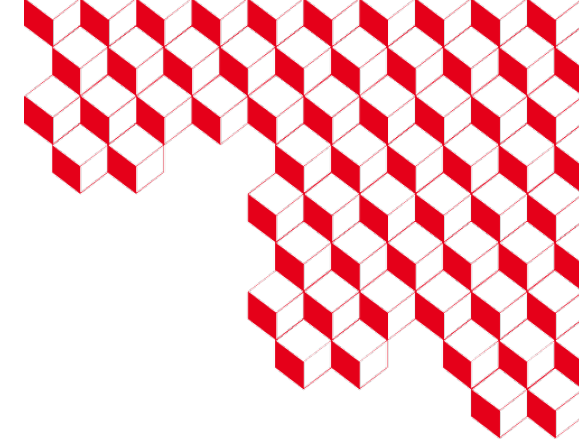




Microscopic description of the nuclear fission process

J. Newsome - EUNPC Sep. 2025

Supervisors: N. Dubray (CEA), N. Pillet (CEA)

Context

What is the TDGCM approach ?

1. **Static:** Generate a set of deformed **HFB** states

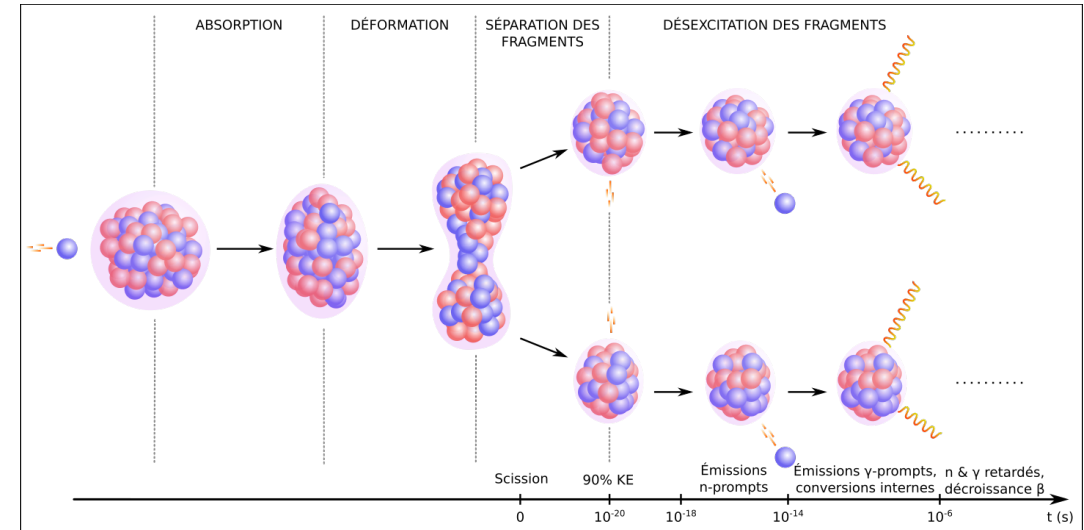
$$|\phi(q)\rangle = \prod_i \sum_j [U_{ji}(q)c_j + V_{ji}(q)c_j^\dagger] |-\rangle \quad \forall q \in Q$$

2. **Dynamic:** Solve the equation of evolution of the **TDGCM** wavefunction at the **GOA** approximation

$$|\psi(t)\rangle = \int_q f(q, t) |\phi(q)\rangle$$

What observables can we predict ?

- Fragment properties
- Mass and Charge yields
- TKE



Context

What is the TDGCM approach ?

1. **Static:** Generate a set of deformed **HFB** states

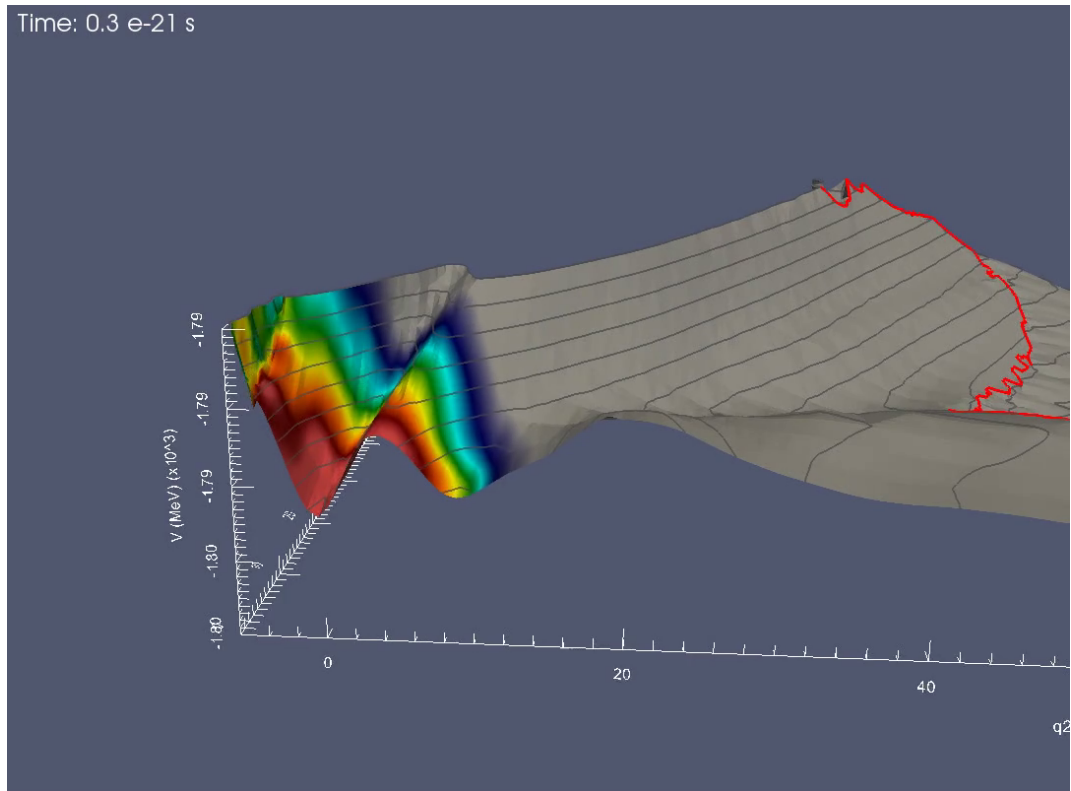
$$|\phi(q)\rangle = \prod_i \sum_j [U_{ji}(q)c_j + V_{ji}(q)c_j^\dagger] |-\rangle \quad \forall q \in Q$$

2. **Dynamic:** Solve the equation of evolution of the **TDGCM** wavefunction at the **GOA** approximation

$$|\psi(t)\rangle = \int_q f(q, t) |\phi(q)\rangle$$

What observables can we predict ?

- Fragment properties
- Mass and Charge yields
- TKE



Dynamic propagation using FELIX code - Courtesy of D. Regnier (CEA)

Context

What is the TDGCM approach ?

1. **Static:** Generate a set of deformed **HFB** states

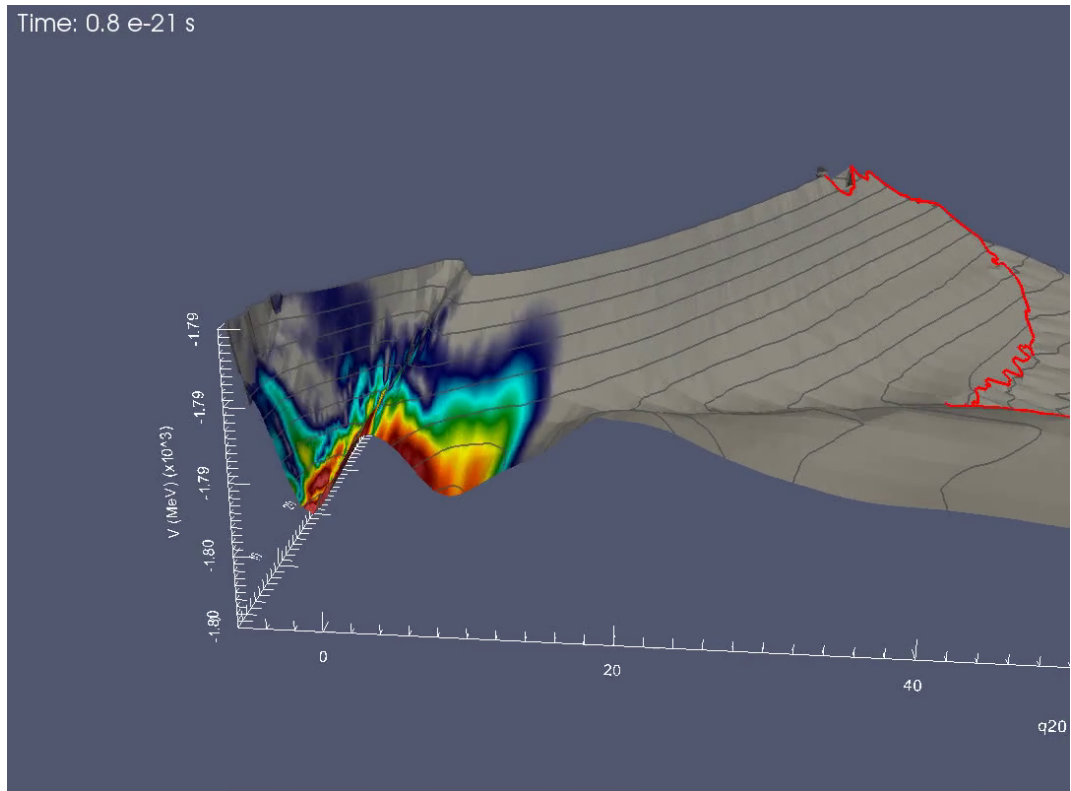
$$|\phi(q)\rangle = \prod_i \sum_j [U_{ji}(q)c_j + V_{ji}(q)c_j^\dagger] |-\rangle \quad \forall q \in Q$$

2. **Dynamic:** Solve the equation of evolution of the **TDGCM** wavefunction at the **GOA** approximation

$$|\psi(t)\rangle = \int_q f(q, t) |\phi(q)\rangle$$

What observables can we predict ?

- Fragment properties
- Mass and Charge yields
- TKE



Dynamic propagation using FELIX code - Courtesy of D. Regnier (CEA)

Context

What is the TDGCM approach ?

1. **Static:** Generate a set of deformed **HFB** states

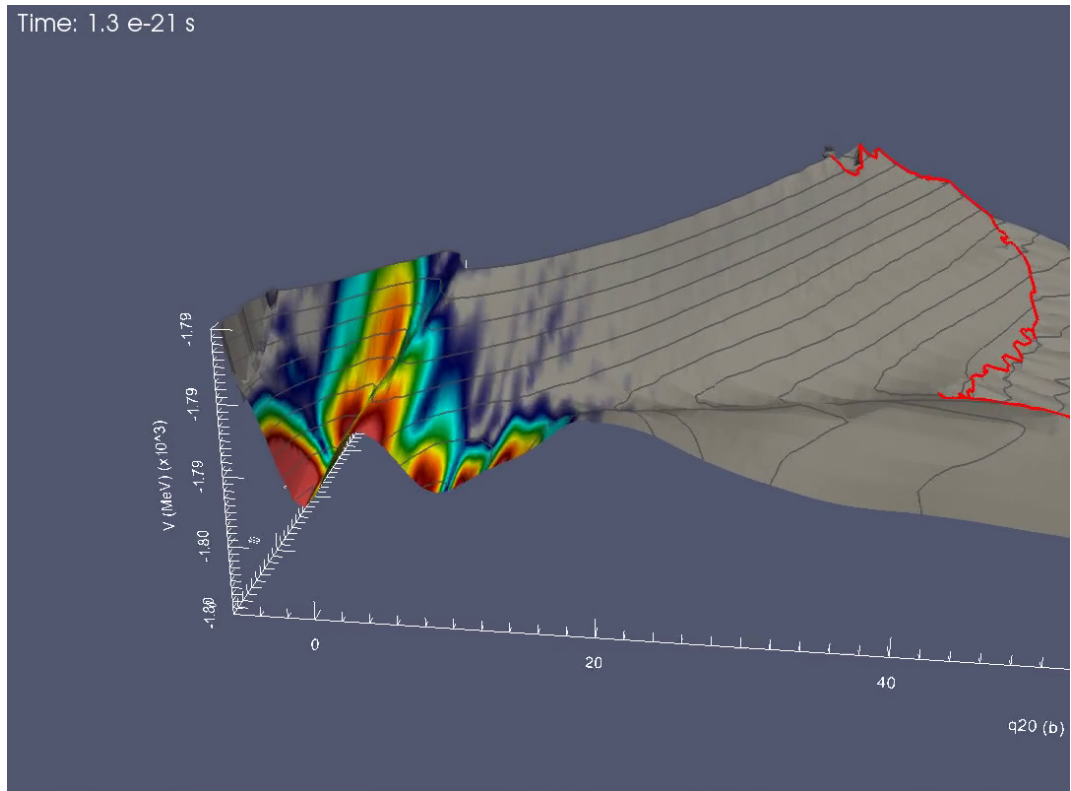
$$|\phi(q)\rangle = \prod_i \sum_j [U_{ji}(q)c_j + V_{ji}(q)c_j^\dagger] |-\rangle \quad \forall q \in Q$$

2. **Dynamic:** Solve the equation of evolution of the **TDGCM** wavefunction at the **GOA** approximation

$$|\psi(t)\rangle = \int_q f(q, t) |\phi(q)\rangle$$

What observables can we predict ?

- Fragment properties
- Mass and Charge yields
- TKE



Dynamic propagation using FELIX code - Courtesy of D. Regnier (CEA)

Context

What is the TDGCM approach ?

1. **Static:** Generate a set of deformed **HFB** states

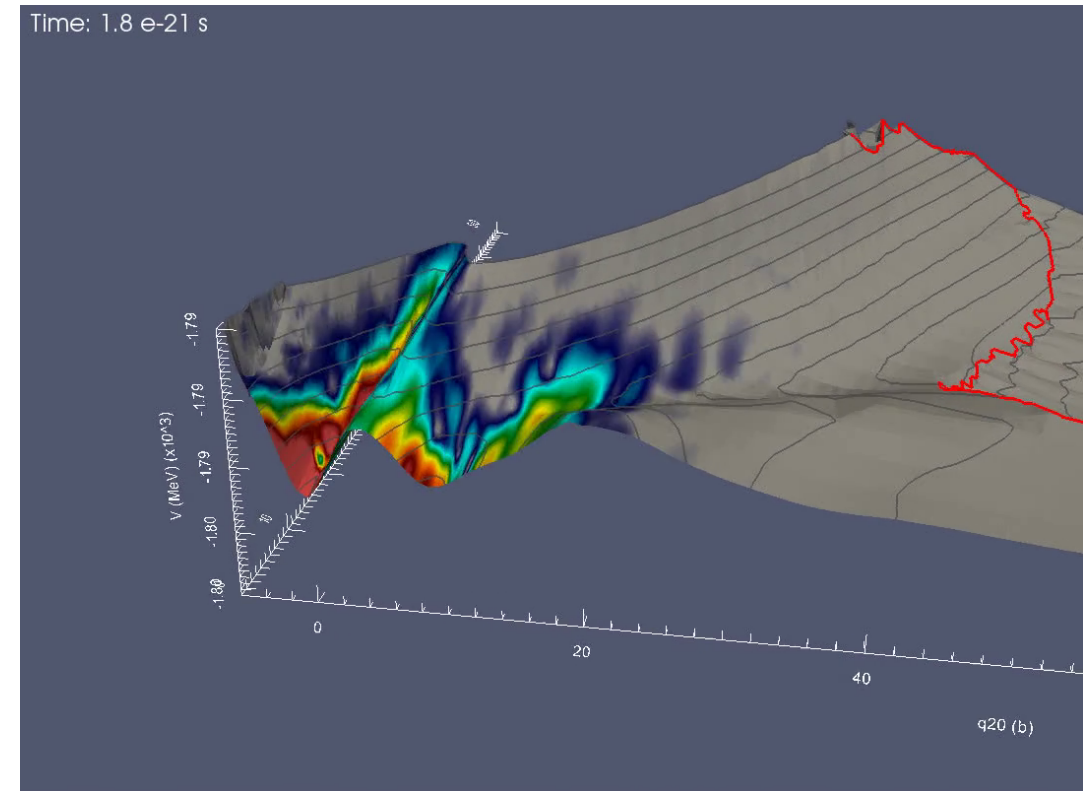
$$|\phi(q)\rangle = \prod_i \sum_j [U_{ji}(q)c_j + V_{ji}(q)c_j^\dagger] |-\rangle \quad \forall q \in Q$$

2. **Dynamic:** Solve the equation of evolution of the **TDGCM** wavefunction at the **GOA** approximation

$$|\psi(t)\rangle = \int_q f(q, t) |\phi(q)\rangle$$

What observables can we predict ?

- Fragment properties
- Mass and Charge yields
- TKE



Dynamic propagation using FELIX code - Courtesy of D. Regnier (CEA)

Context

What is the TDGCM approach ?

1. **Static:** Generate a set of deformed **HFB** states

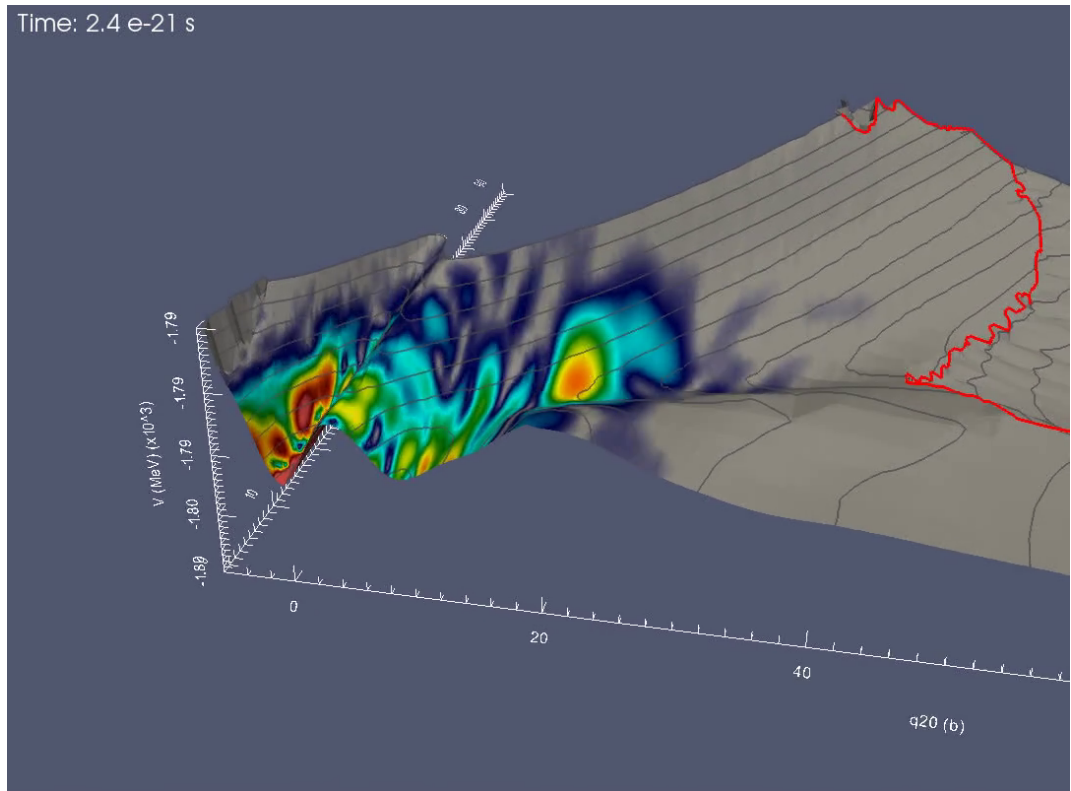
$$|\phi(q)\rangle = \prod_i \sum_j [U_{ji}(q)c_j + V_{ji}(q)c_j^\dagger] |-\rangle \quad \forall q \in Q$$

2. **Dynamic:** Solve the equation of evolution of the **TDGCM** wavefunction at the **GOA** approximation

$$|\psi(t)\rangle = \int_q f(q, t) |\phi(q)\rangle$$

What observables can we predict ?

- Fragment properties
- Mass and Charge yields
- TKE



Dynamic propagation using FELIX code - Courtesy of D. Regnier (CEA)

Context

What is the TDGCM approach ?

1. **Static:** Generate a set of deformed **HFB** states

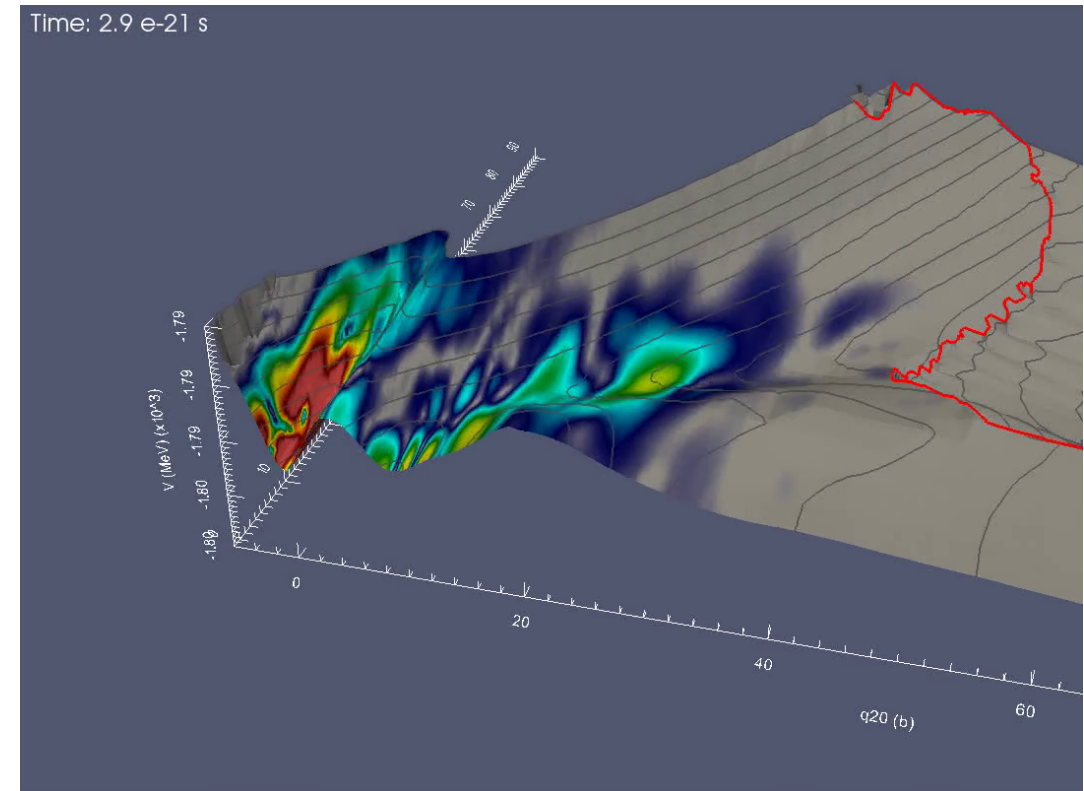
$$|\phi(q)\rangle = \prod_i \sum_j [U_{ji}(q)c_j + V_{ji}(q)c_j^\dagger] |-\rangle \quad \forall q \in Q$$

2. **Dynamic:** Solve the equation of evolution of the **TDGCM** wavefunction at the **GOA** approximation

$$|\psi(t)\rangle = \int_q f(q, t) |\phi(q)\rangle$$

What observables can we predict ?

- Fragment properties
- Mass and Charge yields
- TKE



Dynamic propagation using FELIX code - Courtesy of D. Regnier (CEA)

Context

What is the TDGCM approach ?

1. **Static:** Generate a set of deformed **HFB** states

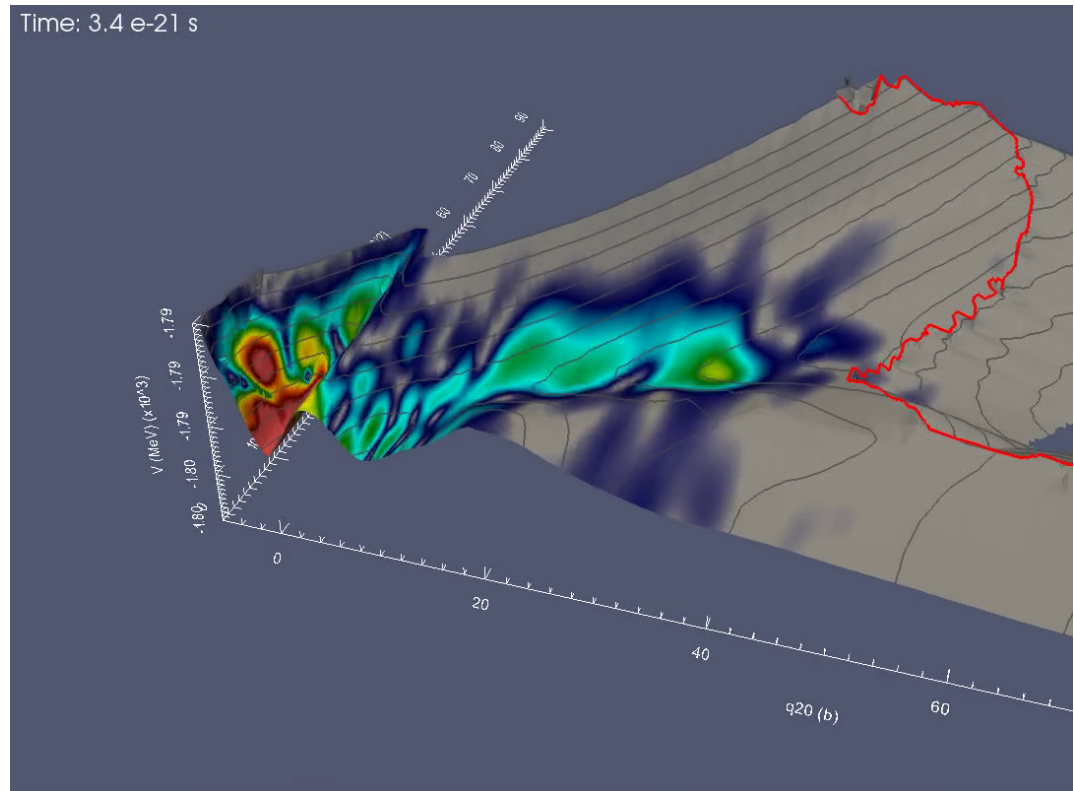
$$|\phi(q)\rangle = \prod_i \sum_j [U_{ji}(q)c_j + V_{ji}(q)c_j^\dagger] |-\rangle \quad \forall q \in Q$$

2. **Dynamic:** Solve the equation of evolution of the **TDGCM** wavefunction at the **GOA** approximation

$$|\psi(t)\rangle = \int_q f(q, t) |\phi(q)\rangle$$

What observables can we predict ?

