}\tilde{T}^{2}_{\tilde{k}_2\tilde{k}_7\tilde{k}_6\tilde{k}_5} \begin{Bmatrix} j_{k_3} & j_{k_1} & J^{(1)} \\ j_{k_2} & J^{(0)} & J_{12}^{(0)} \end{Bmatrix} \begin{Bmatrix} j_{k_2} & j_{k_7} & J^{(2)} \\ j_{k_4} & J^{(0)} & J^{(1)} \end{Bmatrix} \begin{Bmatrix} j_{k_6} & j_{k_5} & J^{(2)} \\ j_{k_4} & J^{(0)} & J_{45}^{(0)} \end{Bmatrix} \\
&+ \dots
\end{aligned}
$$

Conclusions

Spherical reduction of tensor networks:

- **Dimensionality reduction** going from MScheme to JScheme
- By hand **derivation tidious** in advanced formalisms
- Need for an **automatic tool**

Python implementation has already been used for:

- Normal ordering
- (P)BMBPT
- (P)BCC

Ongoing projects:

- Particle-number conserving normal ordering
- Shift invariance for PBMBPT
- ODE version of PBMBPT

My poster

Spherical reduction of tensor networks in nuclear many-body theory: Automated symbolic angular-momentum algebra

Julien Ripoche[1*], Alexander Tichai[2] and Thomas Duguet[3,4]

[1] CEA, DAM, DIF, F-91297 Arpajon, France
[2] ESNT, CEA-Saclay, DRF/Irfu/DPhN, F-91191 Gif-sur-Yvette Cedex, France
[3] CEA-Saclay, DRF/Irfu/DPhN, F-91191 Gif-sur-Yvette Cedex, France
[4] KU Leuven, Instituut voor Kern- en Stralingsfysica, 3001 Leuven, Belgium

Purpose of this work

- Push *ab initio* many-body calculations based on expansion methods:
  - developing new formalisms (PBMBPT,PBCC),
  - including many-body forces,
  - aiming for a higher accuracy.
- Automatize the spherical reduction of tensor networks:
  - developing a symbolic code for MScheme to JScheme reduction,
  - that operates over Clebsch-Gordan (CG) coefficients,
  - via a graphical representation of CG coefficients and reduction rules.

Tensors and tensor networks

Input: Mode-n tensors

- Mode-2 tensor: t_{pq}, kinetic energy (Storage N^2)
- Mode-4 tensor: v_{pqrs}, two-body force (Storage N^4)
- Mode-6 tensor: w_{pqrstu}, three-body force (Storage N^6)

where $pq\ldots \in \mathcal{H}^{(1)}$ of dimension N.

Output: Tensor networks

Ex: $E_0 = E_0^{HF} + \frac{1}{2}\sum_{ijab} \bar{v}_{ijab} t_i^a t_j^b + \frac{1}{4}\sum_{ijab} \bar{v}_{ijab} t_{ij}^{ab}$ (Evaluation N^4)

- Storage/handling of full mode-6 tensors impossible in $\mathcal{H}^1$ (N^6)
- Evaluation of tensor networks CPU intensive (N^7)
- Need to reduce the dimensionality of the problem !

Symmetry reduction from MScheme to JScheme

- **MScheme:** Indices of tensors from spherical basis, $k = (n_k l_k j_k t_k m_k)$ (In $e_{max} = 12$, dimension $N = 1820$)
- **JScheme:** Indices of tensors from coupled basis, $\tilde{k} = (n_k l_k j_k t_k)$ (In $e_{max} = 12$, effective dimension $\tilde{N} = 182 \ll N$)
- **MScheme** tensors are expressed as a function of their **JScheme** equivalents:

$$t^{40(1)}_{k_1k_2k_3k_4} = (-1)^{j_{k_3}+m_{k_3}+j_{k_4}+m_{k_4}} \sum_{JM} {}^{J}\tilde{t}^{40(1)}_{\tilde{k}_1\tilde{k}_2\tilde{k}_3\tilde{k}_4} \begin{pmatrix} j_{k_1} & j_{k_2} & J \\ m_{k_1} & m_{k_2} & M \end{pmatrix} \begin{pmatrix} j_{k_3} & j_{k_4} & J \\ -m_{k_3} & -m_{k_4} & M \end{pmatrix}$$

...