- Fragment properties
- Mass and Charge yields
- TKE



Dynamic propagation using FELIX code - Courtesy of D. Regnier (CEA)

Context

What is the TDGCM approach ?

1. **Static:** Generate a set of deformed **HFB** states

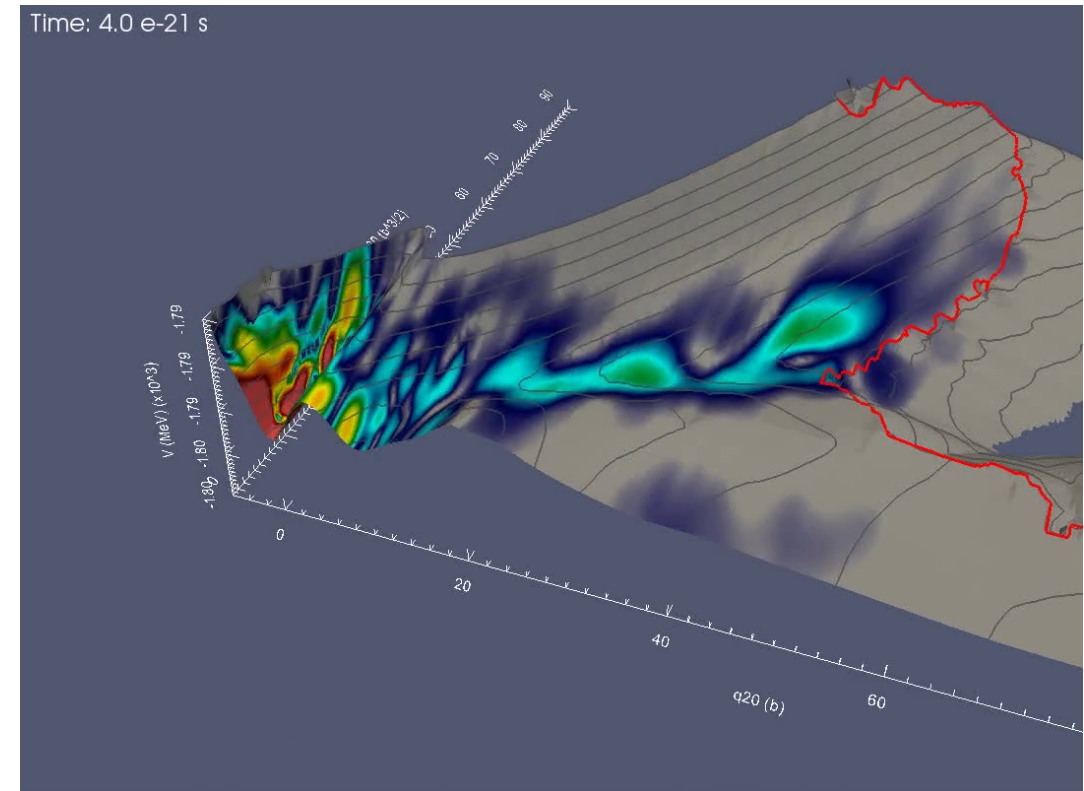
$$|\phi(q)\rangle = \prod_i \sum_j [U_{ji}(q)c_j + V_{ji}(q)c_j^\dagger] |-\rangle \quad \forall q \in Q$$

2. **Dynamic:** Solve the equation of evolution of the **TDGCM** wavefunction at the **GOA** approximation

$$|\psi(t)\rangle = \int_q f(q, t) |\phi(q)\rangle$$

What observables can we predict ?

- Fragment properties
- Mass and Charge yields
- TKE



Dynamic propagation using FELIX code - Courtesy of D. Regnier (CEA)

Context

What is the TDGCM approach ?

1. **Static:** Generate a set of deformed **HFB** states

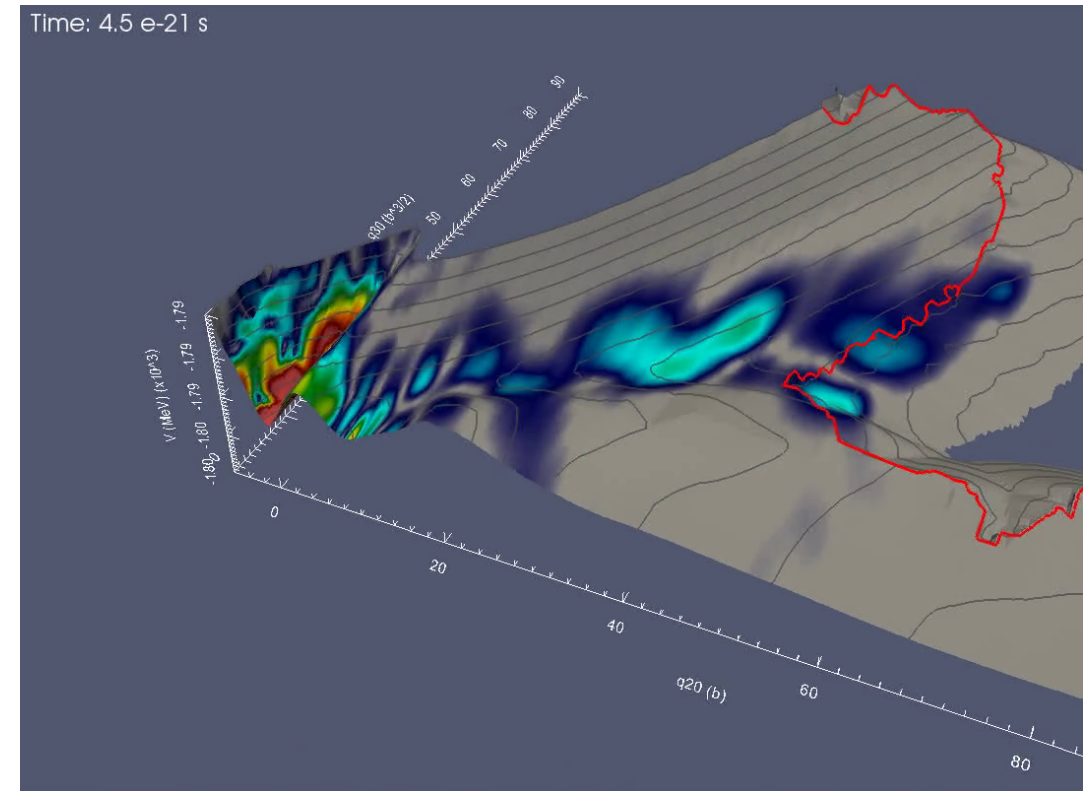
$$|\phi(q)\rangle = \prod_i \sum_j [U_{ji}(q)c_j + V_{ji}(q)c_j^\dagger] |-\rangle \quad \forall q \in Q$$

2. **Dynamic:** Solve the equation of evolution of the **TDGCM** wavefunction at the **GOA** approximation

$$|\psi(t)\rangle = \int_q f(q, t) |\phi(q)\rangle$$

What observables can we predict ?

- Fragment properties
- Mass and Charge yields
- TKE



Dynamic propagation using FELIX code - Courtesy of D. Regnier (CEA)

Context

What is the TDGCM approach ?

1. **Static:** Generate a set of deformed **HFB** states

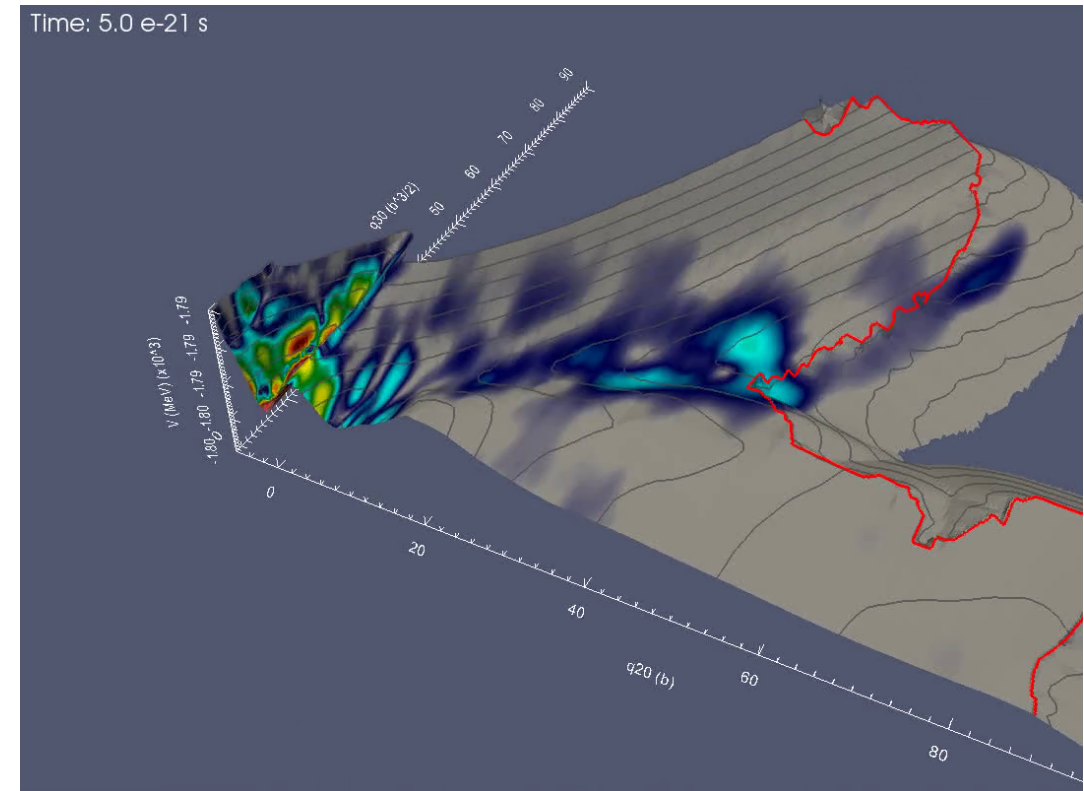
$$|\phi(q)\rangle = \prod_i \sum_j [U_{ji}(q)c_j + V_{ji}(q)c_j^\dagger] |-\rangle \quad \forall q \in Q$$

2. **Dynamic:** Solve the equation of evolution of the **TDGCM** wavefunction at the **GOA** approximation

$$|\psi(t)\rangle = \int_q f(q, t) |\phi(q)\rangle$$

What observables can we predict ?

- Fragment properties
- Mass and Charge yields
- TKE



Dynamic propagation using FELIX code - Courtesy of D. Regnier (CEA)

Context

What is the TDGCM approach ?

1. **Static:** Generate a set of deformed **HFB** states

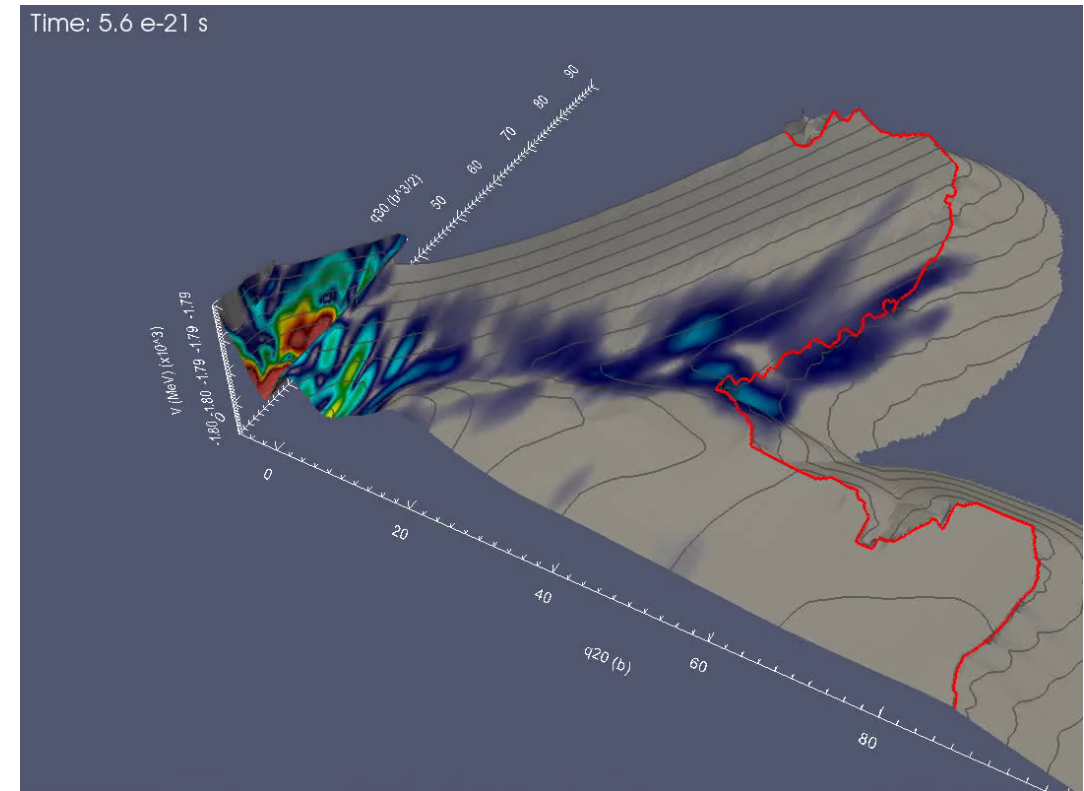
$$|\phi(q)\rangle = \prod_i \sum_j [U_{ji}(q)c_j + V_{ji}(q)c_j^\dagger] |-\rangle \quad \forall q \in Q$$

2. **Dynamic:** Solve the equation of evolution of the **TDGCM** wavefunction at the **GOA** approximation

$$|\psi(t)\rangle = \int_q f(q, t) |\phi(q)\rangle$$

What observables can we predict ?

- Fragment properties
- Mass and Charge yields
- TKE



Dynamic propagation using FELIX code - Courtesy of D. Regnier (CEA)

Context

What is the TDGCM approach ?

1. **Static:** Generate a set of deformed **HFB** states

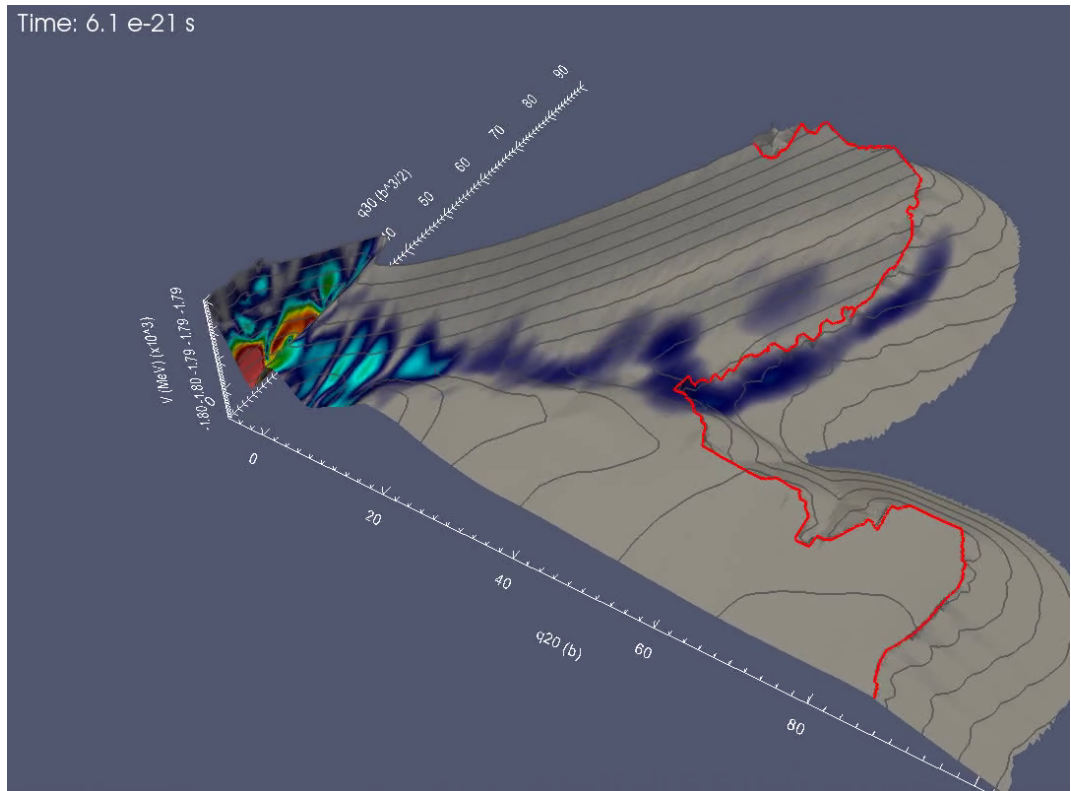
$$|\phi(q)\rangle = \prod_i \sum_j [U_{ji}(q)c_j + V_{ji}(q)c_j^\dagger] |-\rangle \quad \forall q \in Q$$

2. **Dynamic:** Solve the equation of evolution of the **TDGCM** wavefunction at the **GOA** approximation

$$|\psi(t)\rangle = \int_q f(q, t) |\phi(q)\rangle$$

What observables can we predict ?

- Fragment properties
- Mass and Charge yields
- TKE



Dynamic propagation using FELIX code - Courtesy of D. Regnier (CEA)

Context

What is the TDGCM approach ?

1. **Static:** Generate a set of deformed **HFB** states

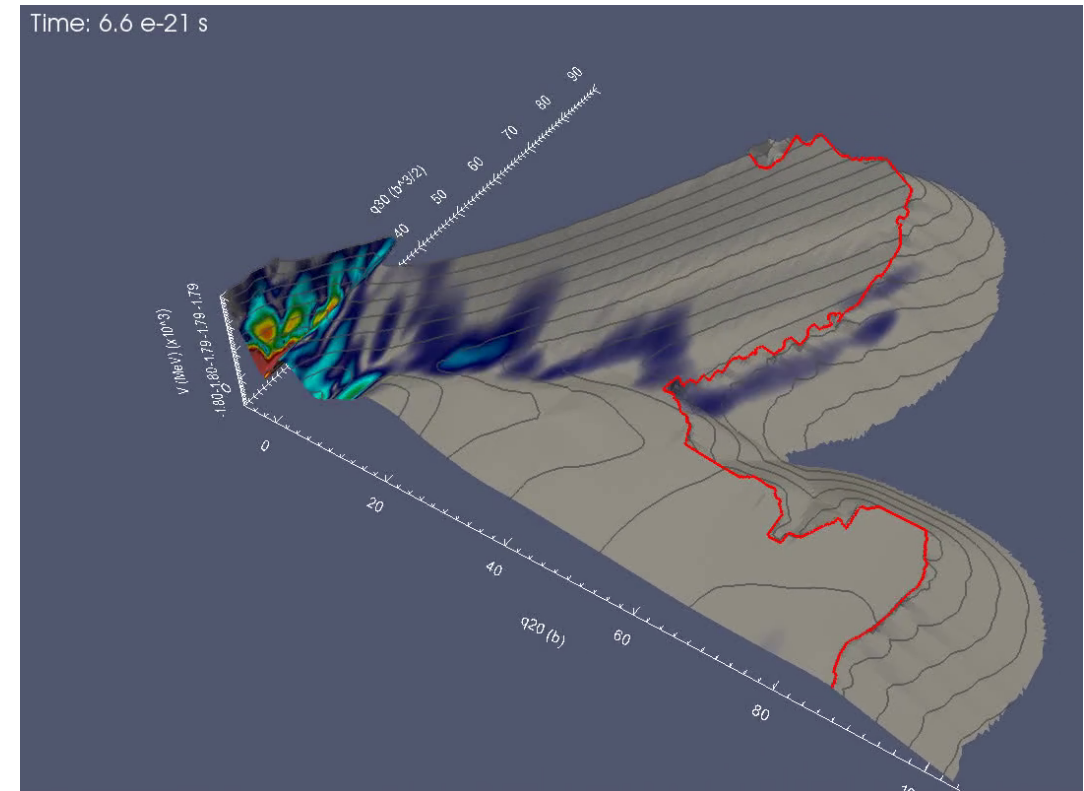
$$|\phi(q)\rangle = \prod_i \sum_j [U_{ji}(q)c_j + V_{ji}(q)c_j^\dagger] |-\rangle \quad \forall q \in Q$$

2. **Dynamic:** Solve the equation of evolution of the **TDGCM** wavefunction at the **GOA** approximation

$$|\psi(t)\rangle = \int_q f(q, t) |\phi(q)\rangle$$

What observables can we predict ?

- Fragment properties
- Mass and Charge yields
- TKE



Dynamic propagation using FELIX code - Courtesy of D. Regnier (CEA)

Context

What is the TDGCM approach ?

1. **Static:** Generate a set of deformed **HFB** states

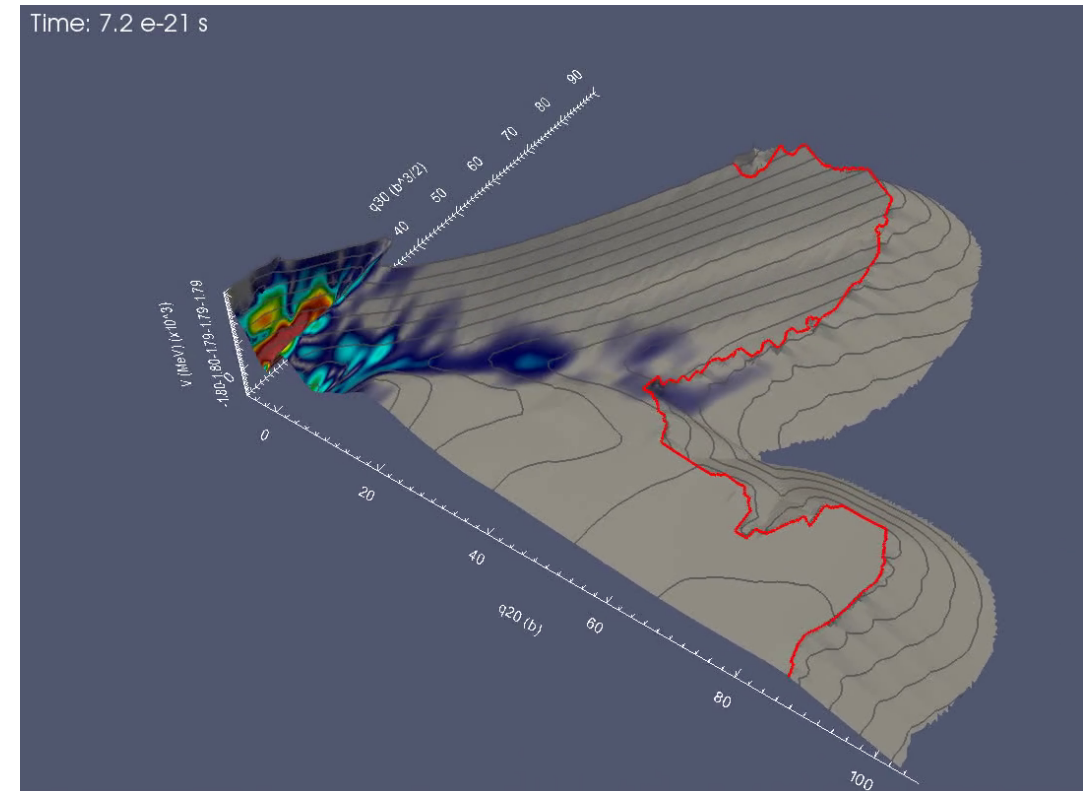
$$|\phi(q)\rangle = \prod_i \sum_j [U_{ji}(q)c_j + V_{ji}(q)c_j^\dagger] |-\rangle \quad \forall q \in Q$$

2. **Dynamic:** Solve the equation of evolution of the **TDGCM** wavefunction at the **GOA** approximation

$$|\psi(t)\rangle = \int_q f(q, t) |\phi(q)\rangle$$

What observables can we predict ?

- Fragment properties
- Mass and Charge yields
- TKE



Dynamic propagation using FELIX code - Courtesy of D. Regnier (CEA)

Context

What is the TDGCM approach ?

1. **Static:** Generate a set of deformed **HFB** states

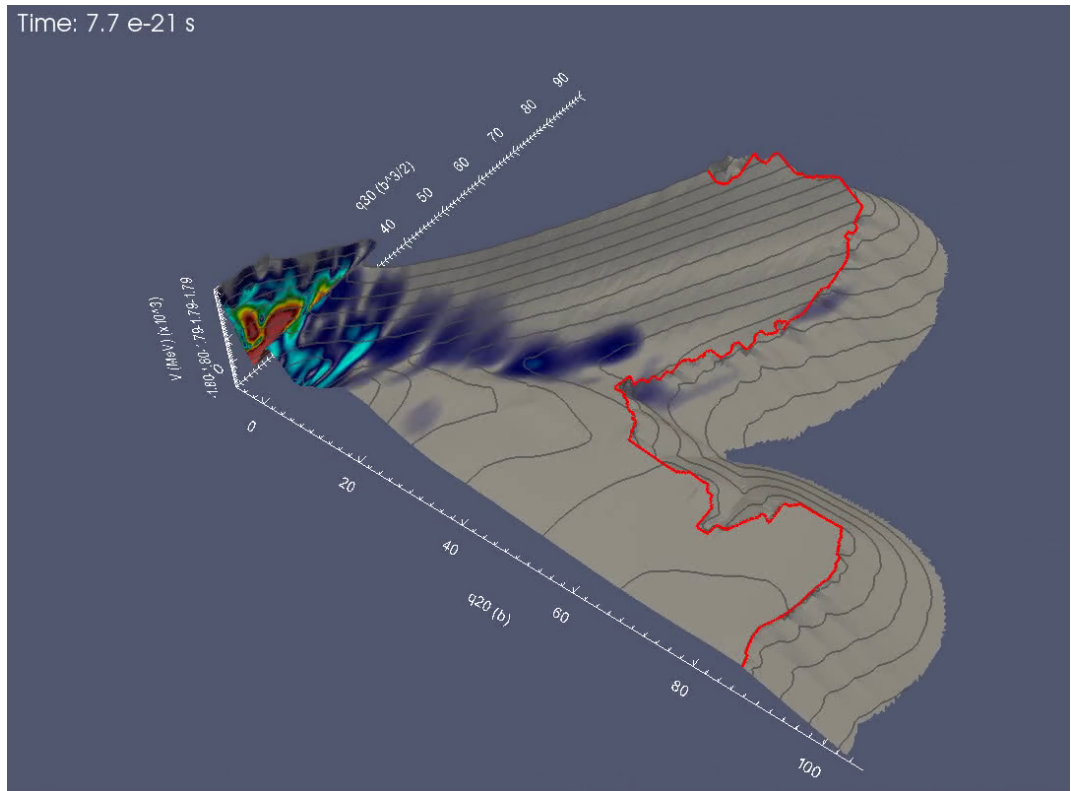
$$|\phi(q)\rangle = \prod_i \sum_j [U_{ji}(q)c_j + V_{ji}(q)c_j^\dagger] |-\rangle \quad \forall q \in Q$$

2. **Dynamic:** Solve the equation of evolution of the **TDGCM** wavefunction at the **GOA** approximation

$$|\psi(t)\rangle = \int_q f(q, t) |\phi(q)\rangle$$

What observables can we predict ?

- Fragment properties
- Mass and Charge yields
- TKE



Dynamic propagation using FELIX code - Courtesy of D. Regnier (CEA)

Context

What is the TDGCM approach ?

1. **Static:** Generate a set of deformed **HFB** states

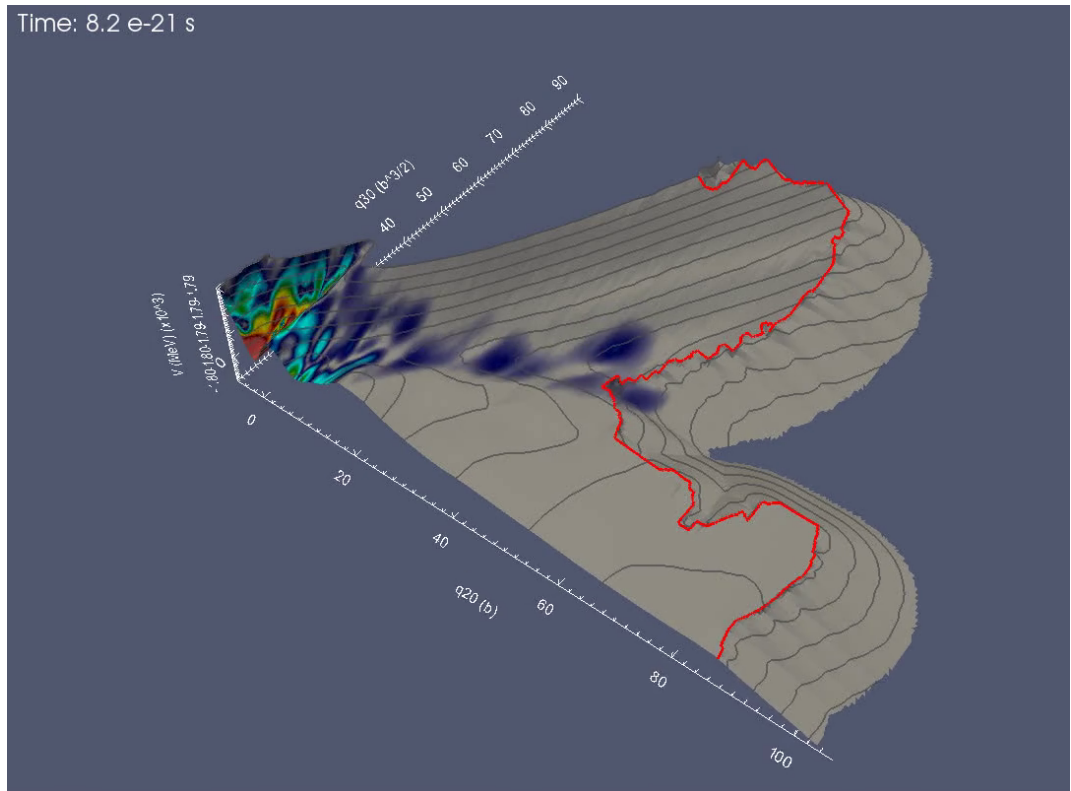
$$|\phi(q)\rangle = \prod_i \sum_j [U_{ji}(q)c_j + V_{ji}(q)c_j^\dagger] |-\rangle \quad \forall q \in Q$$

2. **Dynamic:** Solve the equation of evolution of the **TDGCM** wavefunction at the **GOA** approximation

$$|\psi(t)\rangle = \int_q f(q, t) |\phi(q)\rangle$$

What observables can we predict ?

- Fragment properties
- Mass and Charge yields
- TKE



Dynamic propagation using FELIX code - Courtesy of D. Regnier (CEA)

Context

What is the TDGCM approach ?

1. **Static:** Generate a set of deformed **HFB** states

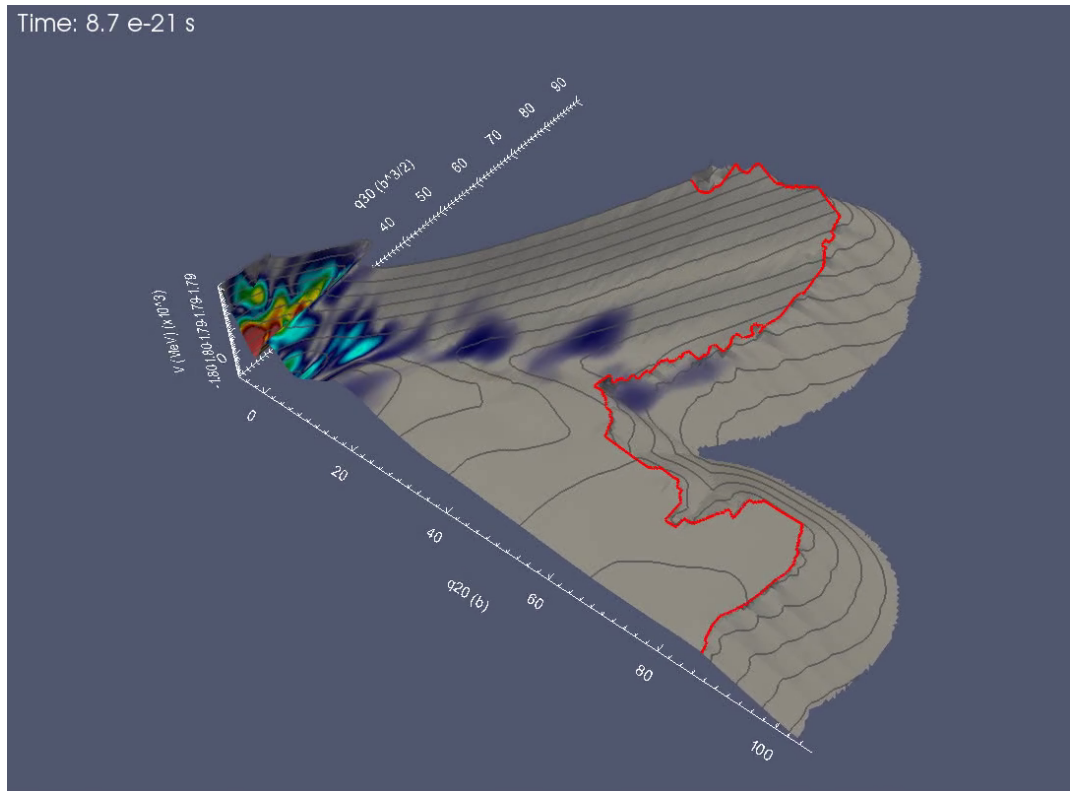
$$|\phi(q)\rangle = \prod_i \sum_j [U_{ji}(q)c_j + V_{ji}(q)c_j^\dagger] |-\rangle \quad \forall q \in Q$$

2. **Dynamic:** Solve the equation of evolution of the **TDGCM** wavefunction at the **GOA** approximation

$$|\psi(t)\rangle = \int_q f(q, t) |\phi(q)\rangle$$

What observables can we predict ?

- Fragment properties
- Mass and Charge yields
- TKE



Dynamic propagation using FELIX code - Courtesy of D. Regnier (CEA)

Context

What is the TDGCM approach ?

1. **Static:** Generate a set of deformed **HFB** states

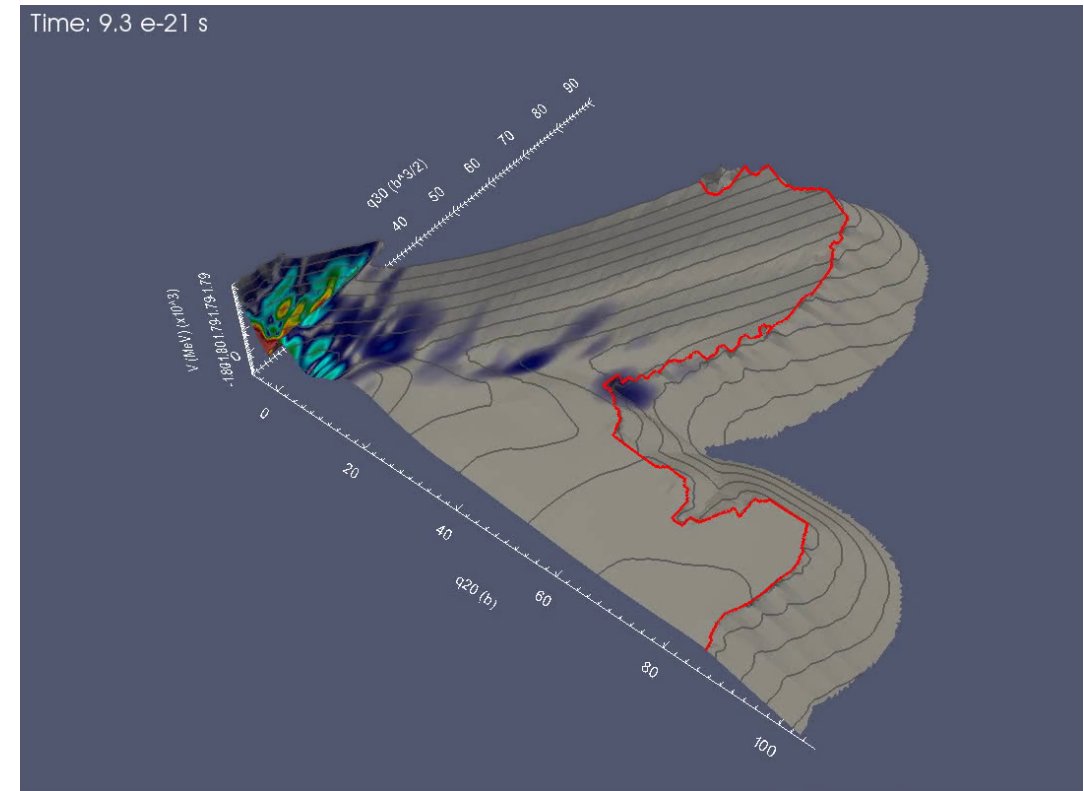
$$|\phi(q)\rangle = \prod_i \sum_j [U_{ji}(q)c_j + V_{ji}(q)c_j^\dagger] |-\rangle \quad \forall q \in Q$$

2. **Dynamic:** Solve the equation of evolution of the **TDGCM** wavefunction at the **GOA** approximation

$$|\psi(t)\rangle = \int_q f(q, t) |\phi(q)\rangle$$

What observables can we predict ?

- Fragment properties
- Mass and Charge yields
- TKE



Dynamic propagation using FELIX code - Courtesy of D. Regnier (CEA)

Context

What is the TDGCM approach ?

1. **Static:** Generate a set of deformed **HFB** states

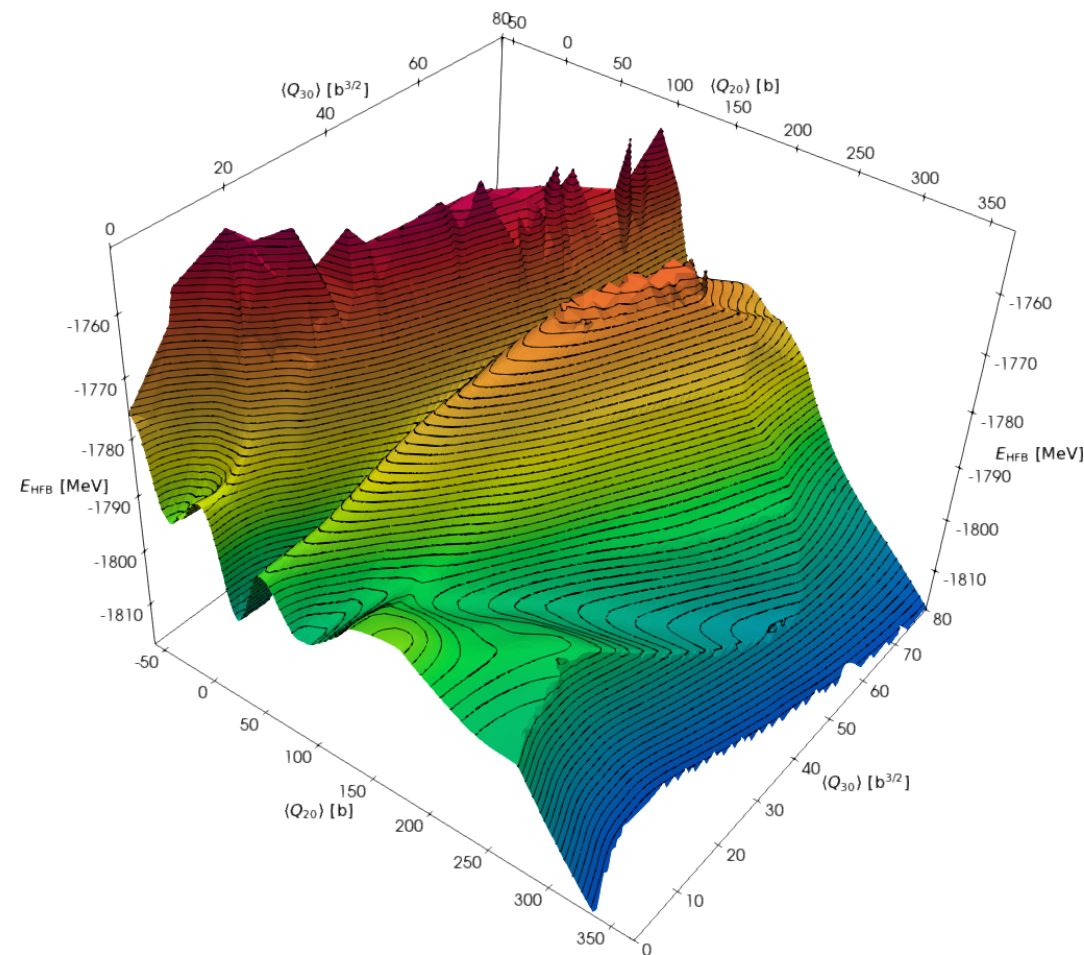
$$|\phi(q)\rangle = \prod_i \sum_j [U_{ji}(q)c_j + V_{ji}(q)c_j^\dagger] |-\rangle \quad \forall q \in Q$$

2. **Dynamic:** Solve the equation of evolution of the **TDGCM** wavefunction at the **GOA** approximation

$$|\psi(t)\rangle = \int_q f(q, t) |\phi(q)\rangle$$

What observables can we predict ?

- Fragment properties
- Mass and Charge yields
- TKE



$^{240}\text{Pu} (q_{20}, q_{30})$ potential energy surface

Table of contents

I. HFB Solver - **HFB3**

II. PES Generation - **CHICON**

III. Odd nuclei

1. **Static:** Generate a set of deformed **HFB** states

$$|\phi(q)\rangle = \prod_i \sum_j [U_{ji}(q)c_j + V_{ji}(q)c_j^\dagger] |-\rangle \quad \forall q \in Q$$

HFB Method

1. Nuclear hamiltonian H

$$H = t + v_{12}$$

- t - kinetic energy
- v_{12} - 2-body interaction

2. Ritz Variational principle

$$\delta \frac{\langle \phi | H | \phi \rangle}{\langle \phi | \phi \rangle} = 0$$

We introduce the **HFB ansatz** $|\phi\rangle$ (independent quasiparticle approximation):

$$|\phi\rangle = \prod_i \xi_i |-\rangle \quad \xi_i = \sum_j [U_{ji} c_j + V_{ji} c_j^+]$$

This leads to the HFB equations:

$$[\mathcal{H}, R] = 0$$

- $\mathcal{H}$ - Quasiparticle hamiltonian
- R - Generalised density matrix

Nuclear interaction

Evolution of the Gogny interaction

- D1:
 - Introduces a finite range central term
- D2:
 - Introduces a finite range density term
- **DG (G. Zietek PhD)**
 - Introduces a finite range spin-orbit term
 - Introduces a tensor term (also finite range)
 - Reproduces **D1** and **D2** type interactions at the **0-range limit**

	Cen.	Den.	S.o.	Ten.
D1	✓	0	0	✗
D2	✓	✓	0	✗
DG	✓	✓	✓	✓

- **0** zero-range

$$V_i(r_{12}) = \delta(r_1, r_2)$$

- **✓ finite range**

$$V_i(r_{12}) = e^{-\frac{(r_1 - r_2)^2}{\mu_i^2}}$$

$$\begin{aligned}
 \hat{v}_{12} \equiv & \sum_{i=1}^2 (W_i + B_i P_\sigma - H_i P_\tau - M_i P_\sigma P_\tau) V_i(r_{12}) && \text{Central term} \\
 & + (\mu_3 \sqrt{\pi})^{-3} (W_3 + B_3 P_\sigma - H_3 P_\tau - M_3 P_\sigma P_\tau) V_3(r_{12}) && \frac{\rho^\alpha(\vec{r}_1) + \rho^\alpha(\vec{r}_2)}{2} \text{ Density dependent term} \\
 & + B(\mu_5) (W_5 - H_5 P_\tau) V_5(r_{12}) && \vec{L} \cdot \vec{S} \text{ Spin-orbit term} \\
 & + (W_7 - H_7 P_\tau) V_7(r_{12}) && S_{12} \text{ Tensor term}
 \end{aligned}$$

HFB Method

1. Diagonalization problem

$$[\mathcal{H}, \mathbf{R}] = 0$$

HFB Method

1. ~~Diagonalization problem~~ Fixed point problem

$$[\mathcal{H}[\mathbf{R}], \mathbf{R}] = 0$$

⚙ **Iterative** diagonalization

HFB Method

1. ~~Diagonalization problem~~ Fixed point problem

$$[\mathcal{H}_c[\mathbf{R}], \mathbf{R}] = 0$$

⚙️ Iterative diagonalization

2. Constraints

1. Constrain the average number of protons and neutrons $\mathbf{N}, \mathbf{P}$

$$\mathcal{H}_c[\mathbf{R}] = \mathbf{H}[\mathbf{R}] + \lambda_n \langle \phi | \mathbf{N} | \phi \rangle + \lambda_p \langle \phi | \mathbf{P} | \phi \rangle + \sum_i \lambda_i \langle \phi | \mathbf{Q}_i | \phi \rangle$$

2. Deformed states: geometrical constraints $\mathbf{Q}_i$

⚙️ Constraints enforced through lagrange multipliers

HFB3 Solver



- C++14 with Python bindings provided
- Uses **Armadillo** for fast linear algebra
- Accepted for publication in EPJA

Symmetries

- Axial symmetry enforced
- Time reversal symmetry enforced
- Parity symmetry broken
- Particle number symmetry broken

Interaction

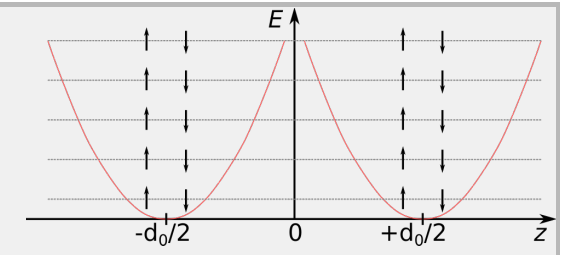
- Effective nucleon-nucleon interaction
- **D1, D2, DG** analytical forms (Gogny)
- User defined interaction

Features

- Gradient/Broyden solver
- Odd-nuclei (blocking + EFA)

Harmonic Oscillator basis with 1 or 2 centers

- 2-center enables a **better description of elongated shapes** as can be seen during fission
- **non orthogonal** basis states
- **automatic basis parameters optimization** for each calculation.

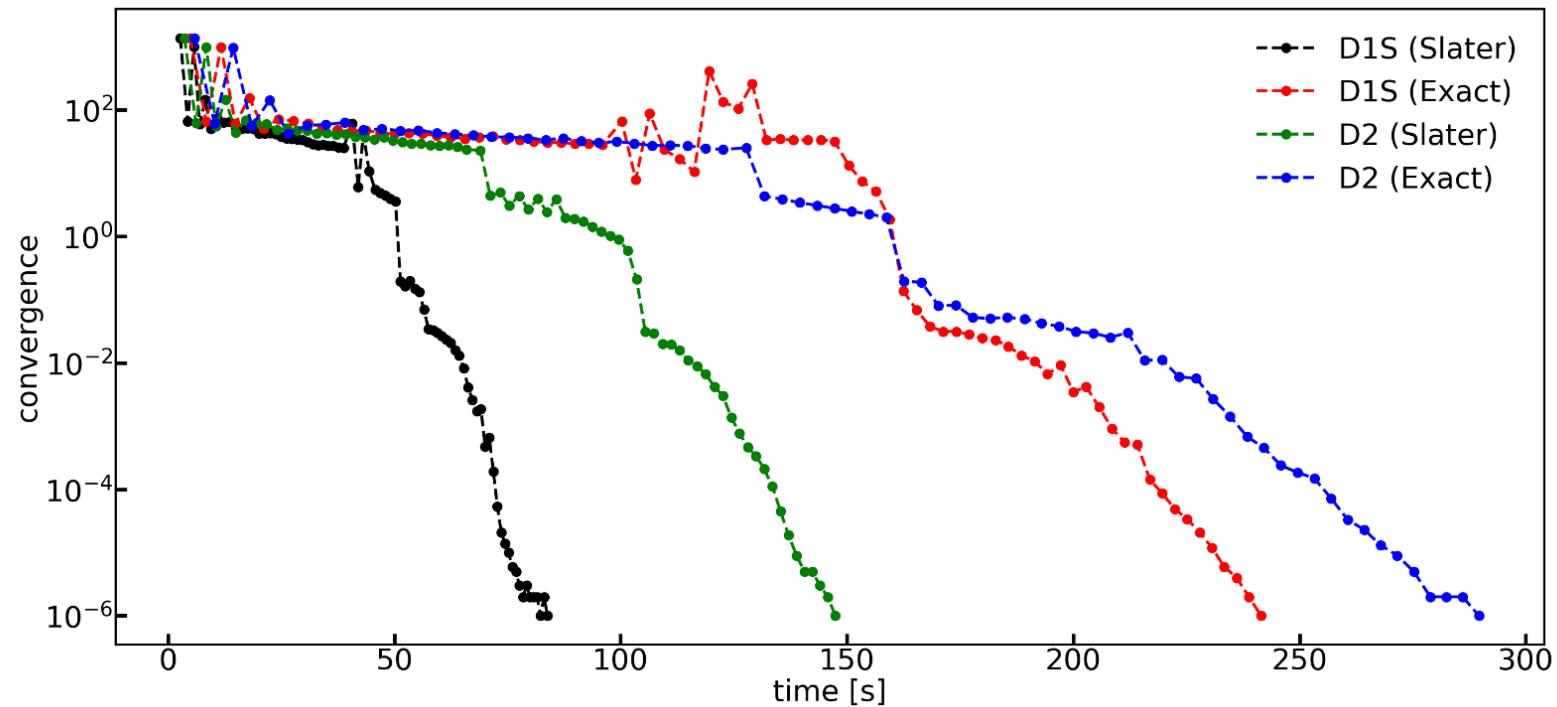


Performance

Why do we need good performance ?