Ex: Second-order triple amplitude in BCC theory

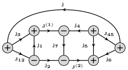
$$\left(t^{60(2)}_{k_1k_2k_3k_4k_5k_6}\right)_3 = -\sum_{k_7} \frac{\Omega^{31}_{k_3k_1k_4k_7} t^{40(1)}_{k_2k_7k_6k_5}}{E_{k_1k_2k_3k_4k_5k_6}} \quad \text{(1 of 20 terms)}$$

$$\longrightarrow \sum_{m_1m_2m_3m_4m_5m_6m_7mM_{12}M_{45}M^{(1)}M^{(2)}} \begin{pmatrix} j_1 & j_2 & J_{12} \\ m_1 & m_2 & -M_{12} \end{pmatrix} \begin{pmatrix} J_{12} & j_3 & j \\ M_{12} & m_3 & -m \end{pmatrix} \begin{pmatrix} j_4 & j_5 & J_{45} \\ m_4 & m_5 & M_{45} \end{pmatrix} \begin{pmatrix} J_{45} & j_6 & j \\ -M_{45} & m_6 & m \end{pmatrix} \begin{pmatrix} j_3 & j_1 & J^{(1)} \\ -m_3 & -m_1 & M^{(1)} \end{pmatrix} \begin{pmatrix} j_4 & j_7 & J^{(1)} \\ -m_4 & m_7 & -M^{(1)} \end{pmatrix} \begin{pmatrix} j_2 & j_7 & J^{(2)} \\ -m_2 & -m_7 & M^{(2)} \end{pmatrix} \begin{pmatrix} j_6 & j_5 & J^{(2)} \\ -m_6 & -m_5 & -M^{(2)} \end{pmatrix}$$

Representation of CG networks as a Yutsis Graph

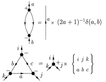
Definition of a Yutsis Graph

- It is a cubic graph, i.e. each node has three edges.
- Each node representes a 3-j symbol.
- Each edge representes angular-momentum indices (j,m).
- Node sign is + if indices are in trigonometric order else −.
- Edge orientation goes from positive projection to negative one.



Algorithm for Yutsis Graph reduction

Simplest reduction rules



ReduceYutsisGraph(**YutsisGraph** Y)

```
while Y != triangular delta do
  if Y has bubble then
    Format and remove arbitrary bubble
  else if Y has triangle then
    Format and remove arbitrary triangle
  else if Y has square then
    Format and remove arbitrary square
  else
    //Implement higher-order rules
  end if
end while
return formula
```

J-coupled version of second-order triple amplitude

- Reduce the Yutsis Graph associated to the expression
- Put back resulting factors into JScheme expression

$$\left({}^{j}\tilde{t}^{60(2)}_{\tilde{k}_1\tilde{k}_2J_{12}\tilde{k}_3;\tilde{k}_4\tilde{k}_5J_{45}\tilde{k}_6}\right)_3 = -(-1)^{j_1+j_2+J_{12}+1} \hat{J}_{12}\hat{J}_{45}\hat{J}^{(1)2}\hat{J}^{(2)2} \times \sum_{\tilde{k}_7 J^{(1)} J^{(2)}} \frac{{}^{J^{(1)}}\tilde{\Omega}^{31}_{\tilde{k}_3\tilde{k}_1\tilde{k}_4\tilde{k}_7}\, {}^{J^{(2)}}\tilde{t}^{40(1)}_{\tilde{k}_2\tilde{k}_7\tilde{k}_6\tilde{k}_5}}{E_{\tilde{k}_1\tilde{k}_2\tilde{k}_3\tilde{k}_4\tilde{k}_5\tilde{k}_6}} \times \begin{Bmatrix} j & J^{(1)} & j_2 \\ j_1 & J_{12} & j_3 \end{Bmatrix} \begin{Bmatrix} j & j_4 & J^{(2)} \\ j_7 & j_2 & J^{(1)} \end{Bmatrix} \begin{Bmatrix} j & J_{45} & j_6 \\ j_5 & J^{(2)} & j_4 \end{Bmatrix}$$

Summary

- State-of-the-art many-body methods require large storage capacities and are CPU intensives.
- Symmetry reduction: Reduction of the dimensionality going from MScheme to JScheme.
- Python code written for automated symbolic manipulation of Yutsis Graph.
- Application to more and more complex tensor networks.
- The code has already been used for: Normal ordering, (P)BMBPT, (P)BCC.

References

[1] J. Ripoche, A. Tichai, in prep.
[2] D. Van Dyck, V. Fack, *New heuristic approach to the calculation of general recoupling coefficients*, Comput. Phys. Comm. **151**, 354-368 (2003).
[3] E. Balcar, S. W. Lovesey, *Introduction to the Graphical Theory of Angular Momentum*, Springer-Verlag (2009).

* e-mail : julien.ripoche@cea.fr

Progress in *Ab Initio* Techniques in Nuclear Physics - Feb 26-Mar 1 2019 - TRIUMF, Vancouver BC, Canada