✓ One HFB calculation: ~80s

✗ One PES: 300 000 HFB calculations → ~7000 cpu/h



HFB3 performance on 240Pu for D1 and D2-type Gogny interactions - P. Carpentier PhD Thesis 2024

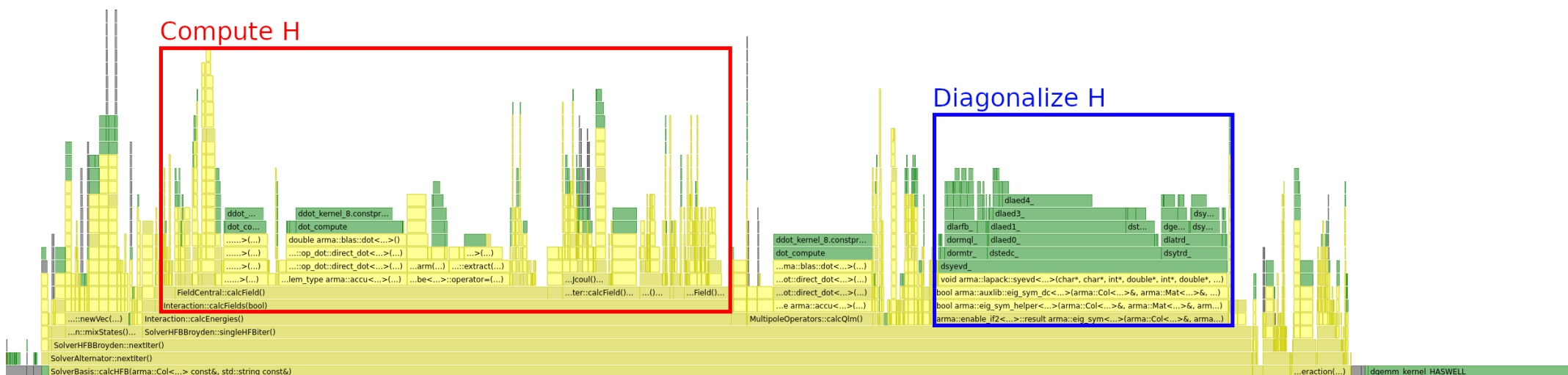
Iterative process

Faster iterations

- Use dedicated libraries for matrix operations
- 🎯 **perf** profiling tool
- ✅ 50% of time spent in lapack/blas calls

Less iterations

- Alternate multiple solvers
 - **Gradient descent**: slow but rock-solid convergence
 - **Broyden-mixing**: fast but fragile convergence
- Sub-iteration techniques



HFB3 profiling graph for a 240Pu D1S calculation - box width is proportional to time spent

🧐 We have a **low** number of **fast** iterations that **converges** very well even at **high deformation**



HFB3: an axial HFB solver with Gogny forces using a 2-center HO basis (C++/Python)

N. Dubray^{1,2}, J.-P. Ebran^{1,2}, P. Carpentier^{1,2}, M. Frosini³, A. Zdeb^{1,2,4}, N. Pillet^{1,2}, J. Newsome^{1,2}, M. Verrière^{1,2}, G. Accorto^{1,2}, and D. Regnier^{1,2}

¹ CEA, DAM, F-91297, Arpajon, France

² Laboratoire Matière en Conditions Extrêmes, Université Paris-Saclay, CEA, 91680, Bruyères-le-Châtel, France

³ CEA, DES, F-13108, Saint-Paul-Lez-Durance, France

⁴ Department of Theoretical Physics, Institute of Physics, Maria Curie-Skłodowska University, Lublin, Poland

Availability

- Manuscript available on **Arxiv**: <https://arxiv.org/abs/2506.10745>
- **HFB3** available on **GitHub**: <https://github.com/cea-phynu/hfb3>
- **HFB3** available on **PyPI**: <https://pypi.org/project/hfb3/>
- Licence GPL-3.0

Table of contents

I. HFB Solver - **HFB3**

II. PES Generation - **CHICON**

III. Odd nuclei

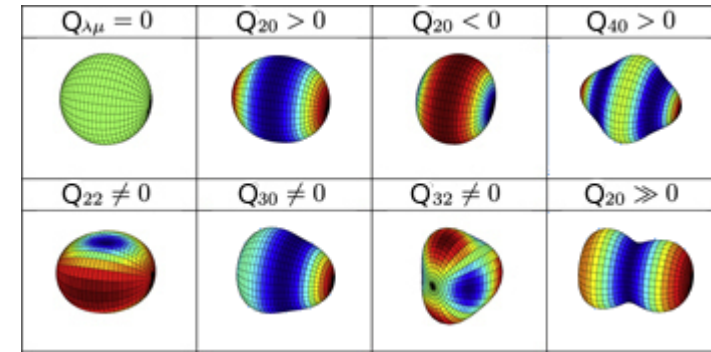
1. **Static:** Generate a set of deformed **HFB** states

$$|\phi(q)\rangle = \prod_i \sum_j [U_{ji}(q)c_j + V_{ji}(q)c_j^\dagger] |-\rangle \quad \forall q \in Q$$

PES Generation

Geometrical constraints

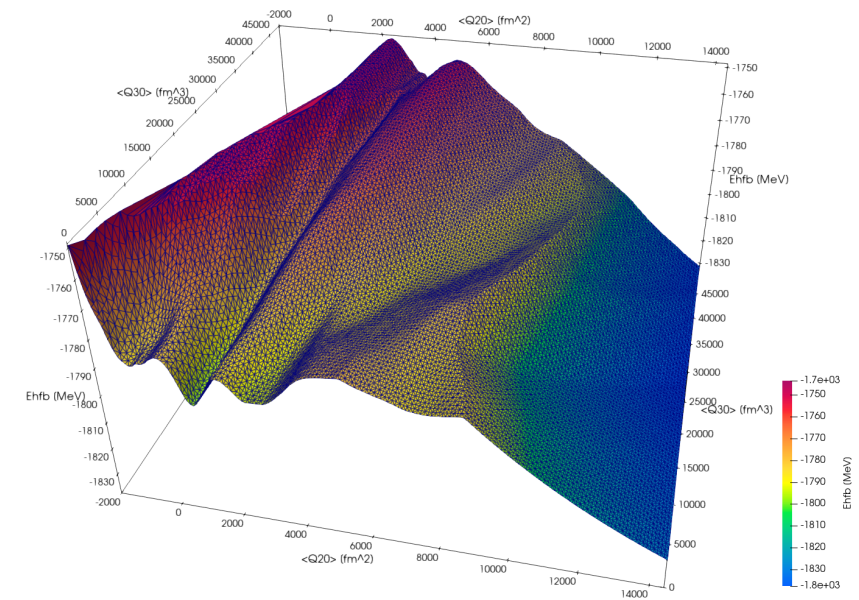
- Infinite dimensional hilbert space cannot be fully explored
- Collective coordinates (degrees of freedom)
 - Q_{20} (elongation)
 - Q_{30} (asymmetry)
 - Q_{40} (scission)



Main idea

- For each constraint $q = (q_{20}, q_{30}, \dots)$ in the mesh Q
 - Perform an HFB3 calculation constrained at q
- Refined mesh, very resource intensive: **20 000pts** (2D)
- Seems easy to parallelize, independent calculations...

```
01 | for q in Q:
02 |     pes[q] = hfb3(q)
```



^{239}U (q_{20}, q_{30}) PES with mesh lines visible

Techniques

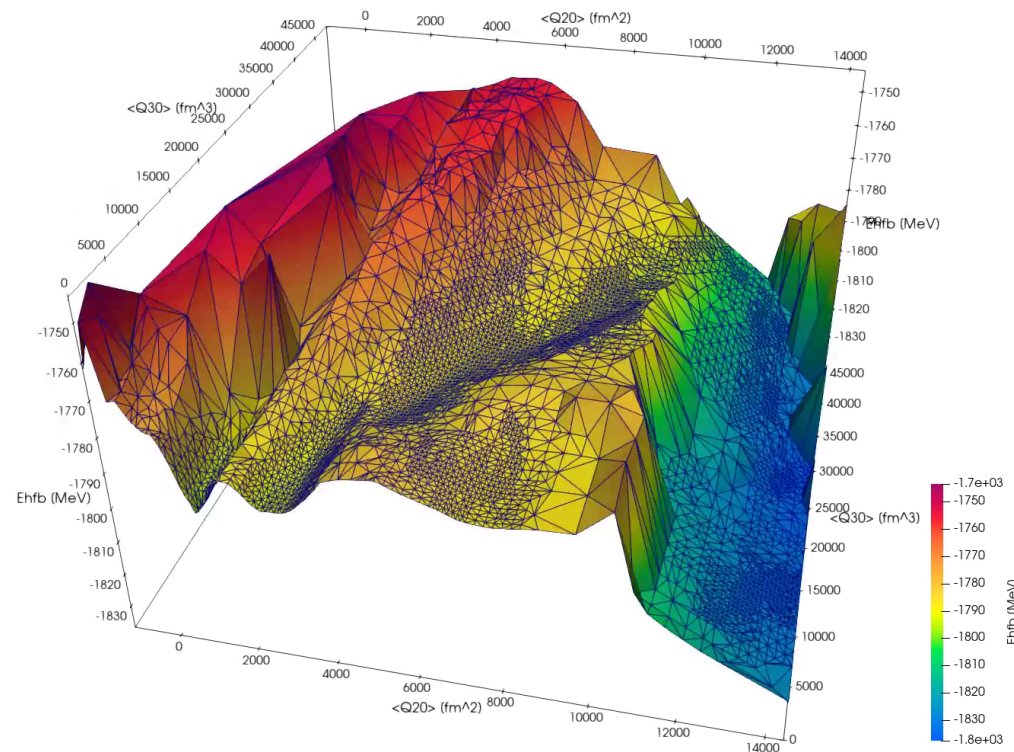
- Propagation techniques
 -  Faster convergence
- Retropropagation techniques
 -  Resolve some local minimas
 -  Ensure continuity in energy

What is CHICON ?

- A **ND-PES generator** written in python
- Developed at CEA by **N. Dubray** & **J. Newsome**
- HPC: MPI + Multiprocess parallelization
- Uses the HFB3 solver

Systematic production for IAEA: RIPL-4

- 1000 PES currently, 20 000 000 cpu/hour
-  Odd-nuclei PES still in progress



^{239}U (q_{20}, q_{30}) PES with mesh lines visible

Techniques

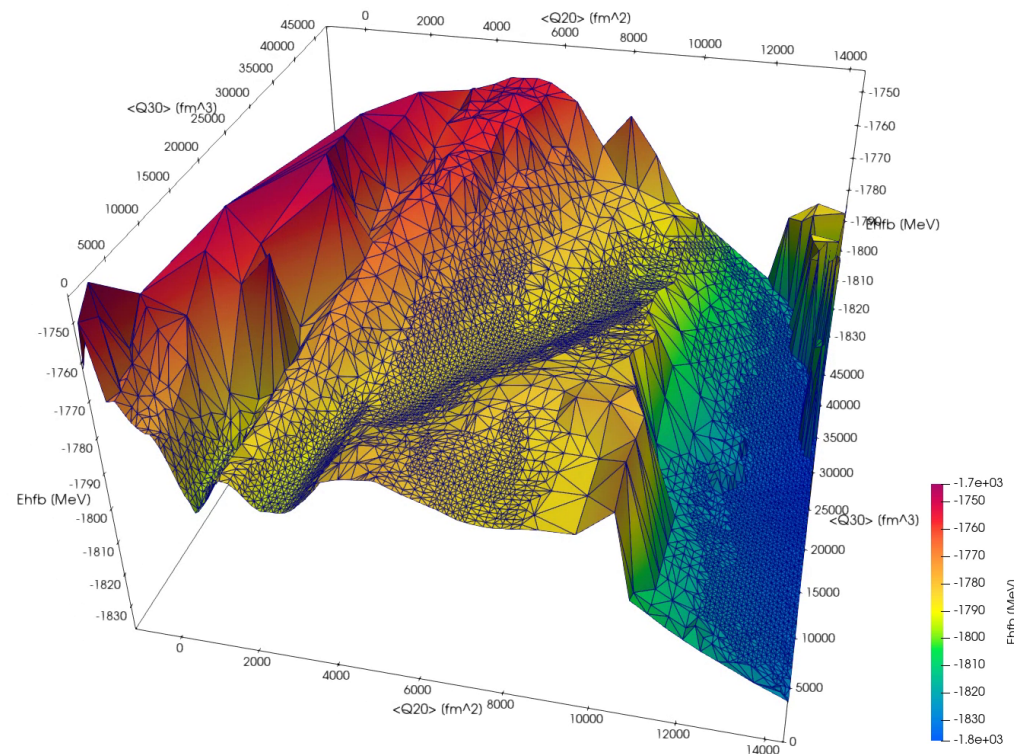
- Propagation techniques
 -  Faster convergence
- Retropropagation techniques
 -  Resolve some local minimas
 -  Ensure continuity in energy

What is CHICON ?

- A **ND-PES generator** written in python
- Developed at CEA by **N. Dubray** & **J. Newsome**
- HPC: MPI + Multiprocess parallelization
- Uses the HFB3 solver

Systematic production for IAEA: RIPL-4

- 1000 PES currently, 20 000 000 cpu/hour
-  Odd-nuclei PES still in progress



^{239}U (q_{20}, q_{30}) PES with mesh lines visible

Techniques

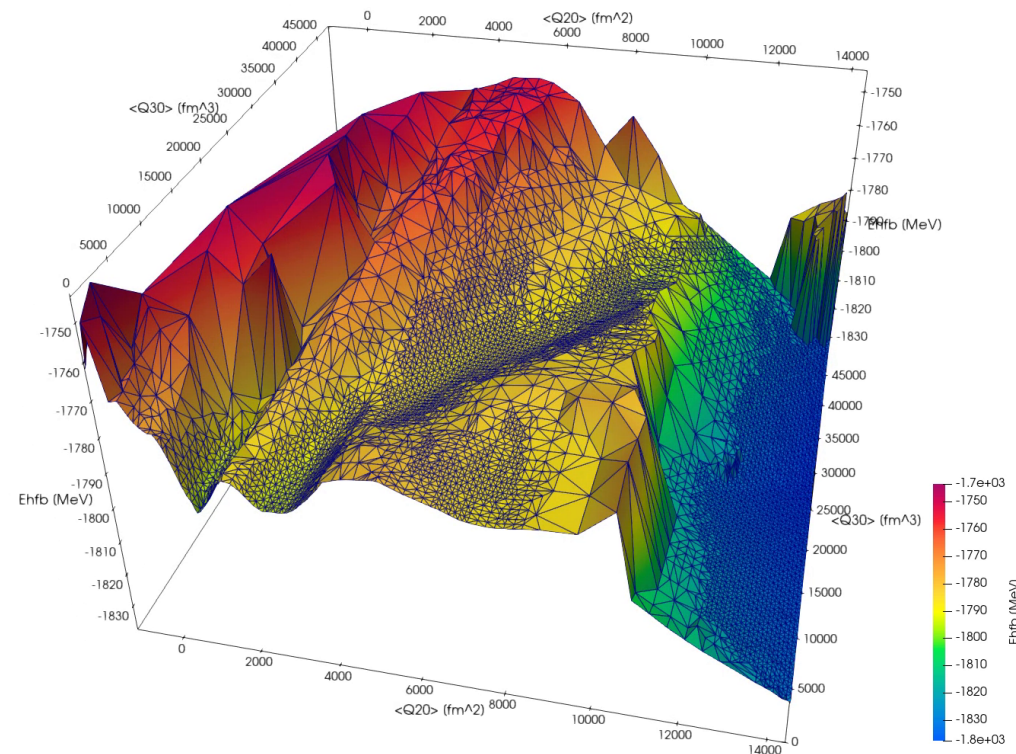
- Propagation techniques
 -  Faster convergence
- Retropropagation techniques
 -  Resolve some local minimas
 -  Ensure continuity in energy

What is CHICON ?

- A **ND-PES generator** written in python
- Developed at CEA by **N. Dubray** & **J. Newsome**
- HPC: MPI + Multiprocess parallelization
- Uses the HFB3 solver

Systematic production for IAEA: RIPL-4

- 1000 PES currently, 20 000 000 cpu/hour
-  Odd-nuclei PES still in progress



^{239}U (q_{20}, q_{30}) PES with mesh lines visible

Techniques

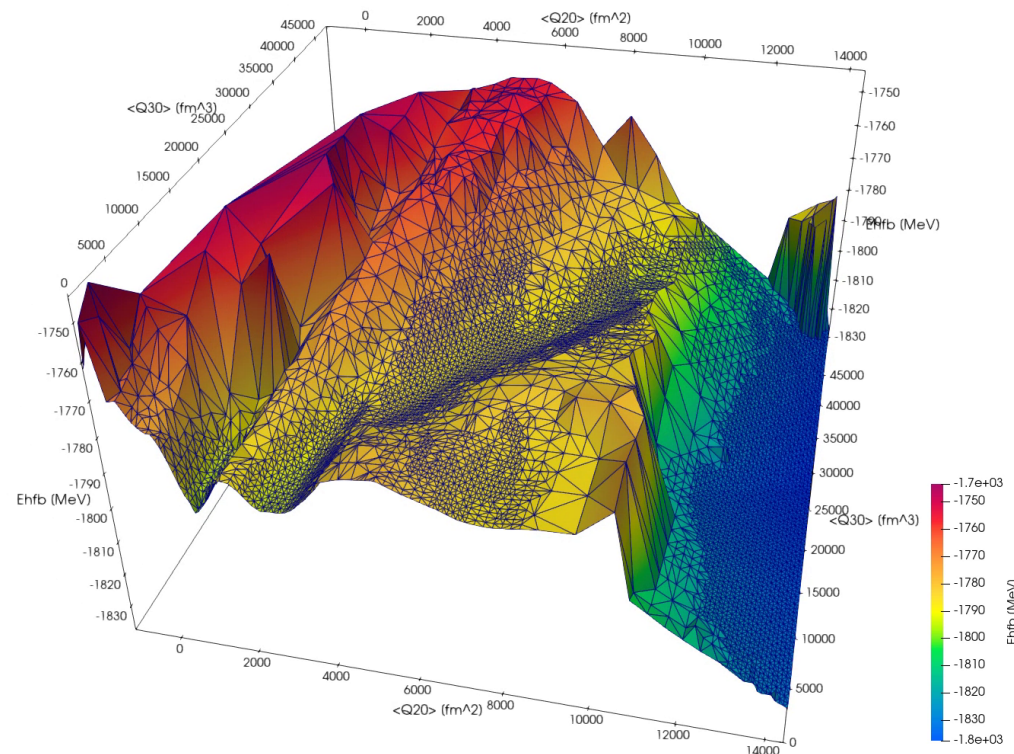
- Propagation techniques
 -  Faster convergence
- Retropropagation techniques
 -  Resolve some local minimas
 -  Ensure continuity in energy

What is CHICON ?

- A **ND-PES generator** written in python
- Developed at CEA by **N. Dubray** & **J. Newsome**
- HPC: MPI + Multiprocess parallelization
- Uses the HFB3 solver

Systematic production for IAEA: RIPL-4

- 1000 PES currently, 20 000 000 cpu/hour
-  Odd-nuclei PES still in progress



^{239}U (q_{20}, q_{30}) PES with mesh lines visible

Techniques

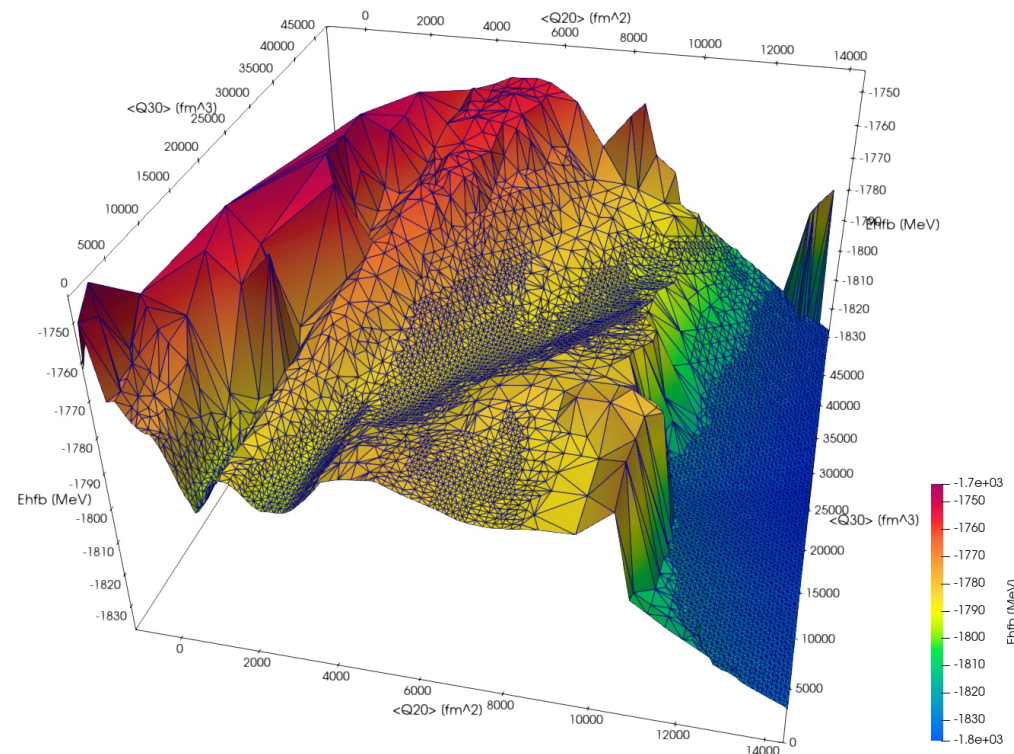
- Propagation techniques
 -  Faster convergence
- Retropropagation techniques
 -  Resolve some local minimas
 -  Ensure continuity in energy

What is CHICON ?

- A **ND-PES generator** written in python
- Developed at CEA by **N. Dubray** & **J. Newsome**
- HPC: MPI + Multiprocess parallelization
- Uses the HFB3 solver

Systematic production for IAEA: RIPL-4

- 1000 PES currently, 20 000 000 cpu/hour
-  Odd-nuclei PES still in progress



^{239}U (q_{20}, q_{30}) PES with mesh lines visible

Techniques

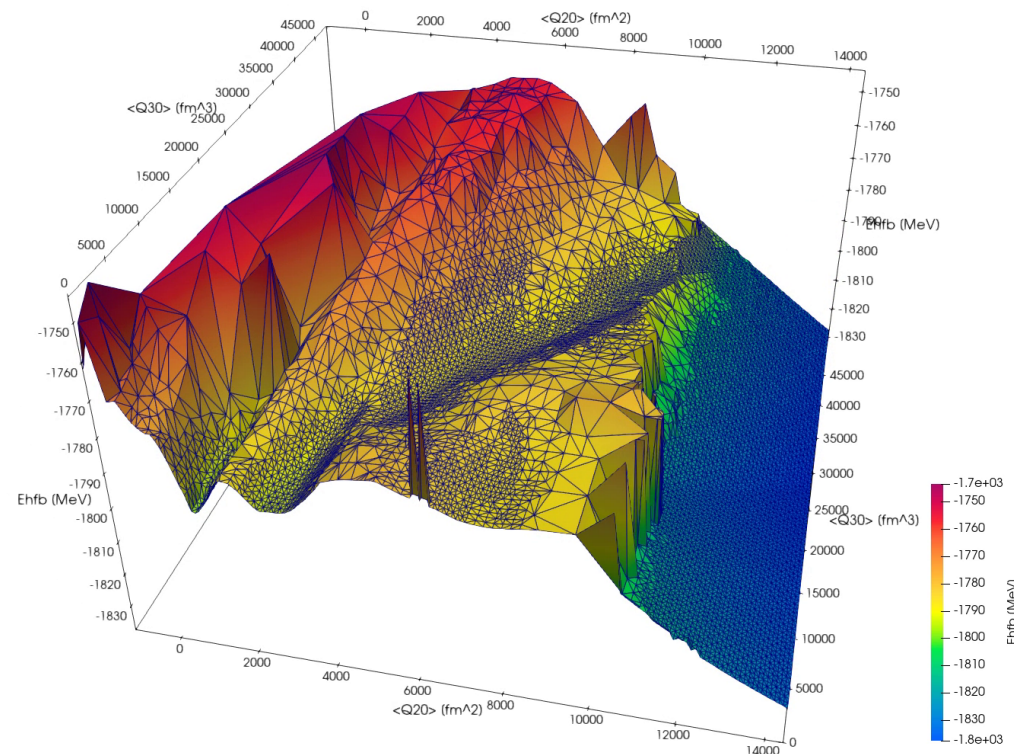
- Propagation techniques
 -  Faster convergence
- Retropropagation techniques
 -  Resolve some local minimas
 -  Ensure continuity in energy

What is CHICON ?

- A **ND-PES generator** written in python
- Developed at CEA by **N. Dubray** & **J. Newsome**
- HPC: MPI + Multiprocess parallelization
- Uses the HFB3 solver

Systematic production for IAEA: RIPL-4

- 1000 PES currently, 20 000 000 cpu/hour
-  Odd-nuclei PES still in progress



^{239}U (q_{20}, q_{30}) PES with mesh lines visible

Techniques

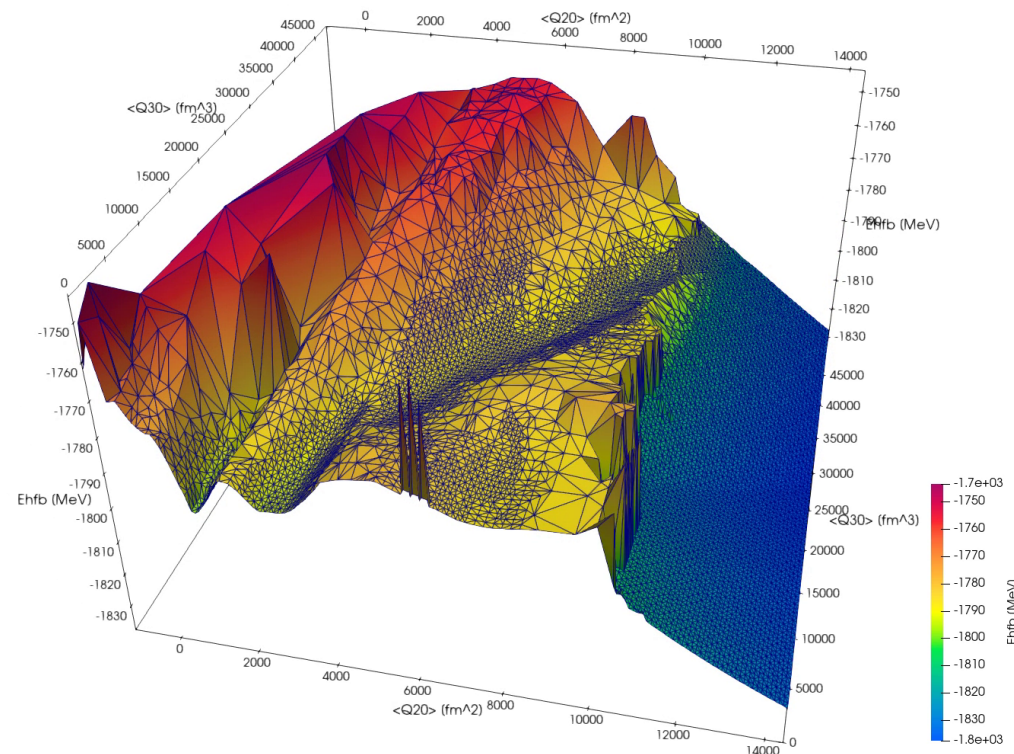
- Propagation techniques
 -  Faster convergence
- Retropropagation techniques
 -  Resolve some local minimas
 -  Ensure continuity in energy

What is CHICON ?

- A **ND-PES generator** written in python
- Developed at CEA by **N. Dubray** & **J. Newsome**
- HPC: MPI + Multiprocess parallelization
- Uses the HFB3 solver

Systematic production for IAEA: RIPL-4

- 1000 PES currently, 20 000 000 cpu/hour
-  Odd-nuclei PES still in progress



^{239}U (q_{20}, q_{30}) PES with mesh lines visible

Techniques

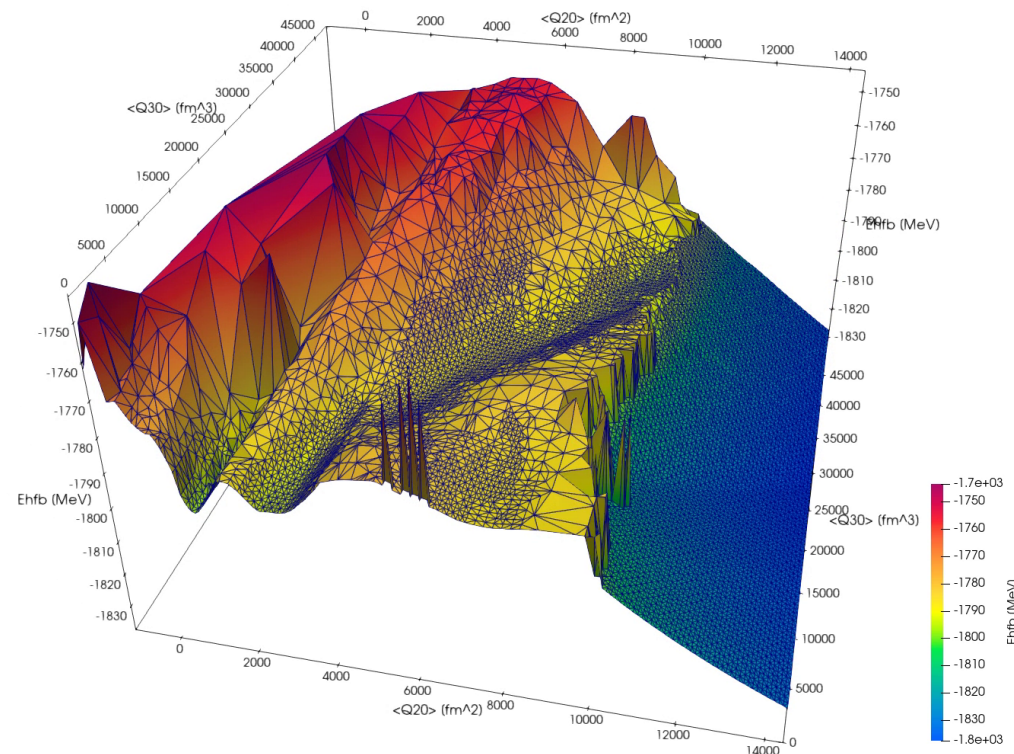
- Propagation techniques
 -  Faster convergence
- Retropropagation techniques
 -  Resolve some local minimas
 -  Ensure continuity in energy

What is CHICON ?

- A **ND-PES generator** written in python
- Developed at CEA by **N. Dubray** & **J. Newsome**
- HPC: MPI + Multiprocess parallelization
- Uses the HFB3 solver

Systematic production for IAEA: RIPL-4

- 1000 PES currently, 20 000 000 cpu/hour
-  Odd-nuclei PES still in progress



^{239}U (q_{20}, q_{30}) PES with mesh lines visible

Techniques

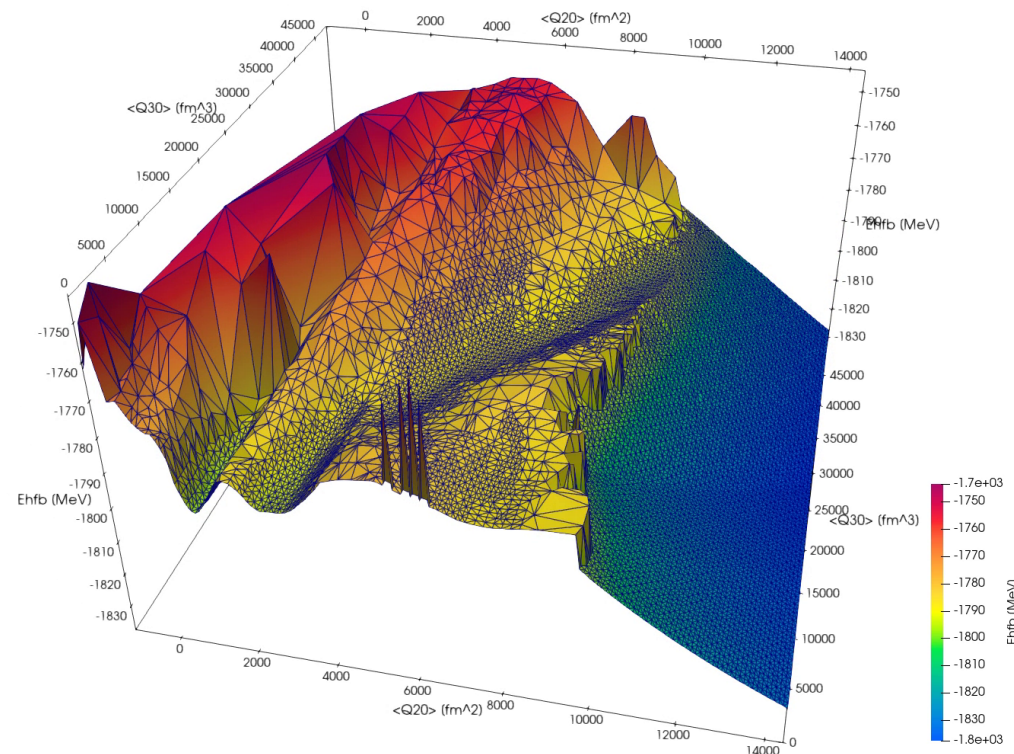
- Propagation techniques
 -  Faster convergence
- Retropropagation techniques
 -  Resolve some local minimas
 -  Ensure continuity in energy

What is CHICON ?

- A **ND-PES generator** written in python
- Developed at CEA by **N. Dubray** & **J. Newsome**
- HPC: MPI + Multiprocess parallelization
- Uses the HFB3 solver

Systematic production for IAEA: RIPL-4

- 1000 PES currently, 20 000 000 cpu/hour
-  Odd-nuclei PES still in progress



^{239}U (q_{20}, q_{30}) PES with mesh lines visible

Techniques

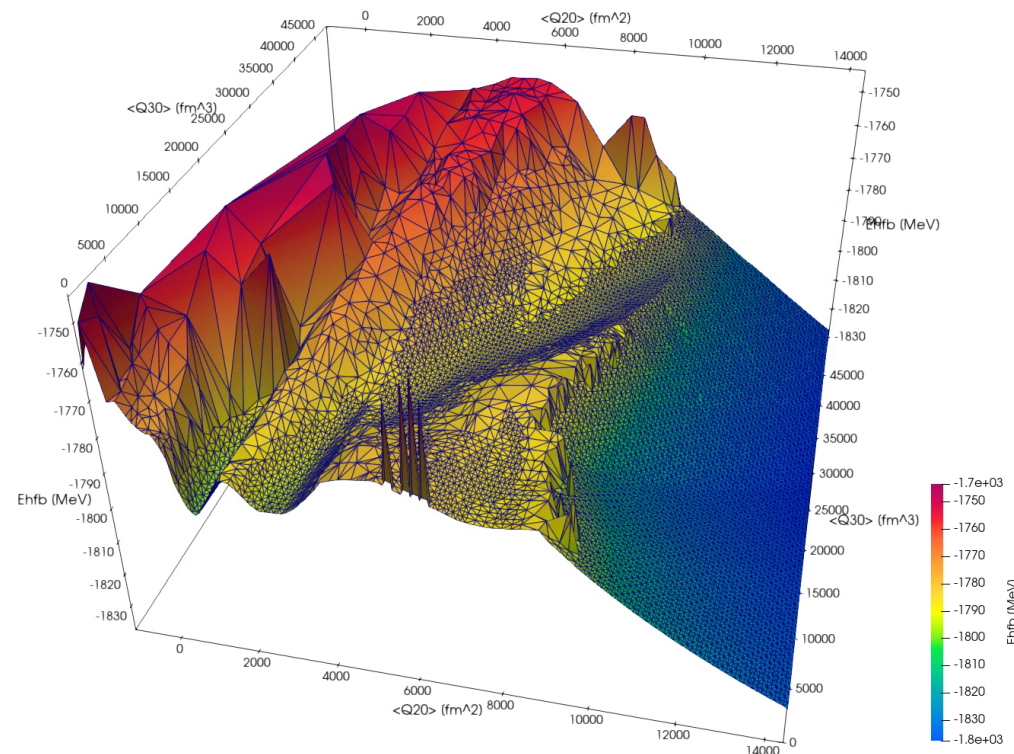
- Propagation techniques
 -  Faster convergence
- Retropropagation techniques
 -  Resolve some local minimas
 -  Ensure continuity in energy

What is CHICON ?

- A **ND-PES generator** written in python
- Developed at CEA by **N. Dubray** & **J. Newsome**
- HPC: MPI + Multiprocess parallelization
- Uses the HFB3 solver

Systematic production for IAEA: RIPL-4

- 1000 PES currently, 20 000 000 cpu/hour
-  Odd-nuclei PES still in progress



^{239}U (q_{20}, q_{30}) PES with mesh lines visible

Techniques

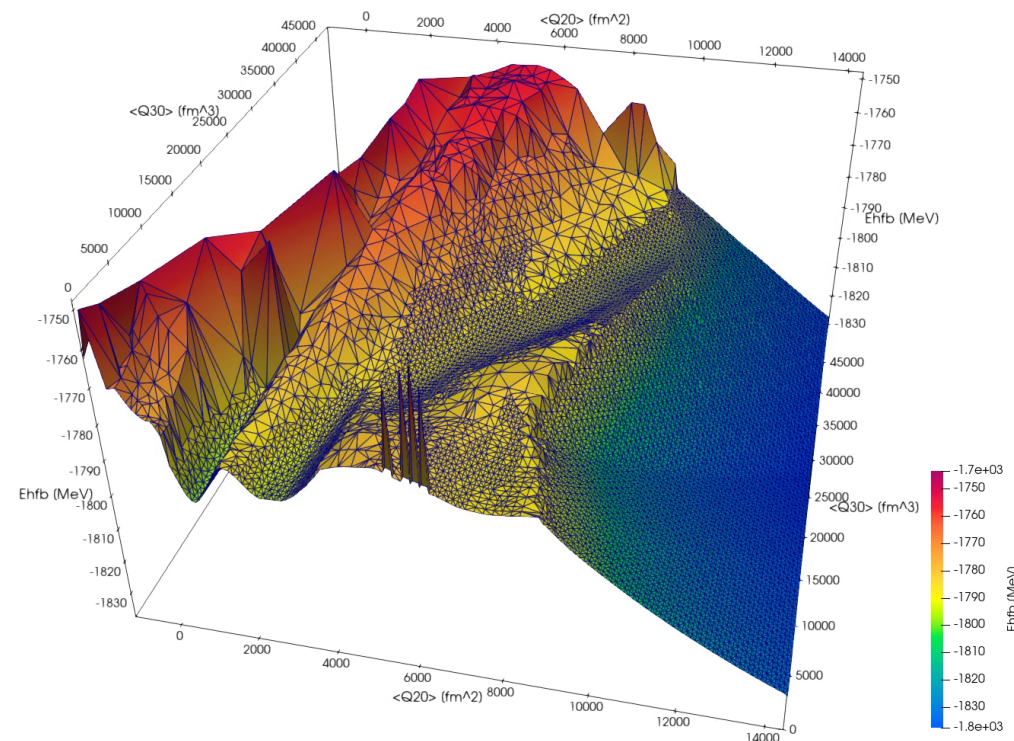
- Propagation techniques
 -  Faster convergence
- Retropropagation techniques
 -  Resolve some local minimas
 -  Ensure continuity in energy

What is CHICON ?

- A **ND-PES generator** written in python
- Developed at CEA by **N. Dubray** & **J. Newsome**
- HPC: MPI + Multiprocess parallelization
- Uses the HFB3 solver

Systematic production for IAEA: RIPL-4

- 1000 PES currently, 20 000 000 cpu/hour
-  Odd-nuclei PES still in progress



^{239}U (q_{20}, q_{30}) PES with mesh lines visible

Techniques

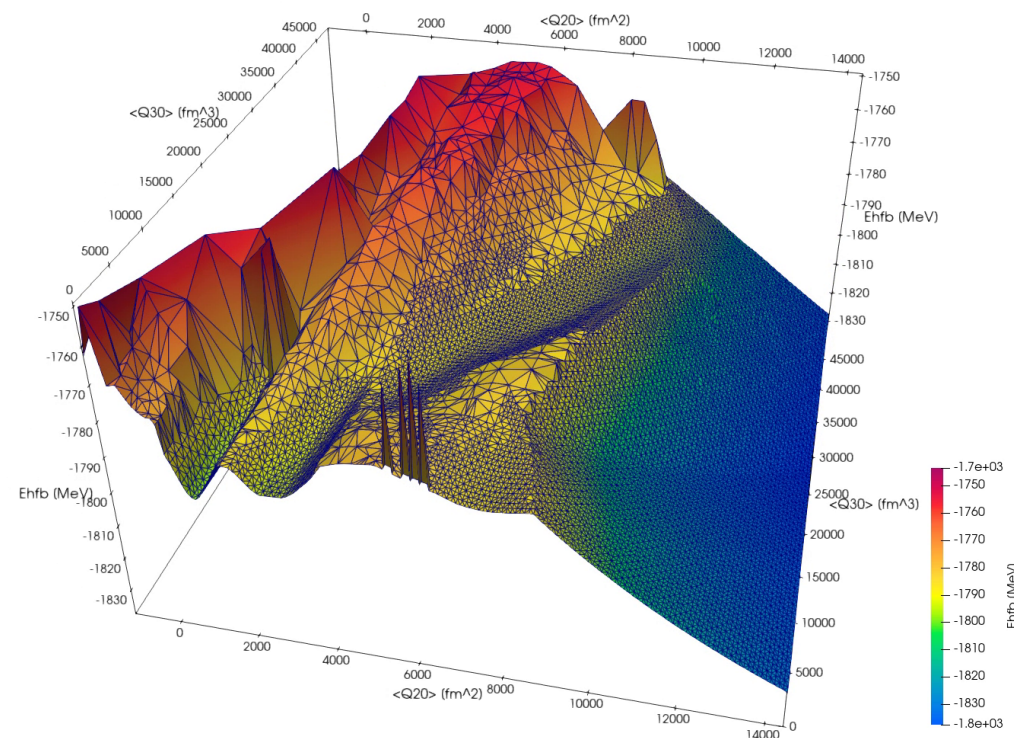
- Propagation techniques
 -  Faster convergence
- Retropropagation techniques
 -  Resolve some local minimas
 -  Ensure continuity in energy

What is CHICON ?

- A **ND-PES generator** written in python
- Developed at CEA by **N. Dubray** & **J. Newsome**
- HPC: MPI + Multiprocess parallelization
- Uses the HFB3 solver

Systematic production for IAEA: RIPL-4

- 1000 PES currently, 20 000 000 cpu/hour
-  Odd-nuclei PES still in progress



^{239}U (q_{20}, q_{30}) PES with mesh lines visible

Techniques

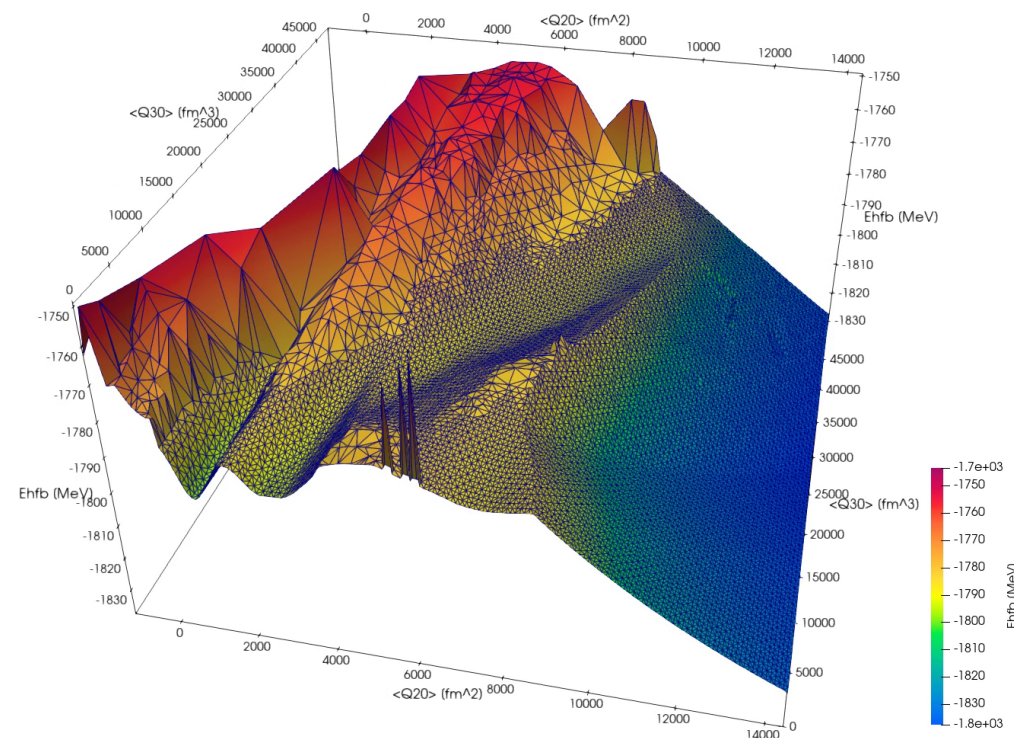
- Propagation techniques
 -  Faster convergence
- Retropropagation techniques
 -  Resolve some local minimas
 -  Ensure continuity in energy

What is CHICON ?

- A **ND-PES generator** written in python
- Developed at CEA by **N. Dubray** & **J. Newsome**
- HPC: MPI + Multiprocess parallelization
- Uses the HFB3 solver

Systematic production for IAEA: RIPL-4

- 1000 PES currently, 20 000 000 cpu/hour
-  Odd-nuclei PES still in progress



^{239}U (q_{20}, q_{30}) PES with mesh lines visible

Techniques

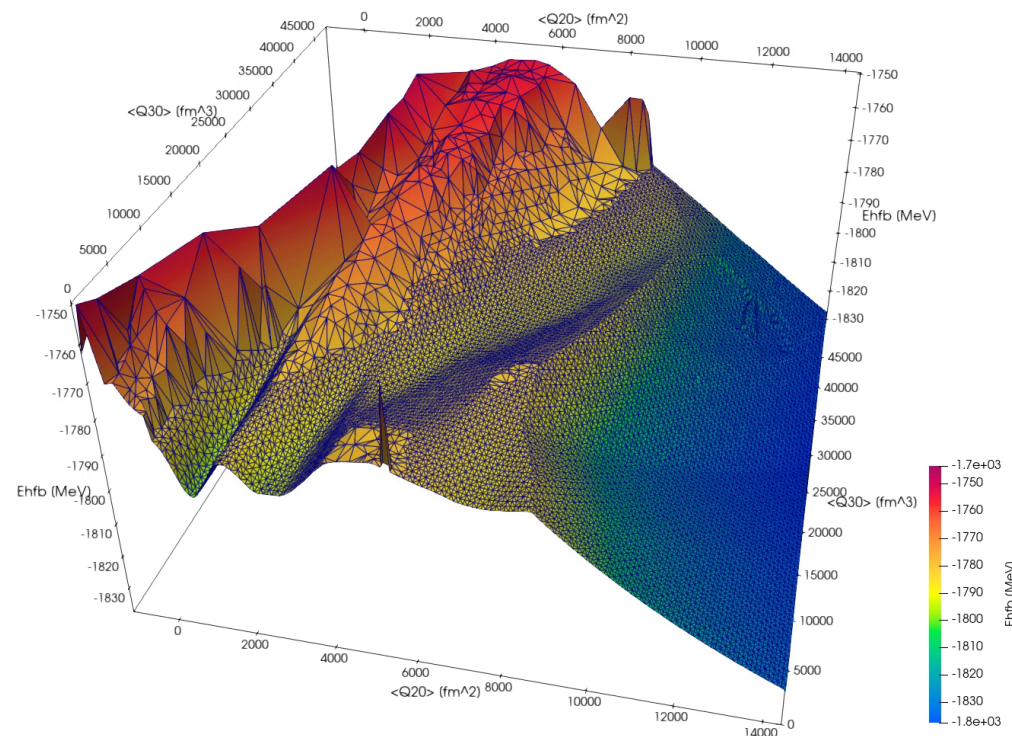
- Propagation techniques
 -  Faster convergence
- Retropropagation techniques
 -  Resolve some local minimas
 -  Ensure continuity in energy

What is CHICON ?

- A **ND-PES generator** written in python
- Developed at CEA by **N. Dubray** & **J. Newsome**
- HPC: MPI + Multiprocess parallelization
- Uses the HFB3 solver

Systematic production for IAEA: RIPL-4

- 1000 PES currently, 20 000 000 cpu/hour
-  Odd-nuclei PES still in progress



^{239}U (q_{20}, q_{30}) PES with mesh lines visible

Techniques

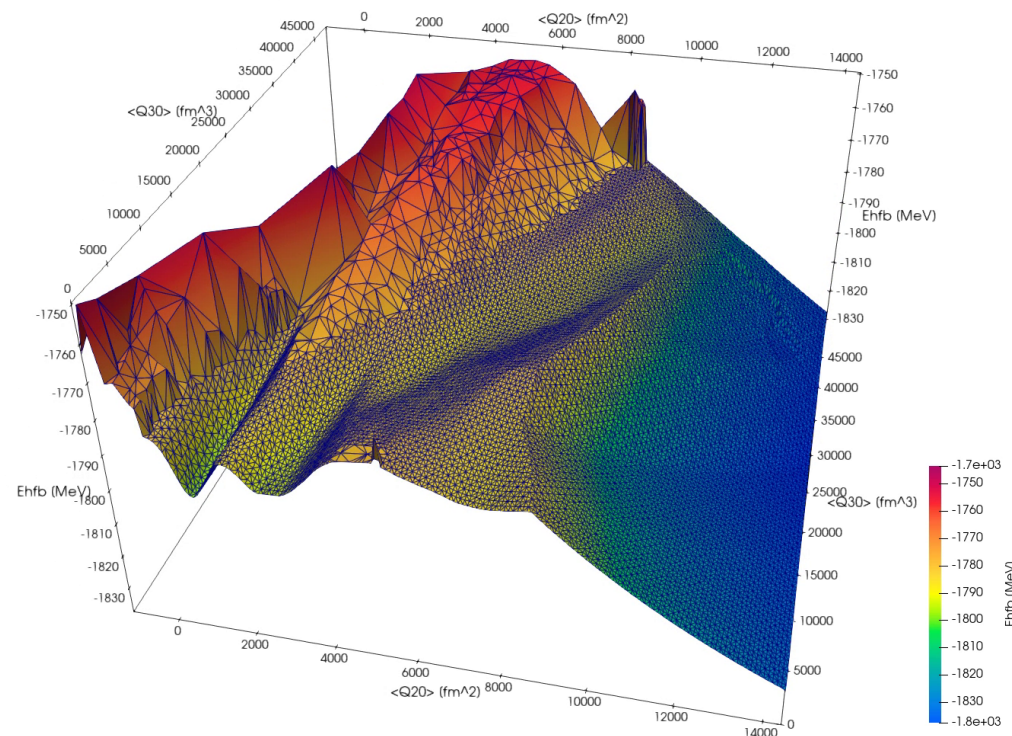
- Propagation techniques
 -  Faster convergence
- Retropropagation techniques
 -  Resolve some local minimas
 -  Ensure continuity in energy

What is CHICON ?

- A **ND-PES generator** written in python
- Developed at CEA by **N. Dubray** & **J. Newsome**
- HPC: MPI + Multiprocess parallelization
- Uses the HFB3 solver

Systematic production for IAEA: RIPL-4

- 1000 PES currently, 20 000 000 cpu/hour
-  Odd-nuclei PES still in progress



^{239}U (q_{20}, q_{30}) PES with mesh lines visible

Techniques

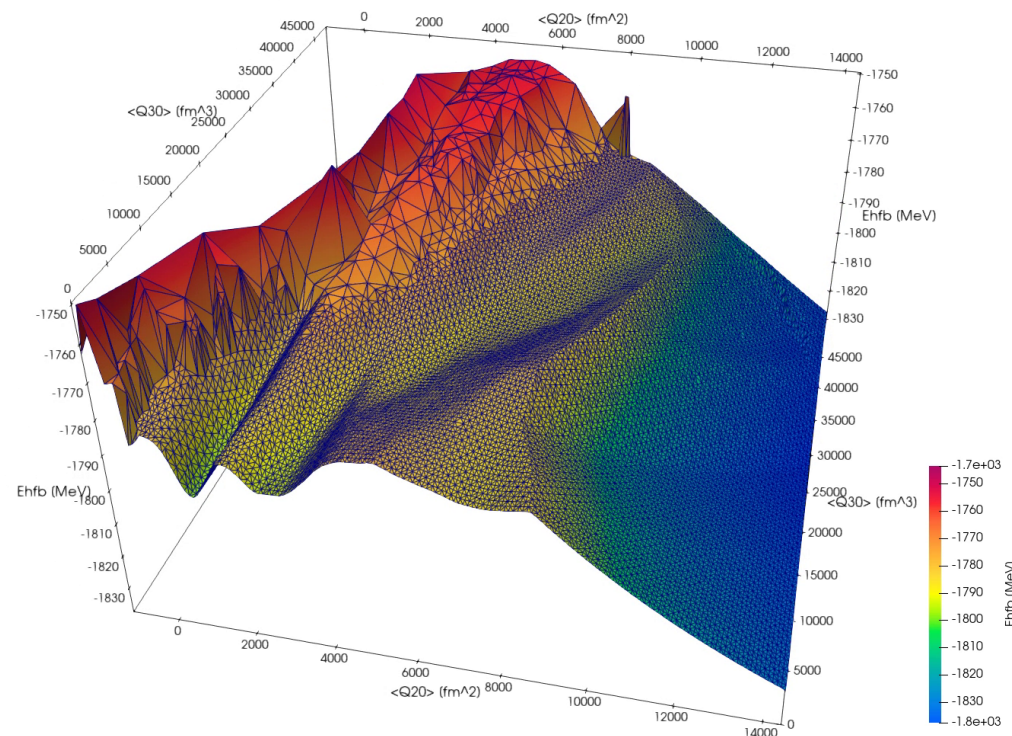
- Propagation techniques
 -  Faster convergence
- Retropropagation techniques
 -  Resolve some local minimas
 -  Ensure continuity in energy

What is CHICON ?

- A **ND-PES generator** written in python
- Developed at CEA by **N. Dubray** & **J. Newsome**
- HPC: MPI + Multiprocess parallelization
- Uses the HFB3 solver

Systematic production for IAEA: RIPL-4

- 1000 PES currently, 20 000 000 cpu/hour
-  Odd-nuclei PES still in progress



^{239}U (q_{20}, q_{30}) PES with mesh lines visible

Techniques

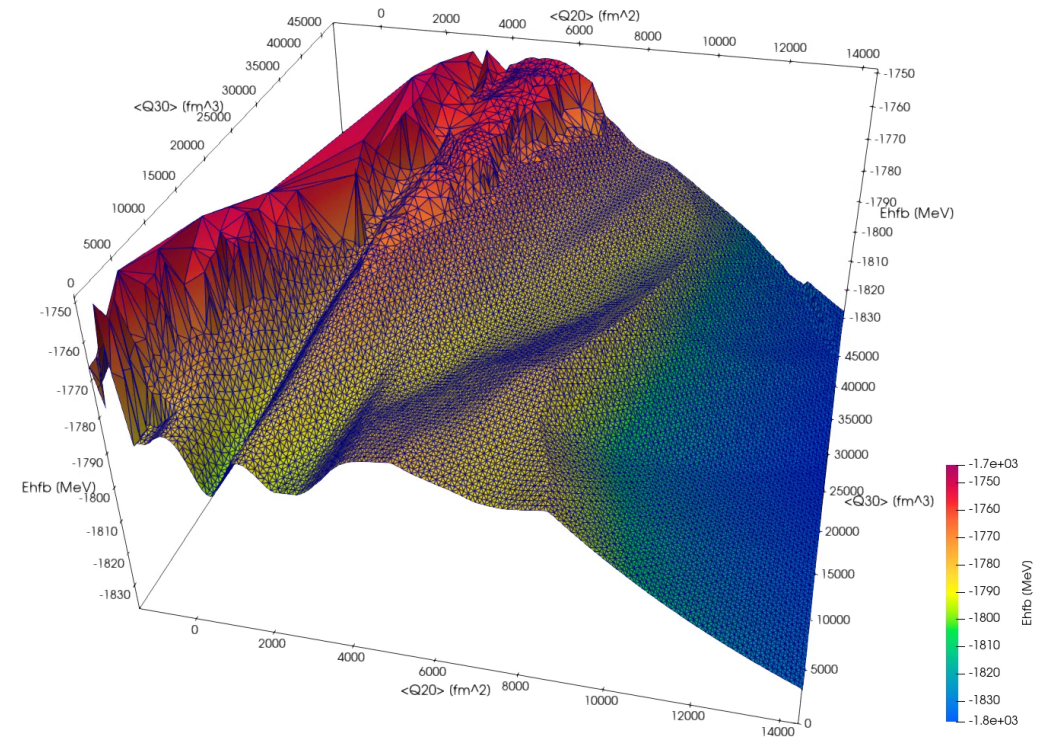
- Propagation techniques
 -  Faster convergence
- Retropropagation techniques
 -  Resolve some local minimas
 -  Ensure continuity in energy

What is CHICON ?

- A **ND-PES generator** written in python
- Developed at CEA by **N. Dubray** & **J. Newsome**
- HPC: MPI + Multiprocess parallelization
- Uses the HFB3 solver

Systematic production for IAEA: RIPL-4

- 1000 PES currently, 20 000 000 cpu/hour
-  Odd-nuclei PES still in progress



^{239}U (q_{20}, q_{30}) PES with mesh lines visible

Techniques

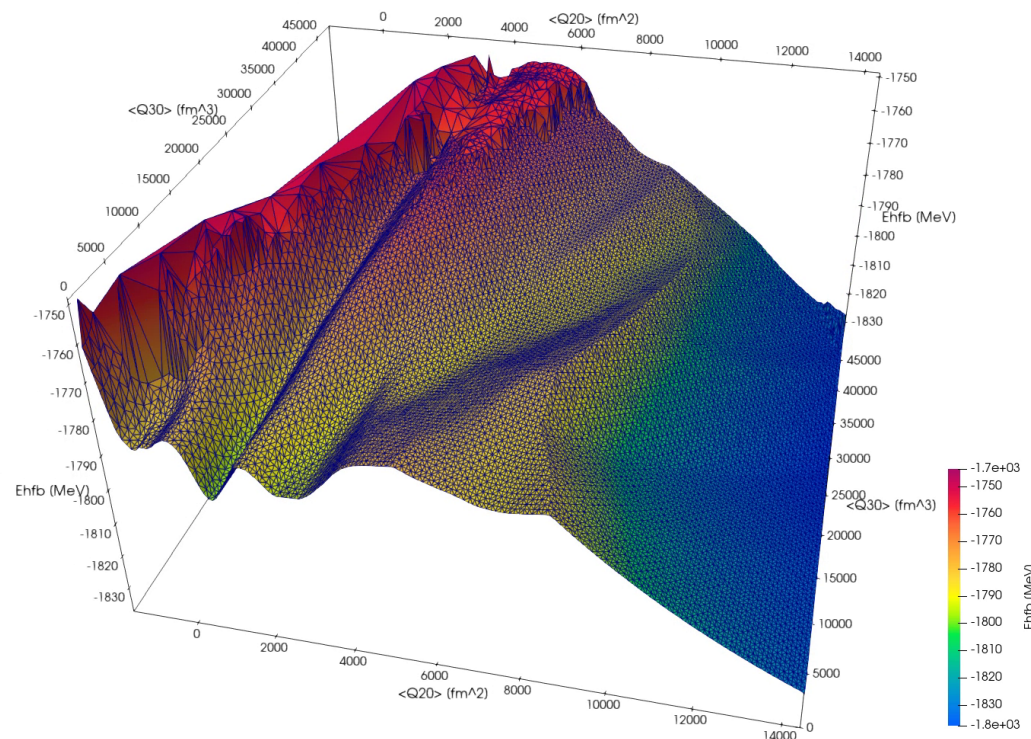
- Propagation techniques
 -  Faster convergence
- Retropropagation techniques
 -  Resolve some local minimas
 -  Ensure continuity in energy

What is CHICON ?

- A **ND-PES generator** written in python
- Developed at CEA by **N. Dubray** & **J. Newsome**
- HPC: MPI + Multiprocess parallelization
- Uses the HFB3 solver

Systematic production for IAEA: RIPL-4

- 1000 PES currently, 20 000 000 cpu/hour
-  Odd-nuclei PES still in progress



^{239}U (q_{20}, q_{30}) PES with mesh lines visible

Techniques

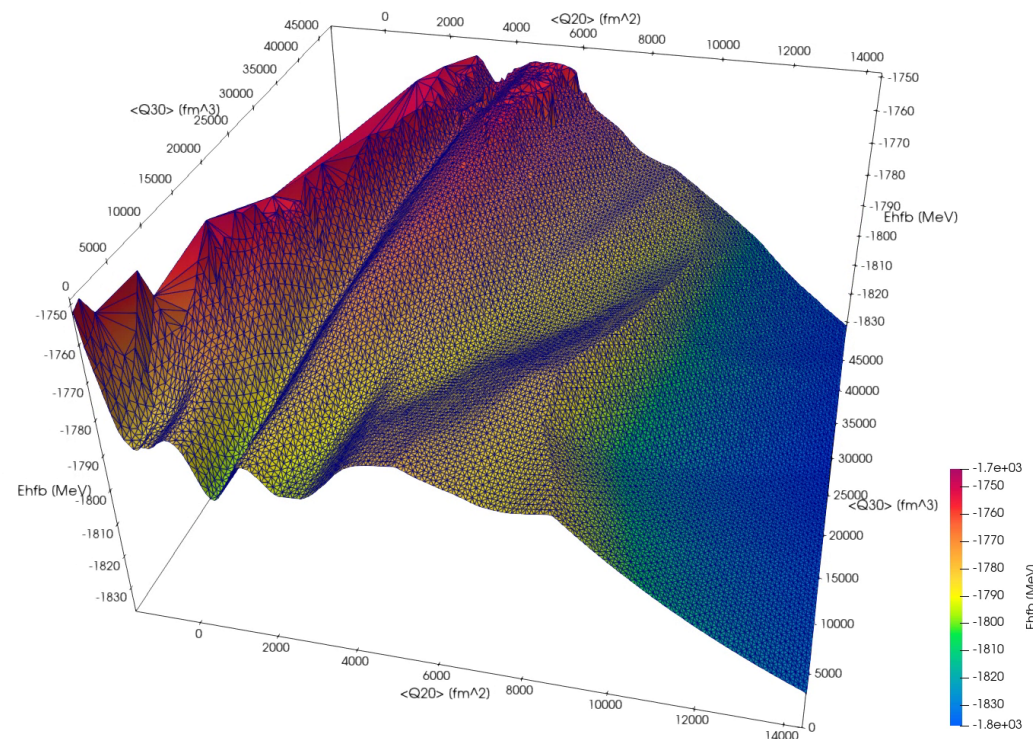
- Propagation techniques
 -  Faster convergence
- Retropropagation techniques
 -  Resolve some local minimas
 -  Ensure continuity in energy

What is CHICON ?

- A **ND-PES generator** written in python
- Developed at CEA by **N. Dubray** & **J. Newsome**
- HPC: MPI + Multiprocess parallelization
- Uses the HFB3 solver

Systematic production for IAEA: RIPL-4

- 1000 PES currently, 20 000 000 cpu/hour
-  Odd-nuclei PES still in progress



^{239}U (q_{20}, q_{30}) PES with mesh lines visible

Techniques

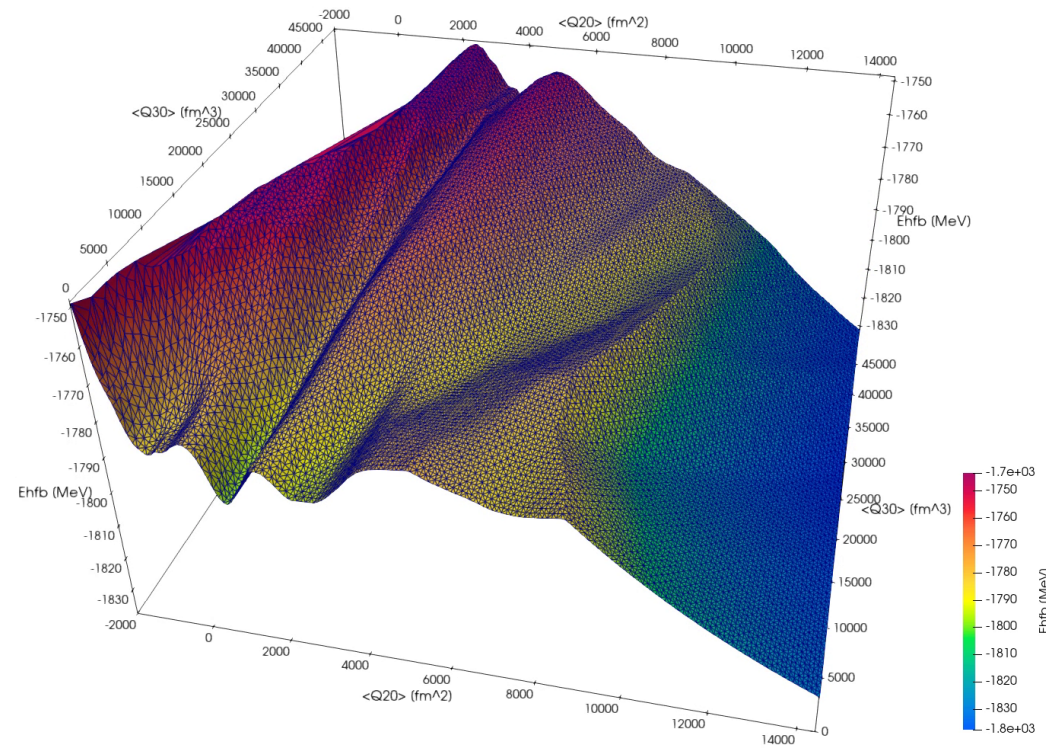
- Propagation techniques
 -  Faster convergence
- Retropropagation techniques
 -  Resolve some local minimas
 -  Ensure continuity in energy

What is CHICON ?

- A **ND-PES generator** written in python
- Developed at CEA by **N. Dubray** & **J. Newsome**
- HPC: MPI + Multiprocess parallelization
- Uses the HFB3 solver

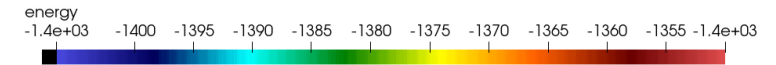
Systematic production for IAEA: RIPL-4

- 1000 PES currently, 20 000 000 cpu/hour
-  Odd-nuclei PES still in progress



^{239}U (q_{20}, q_{30}) PES with mesh lines visible

Recent developments

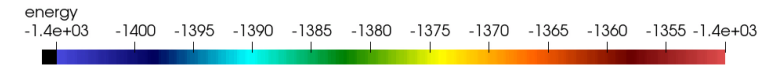


CHICON 3D

- Developed at CEA by J. Newsome, **B. Martinez** & N. Dubray
- Full rewrite of **CHICON** with a more scalable structure
- 🚀 3D PES are now possible
- MPI-IO / Database sharding

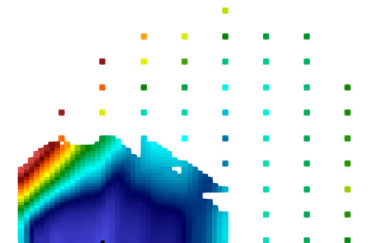


Recent developments

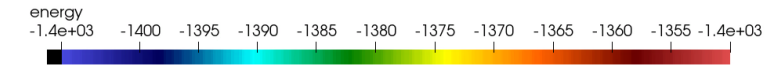


CHICON 3D

- Developed at CEA by J. Newsome, **B. Martinez** & N. Dubray
- Full rewrite of **CHICON** with a more scalable structure
- 🚀 3D PES are now possible
- MPI-IO / Database sharding

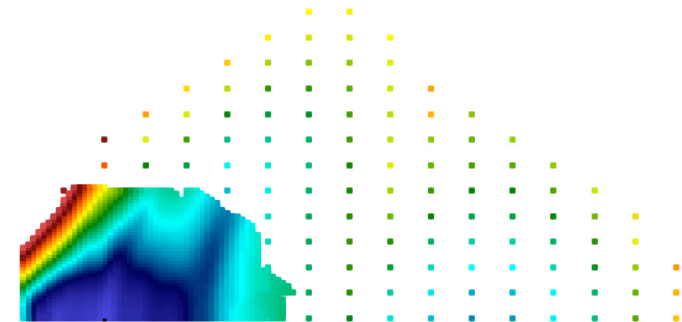


Recent developments



CHICON 3D

- Developed at CEA by J. Newsome, **B. Martinez** & N. Dubray
- Full rewrite of **CHICON** with a more scalable structure
- 🚀 3D PES are now possible
- MPI-IO / Database sharding

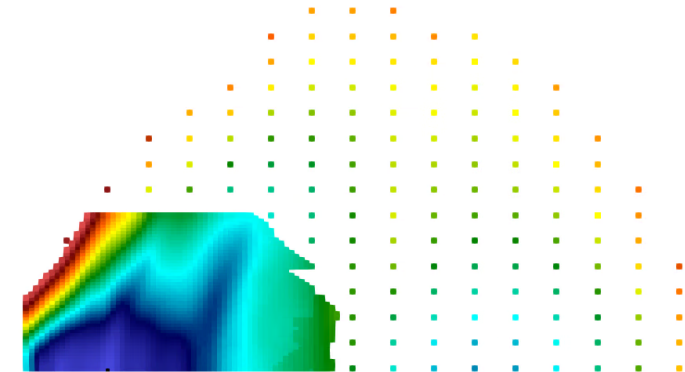
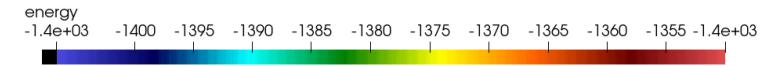


Recent developments



CHICON 3D

- Developed at CEA by J. Newsome, **B. Martinez** & N. Dubray
- Full rewrite of **CHICON** with a more scalable structure
- 🚀 3D PES are now possible
- MPI-IO / Database sharding

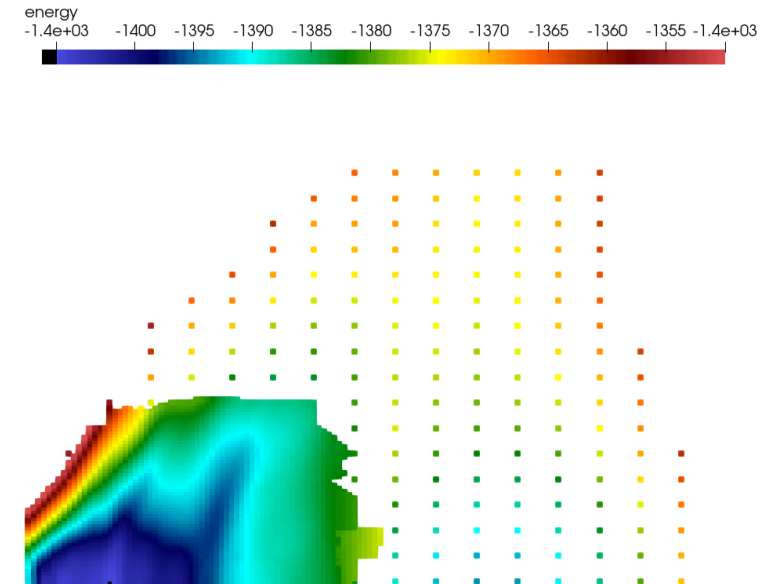


Recent developments



CHICON 3D

- Developed at CEA by J. Newsome, **B. Martinez** & N. Dubray
- Full rewrite of **CHICON** with a more scalable structure
- 🚧 3D PES are now possible
- MPI-IO / Database sharding

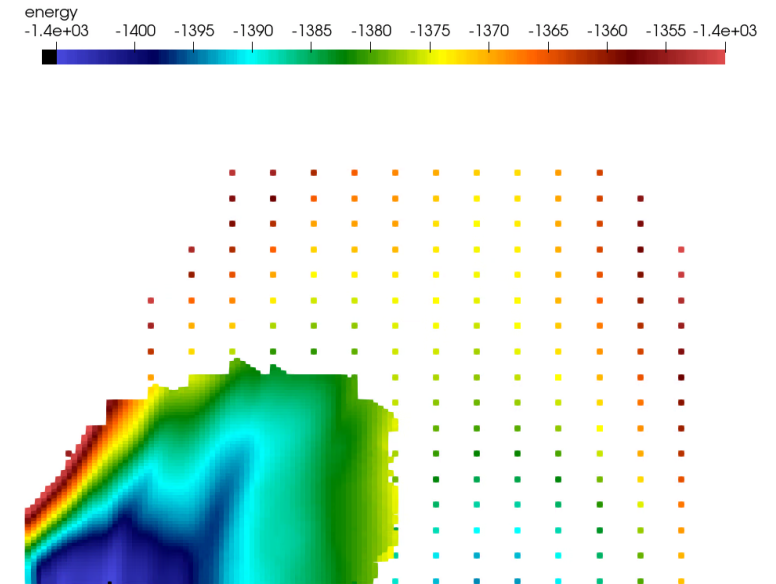


Recent developments



CHICON 3D

- Developed at CEA by J. Newsome, **B. Martinez** & N. Dubray
- Full rewrite of **CHICON** with a more scalable structure
- 🚀 3D PES are now possible
- MPI-IO / Database sharding

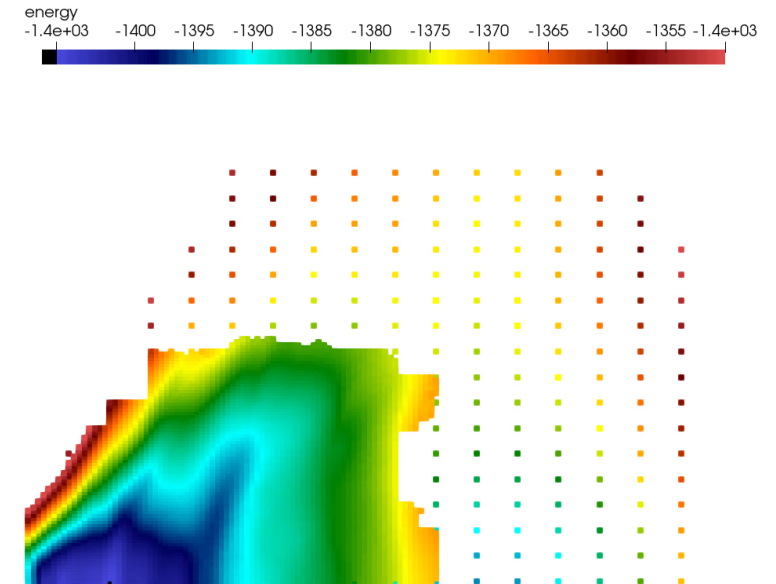


Recent developments



CHICON 3D

- Developed at CEA by J. Newsome, **B. Martinez** & N. Dubray
- Full rewrite of **CHICON** with a more scalable structure
- 🚧 3D PES are now possible
- MPI-IO / Database sharding

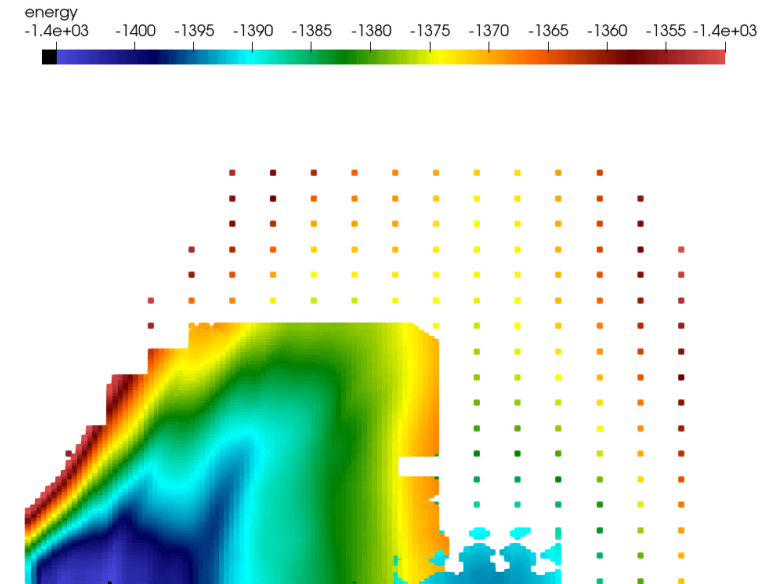


Recent developments



CHICON 3D

- Developed at CEA by J. Newsome, **B. Martinez** & N. Dubray
- Full rewrite of **CHICON** with a more scalable structure
- 🚧 3D PES are now possible
- MPI-IO / Database sharding

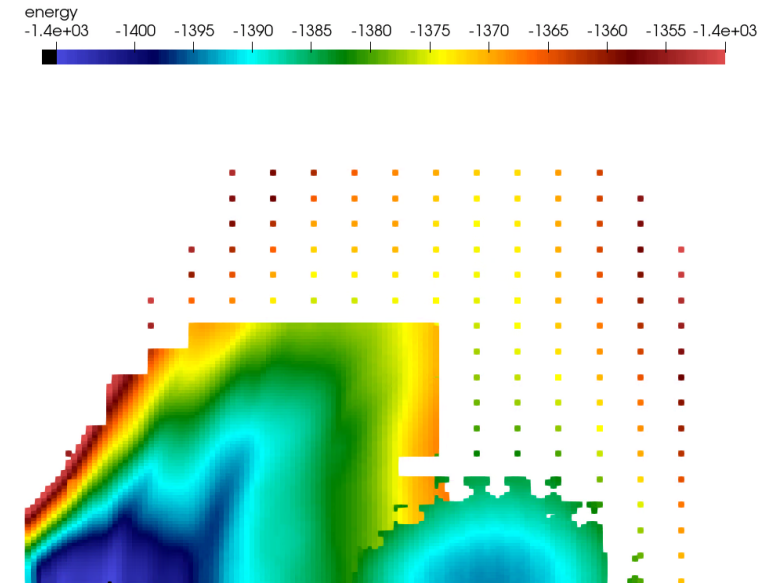


Recent developments



CHICON 3D

- Developed at CEA by J. Newsome, **B. Martinez** & N. Dubray
- Full rewrite of **CHICON** with a more scalable structure
- 🚧 3D PES are now possible
- MPI-IO / Database sharding

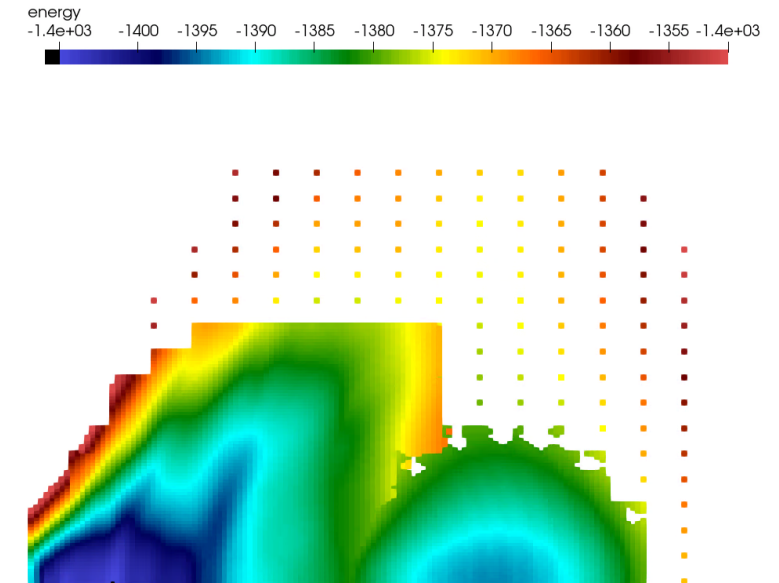


Recent developments



CHICON 3D

- Developed at CEA by J. Newsome, **B. Martinez** & N. Dubray
- Full rewrite of **CHICON** with a more scalable structure
- 🚧 3D PES are now possible
- MPI-IO / Database sharding

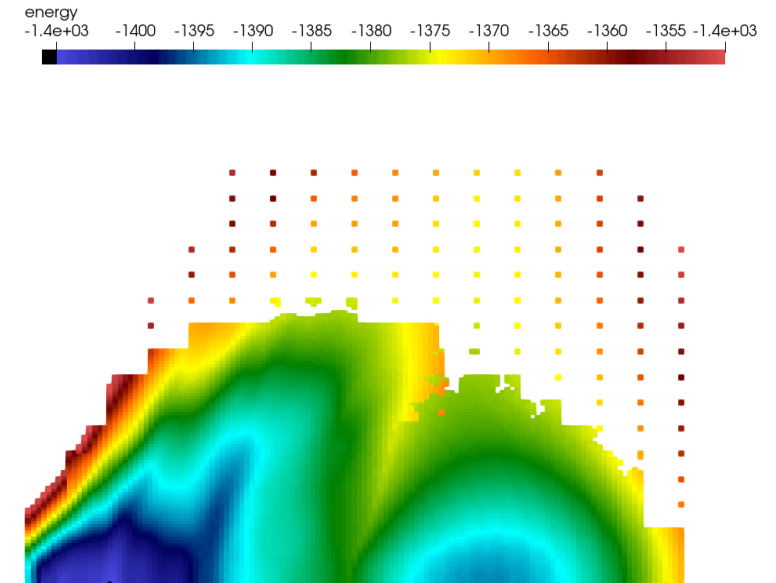


Recent developments



CHICON 3D

- Developed at CEA by J. Newsome, **B. Martinez** & N. Dubray
- Full rewrite of **CHICON** with a more scalable structure
- 🚧 3D PES are now possible
- MPI-IO / Database sharding

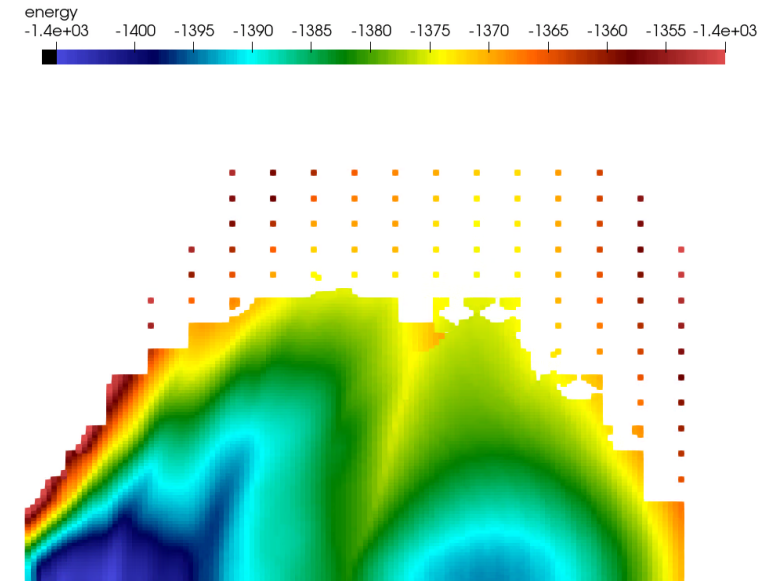


Recent developments



CHICON 3D

- Developed at CEA by J. Newsome, **B. Martinez** & N. Dubray
- Full rewrite of **CHICON** with a more scalable structure
- 🚧 3D PES are now possible
- MPI-IO / Database sharding

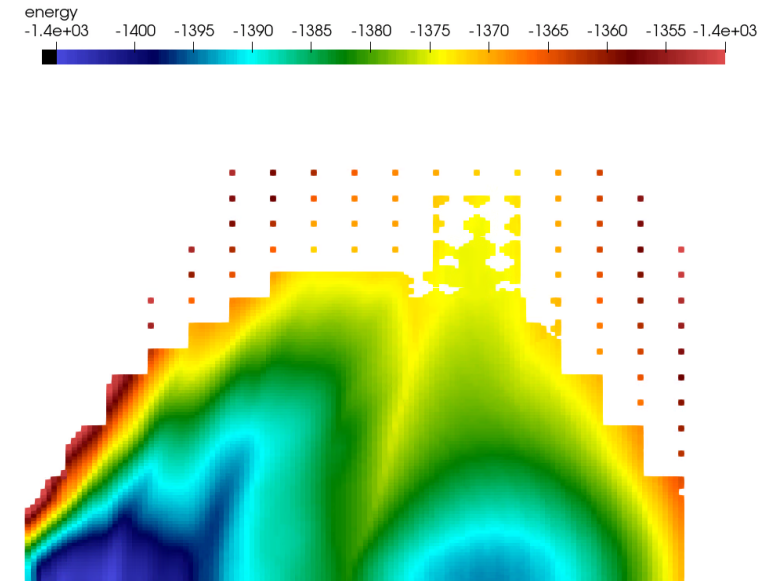


Recent developments



CHICON 3D

- Developed at CEA by J. Newsome, **B. Martinez** & N. Dubray
- Full rewrite of **CHICON** with a more scalable structure
- 🎨 3D PES are now possible
- MPI-IO / Database sharding



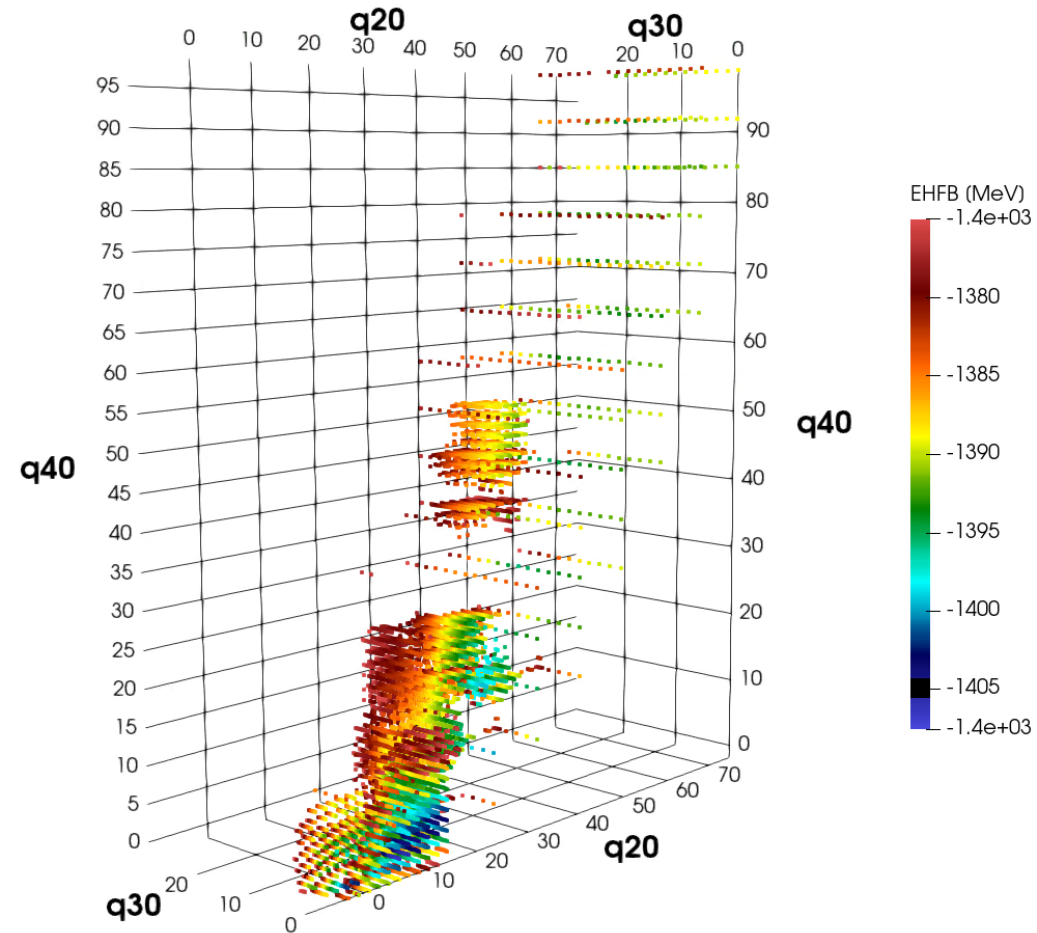
3D PES

^{180}Hg 3D PES

- 300 000 HFB calculations
- 12 500 HFB points in the PES
- 2ct optimized axial HO basis

Run time

- 3 days
 - 16 nodes (128 cpu per node)
- ➔ 150 000 cpu/h



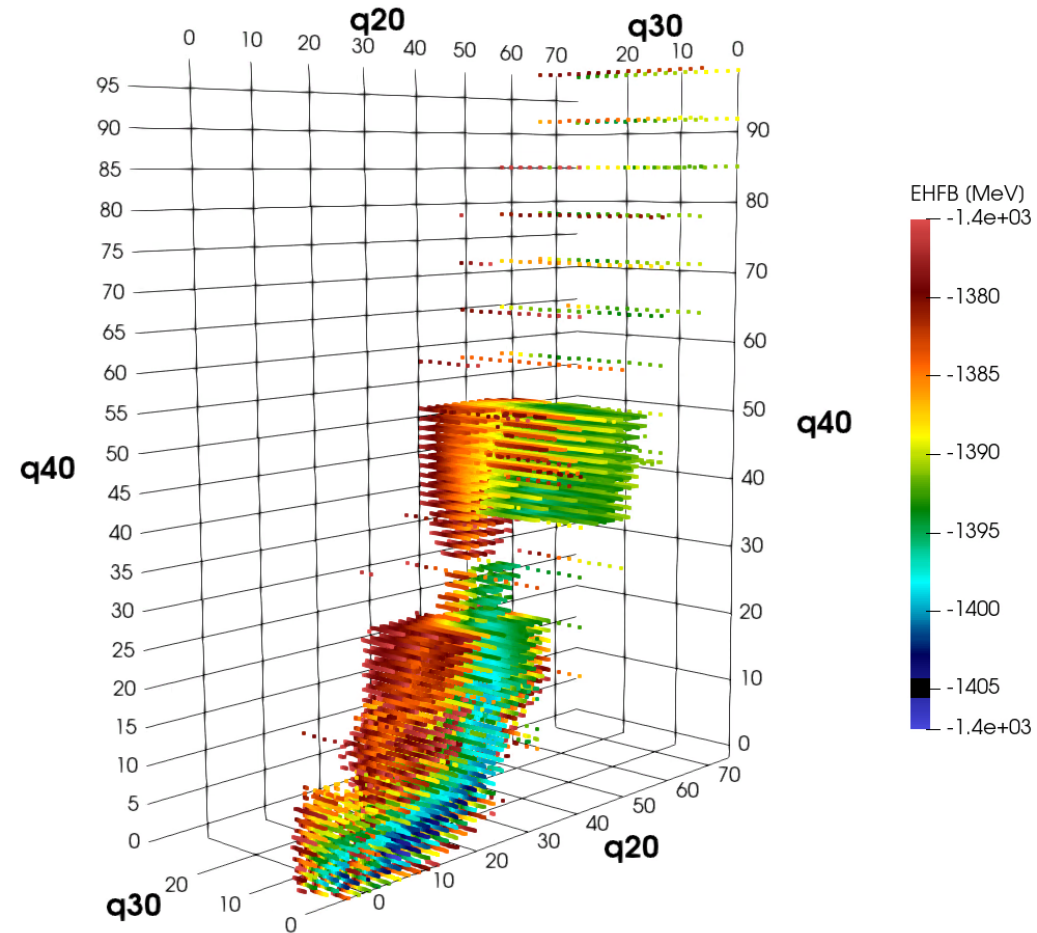
3D PES

^{180}Hg 3D PES

- 300 000 HFB calculations
- 12 500 HFB points in the PES
- 2ct optimized axial HO basis

Run time

- 3 days
 - 16 nodes (128 cpu per node)
- ➔ 150 000 cpu/h



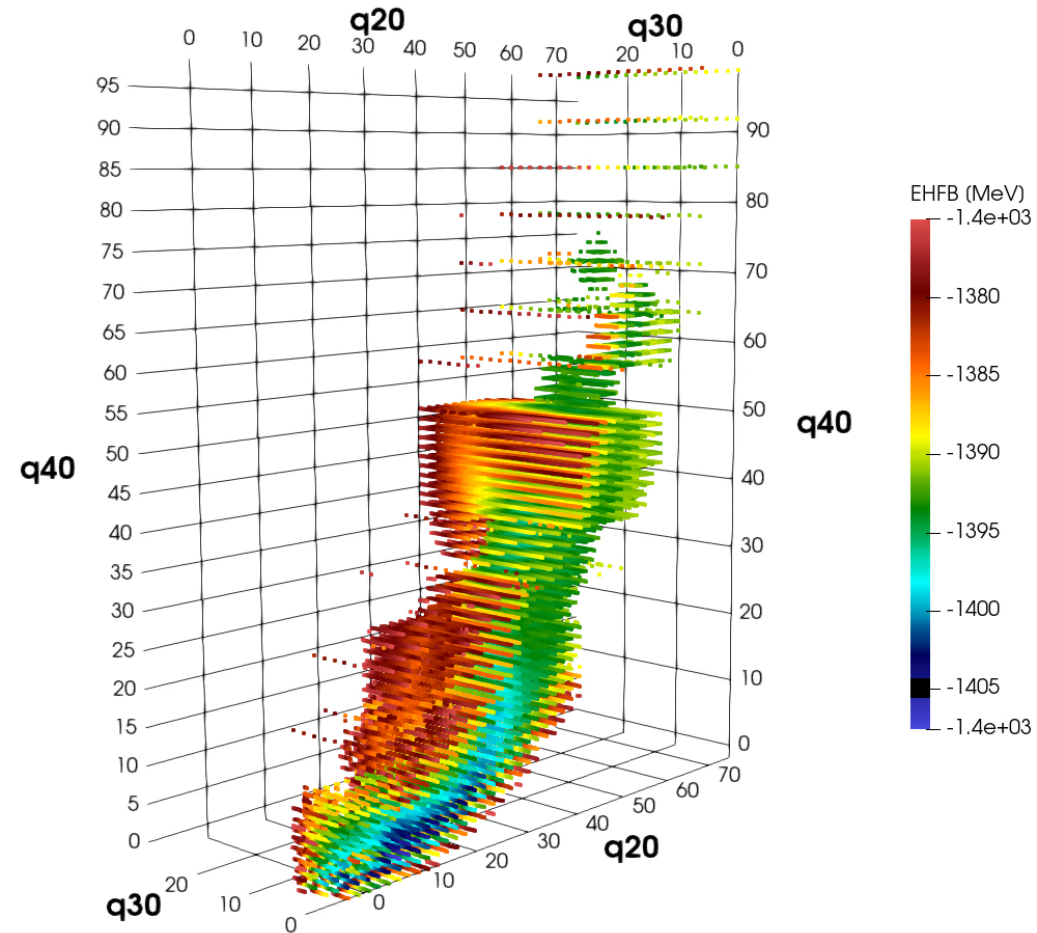
3D PES

^{180}Hg 3D PES

- 300 000 HFB calculations
- 12 500 HFB points in the PES
- 2ct optimized axial HO basis

Run time

- 3 days
 - 16 nodes (128 cpu per node)
- ➔ 150 000 cpu/h



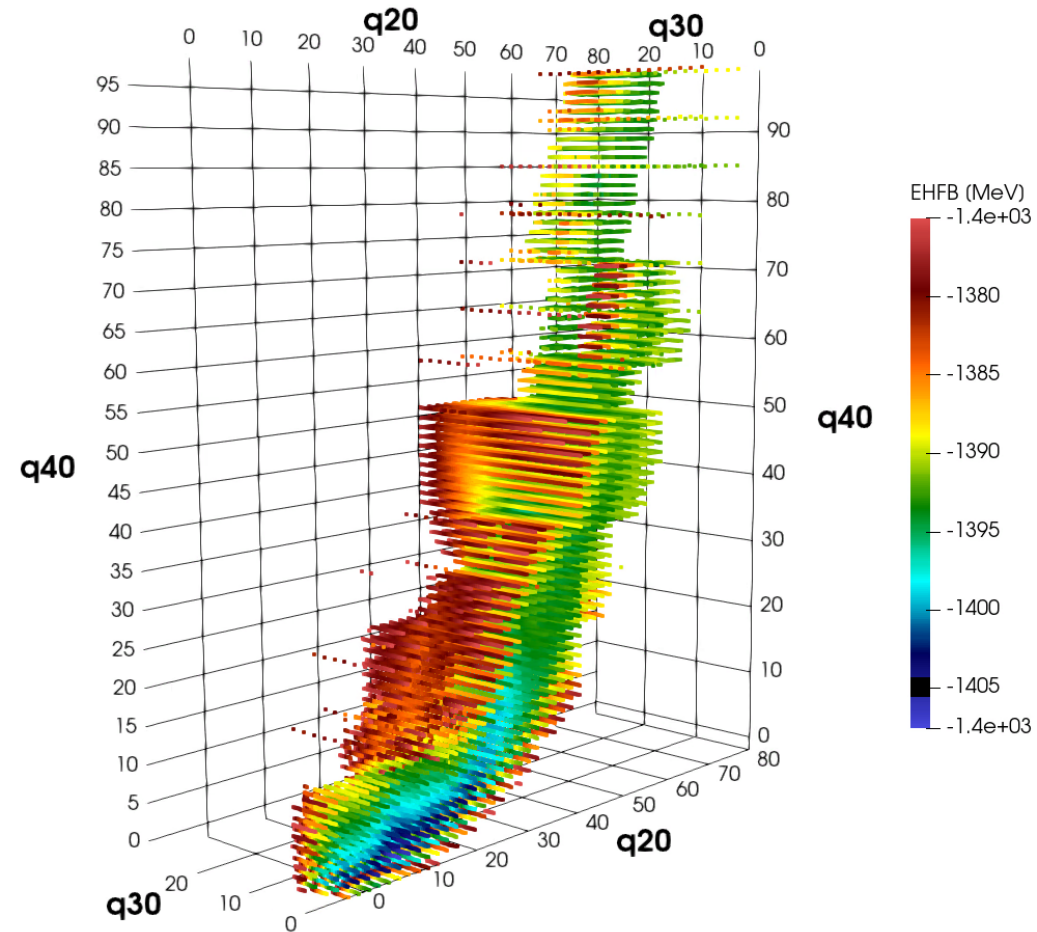
3D PES

^{180}Hg 3D PES

- 300 000 HFB calculations
- 12 500 HFB points in the PES
- 2ct optimized axial HO basis

Run time

- 3 days
 - 16 nodes (128 cpu per node)
- ➔ 150 000 cpu/h



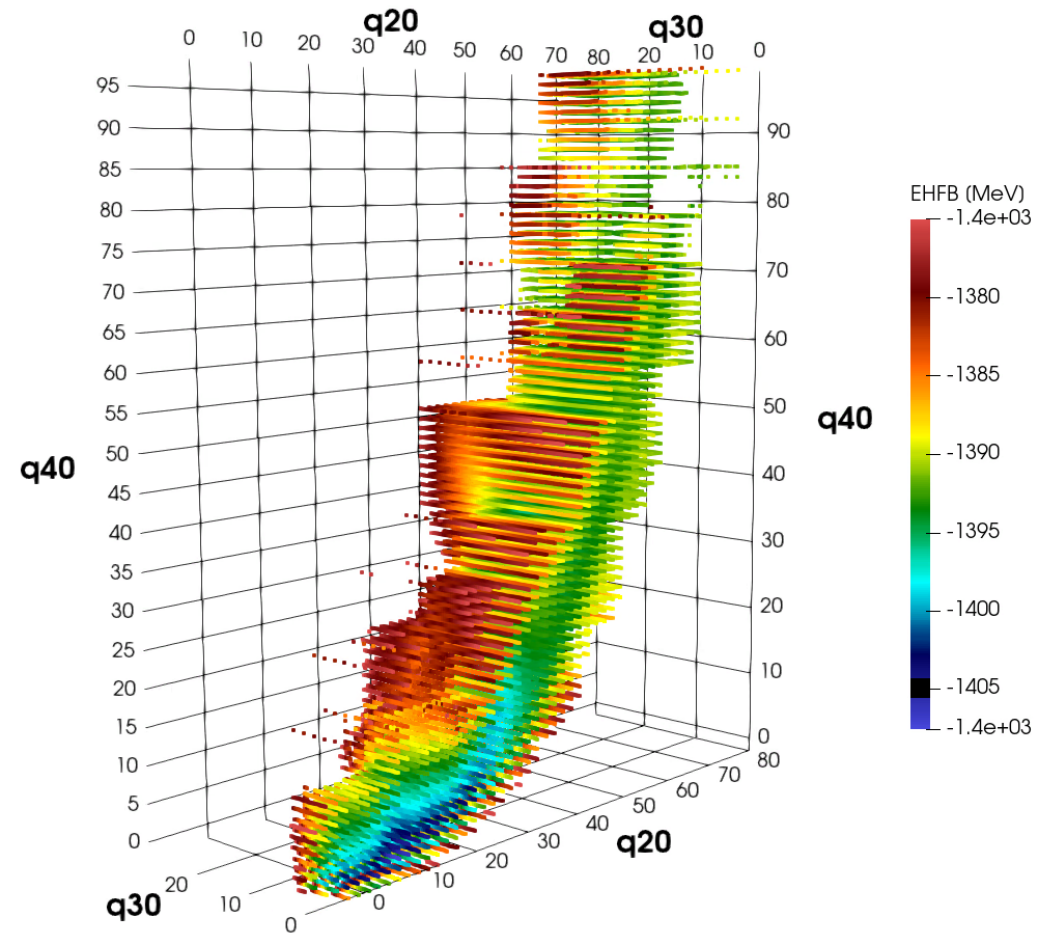
3D PES

^{180}Hg 3D PES

- 300 000 HFB calculations
- 12 500 HFB points in the PES
- 2ct optimized axial HO basis

Run time

- 3 days
 - 16 nodes (128 cpu per node)
- ➔ 150 000 cpu/h



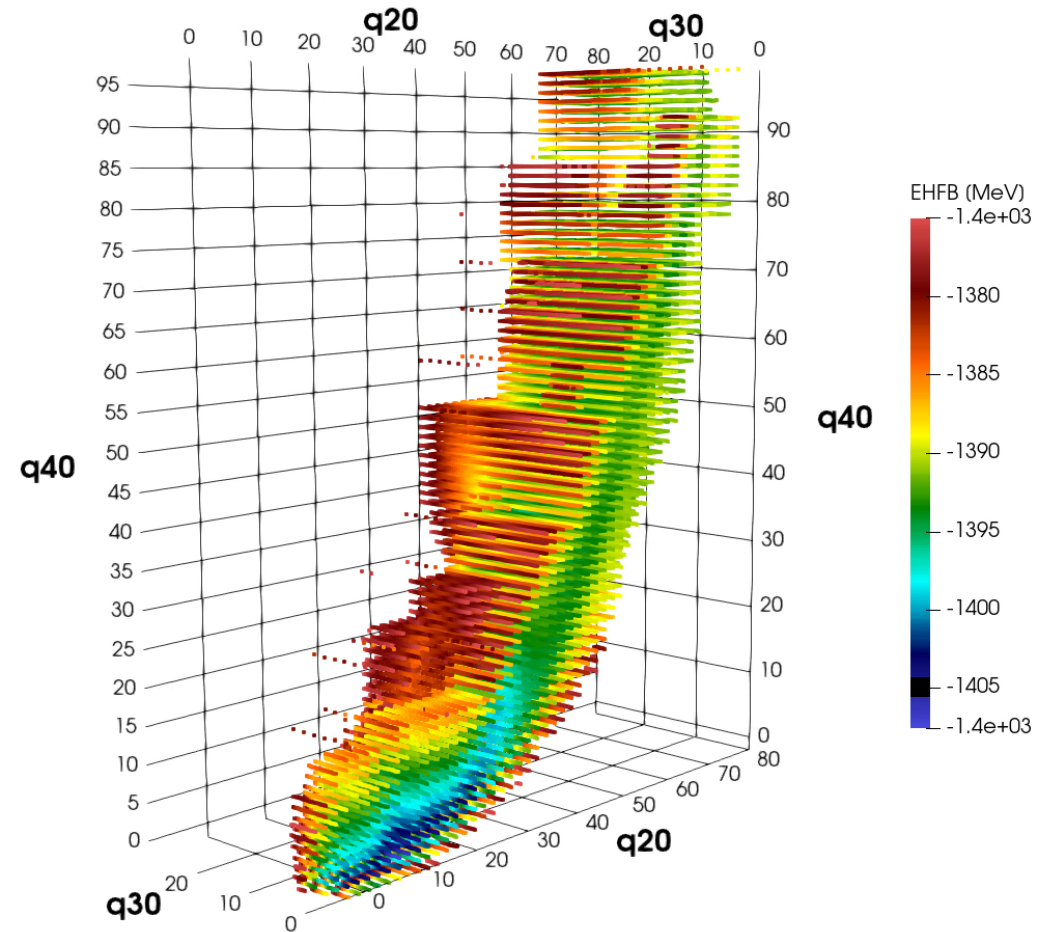
3D PES

^{180}Hg 3D PES

- 300 000 HFB calculations
- 12 500 HFB points in the PES
- 2ct optimized axial HO basis

Run time

- 3 days
 - 16 nodes (128 cpu per node)
- ➔ 150 000 cpu/h



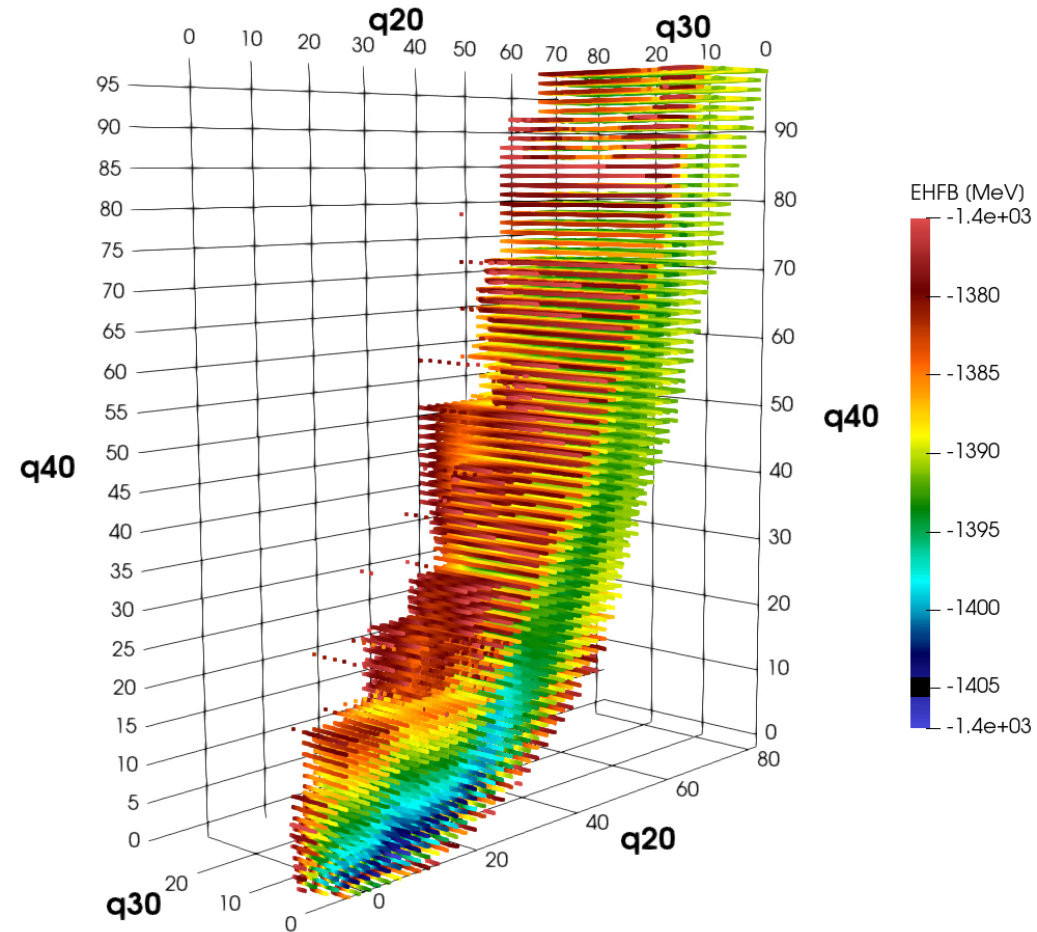
3D PES

^{180}Hg 3D PES

- 300 000 HFB calculations
- 12 500 HFB points in the PES
- 2ct optimized axial HO basis

Run time

- 3 days
 - 16 nodes (128 cpu per node)
- ➔ 150 000 cpu/h



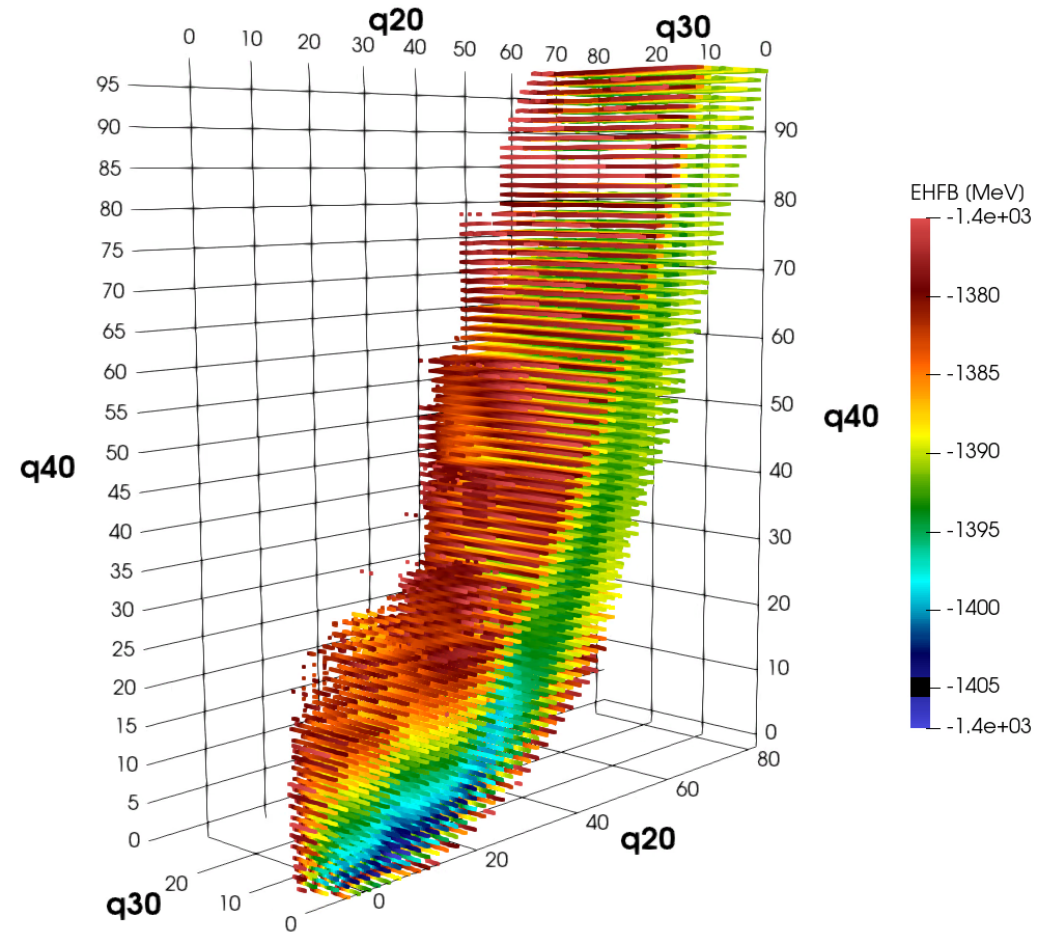
3D PES

^{180}Hg 3D PES

- 300 000 HFB calculations
- 12 500 HFB points in the PES
- 2ct optimized axial HO basis

Run time

- 3 days
 - 16 nodes (128 cpu per node)
- ➔ 150 000 cpu/h



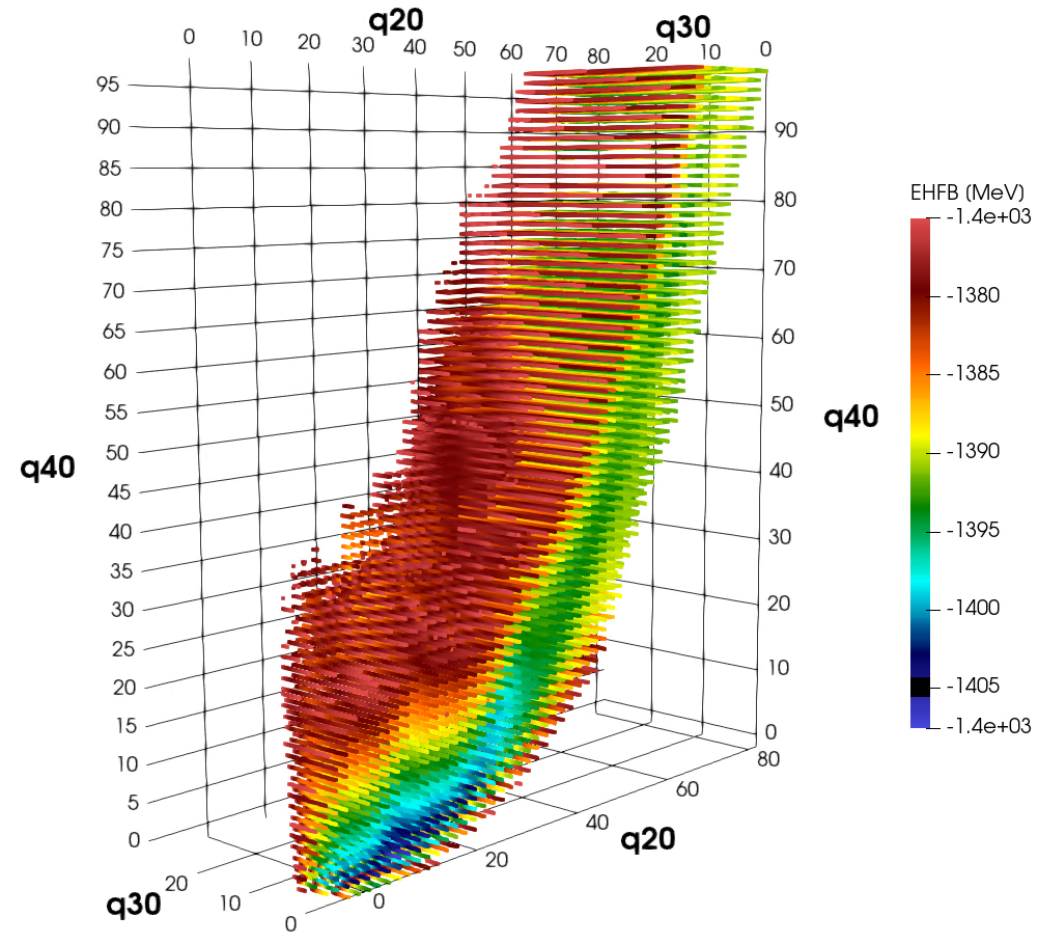
3D PES

^{180}Hg 3D PES

- 300 000 HFB calculations
- 12 500 HFB points in the PES
- 2ct optimized axial HO basis

Run time

- 3 days
 - 16 nodes (128 cpu per node)
- ➔ 150 000 cpu/h



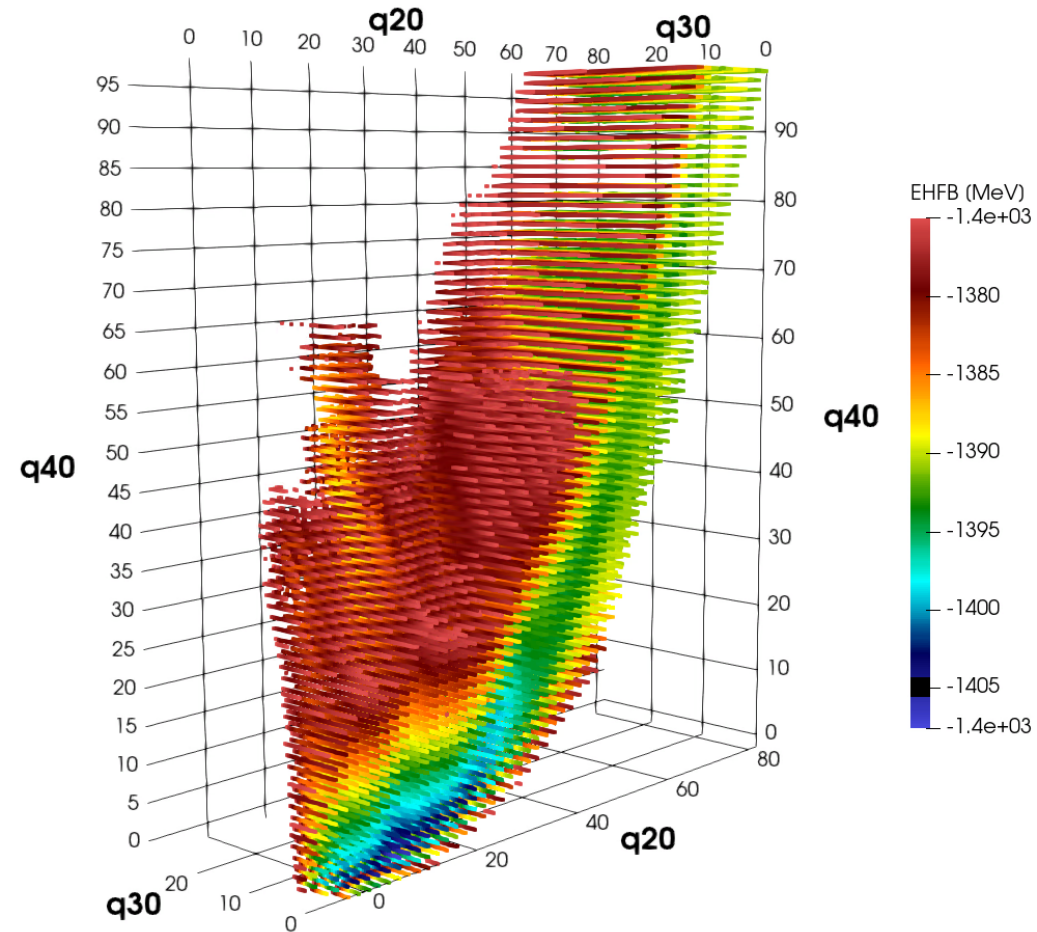
3D PES

^{180}Hg 3D PES

- 300 000 HFB calculations
- 12 500 HFB points in the PES
- 2ct optimized axial HO basis

Run time

- 3 days
 - 16 nodes (128 cpu per node)
- ➔ 150 000 cpu/h



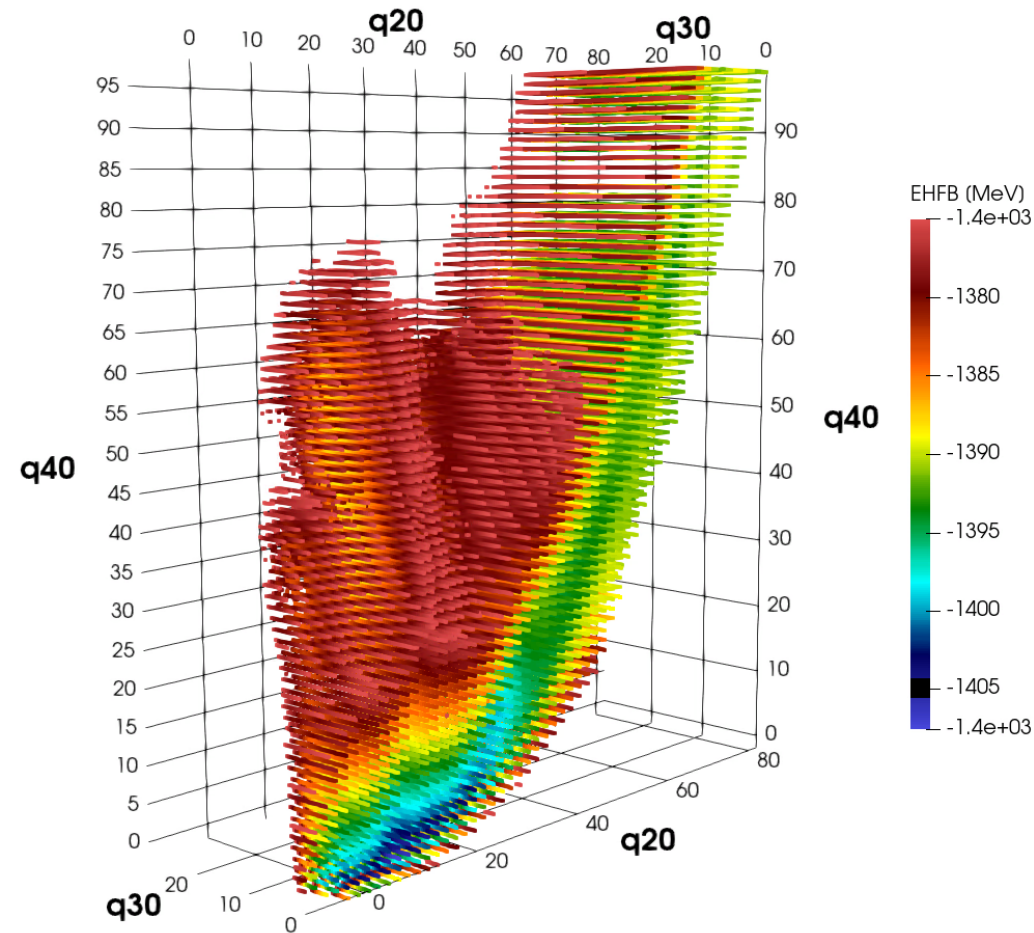
3D PES

^{180}Hg 3D PES

- 300 000 HFB calculations
- 12 500 HFB points in the PES
- 2ct optimized axial HO basis

Run time

- 3 days
 - 16 nodes (128 cpu per node)
- ➔ 150 000 cpu/h



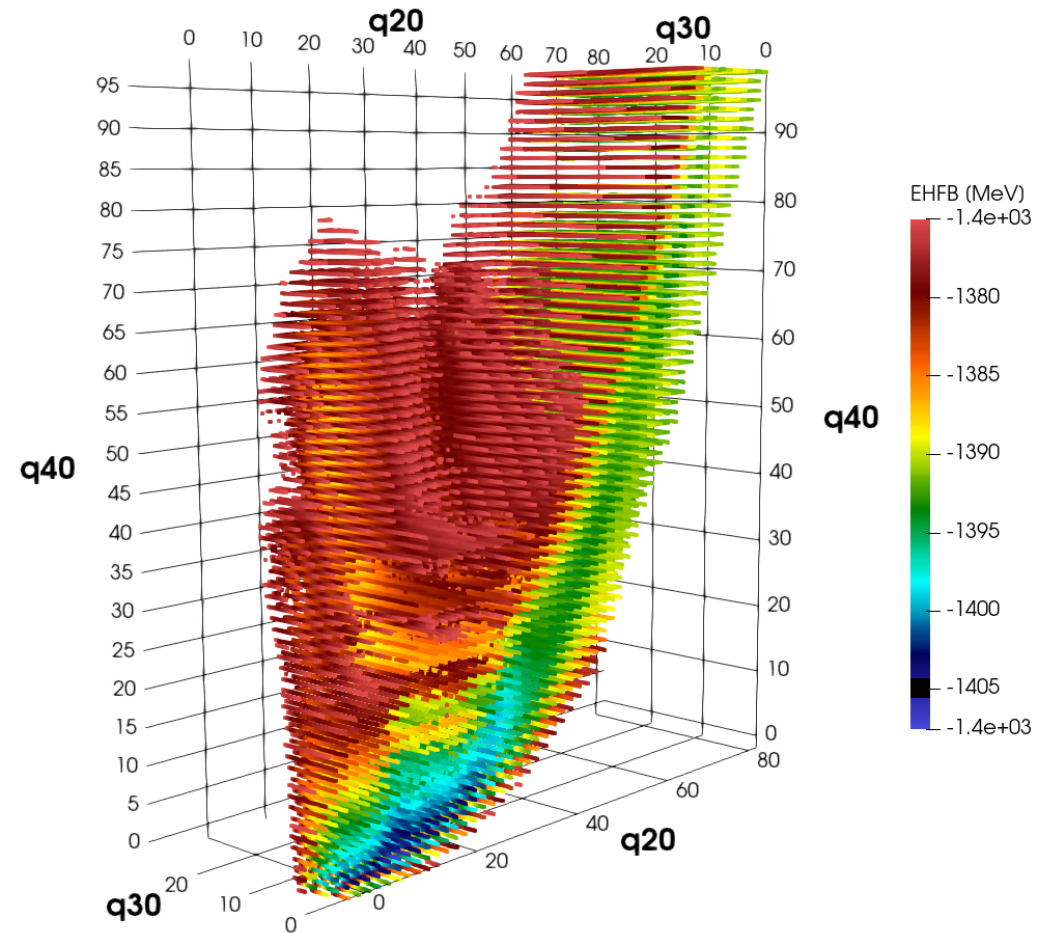
3D PES

^{180}Hg 3D PES

- 300 000 HFB calculations
- 12 500 HFB points in the PES
- 2ct optimized axial HO basis

Run time

- 3 days
 - 16 nodes (128 cpu per node)
- ➔ 150 000 cpu/h



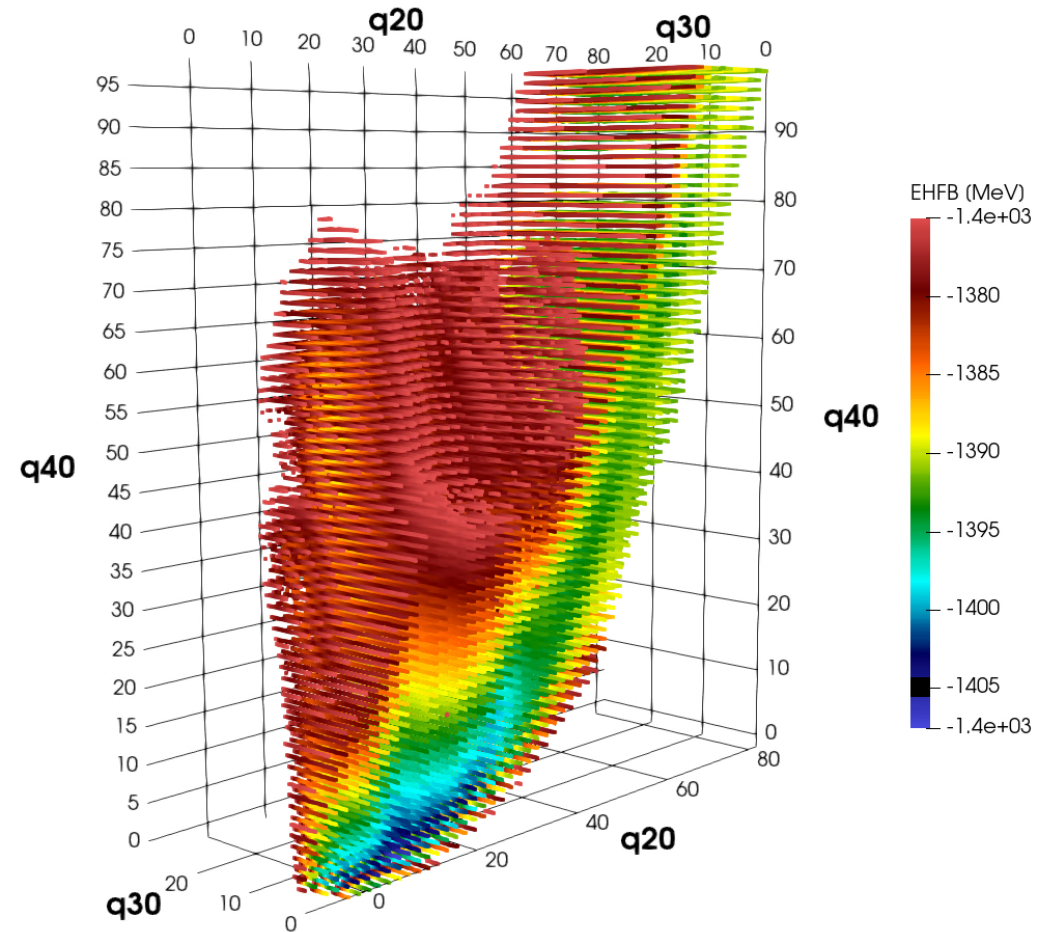
3D PES

^{180}Hg 3D PES

- 300 000 HFB calculations
- 12 500 HFB points in the PES
- 2ct optimized axial HO basis

Run time

- 3 days
 - 16 nodes (128 cpu per node)
- ➔ 150 000 cpu/h



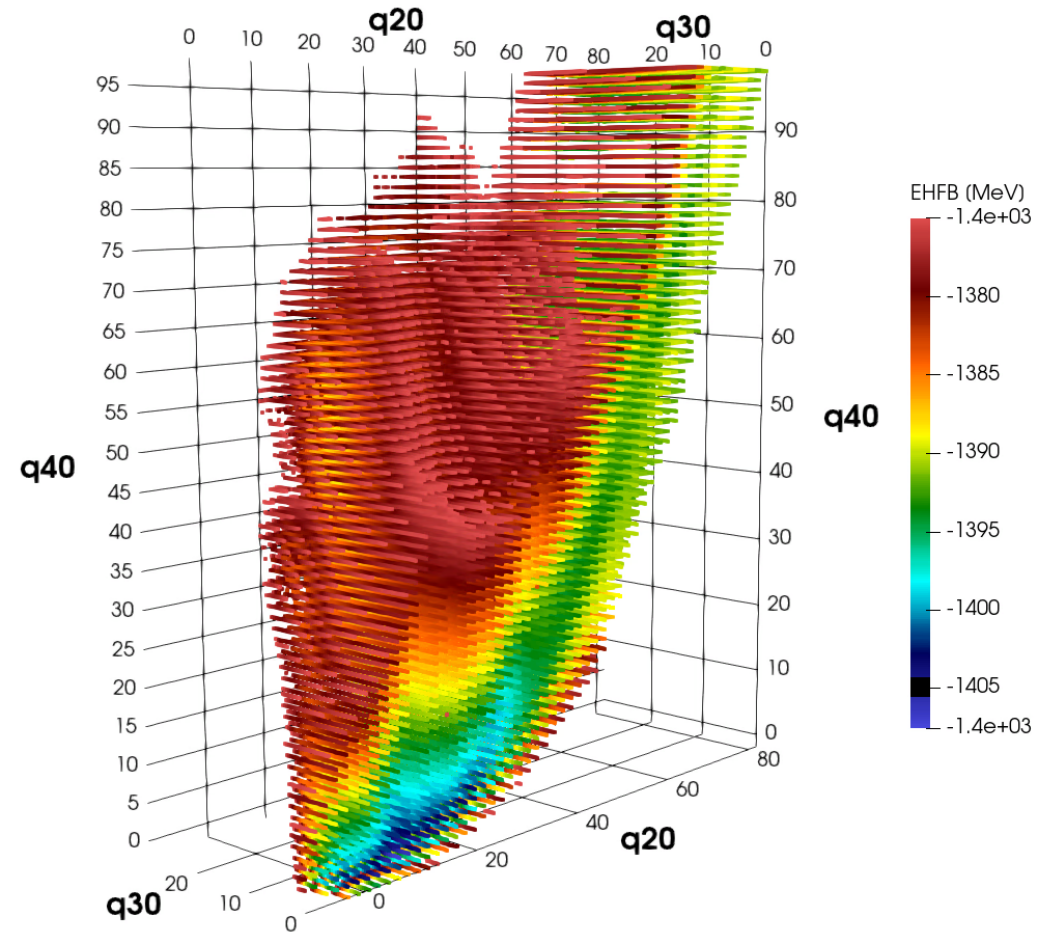
3D PES

^{180}Hg 3D PES

- 300 000 HFB calculations
- 12 500 HFB points in the PES
- 2ct optimized axial HO basis

Run time

- 3 days
 - 16 nodes (128 cpu per node)
- ➔ 150 000 cpu/h



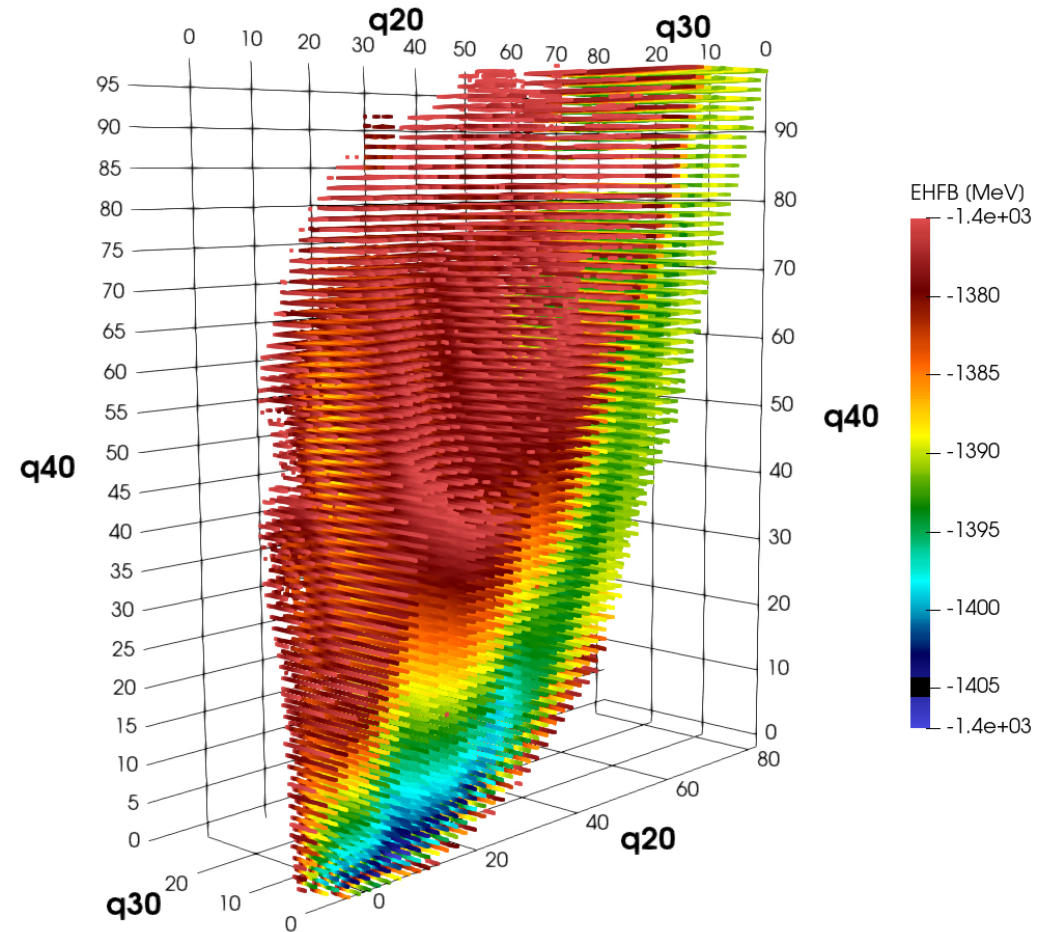
3D PES

^{180}Hg 3D PES

- 300 000 HFB calculations
- 12 500 HFB points in the PES
- 2ct optimized axial HO basis

Run time

- 3 days
 - 16 nodes (128 cpu per node)
- ➔ 150 000 cpu/h



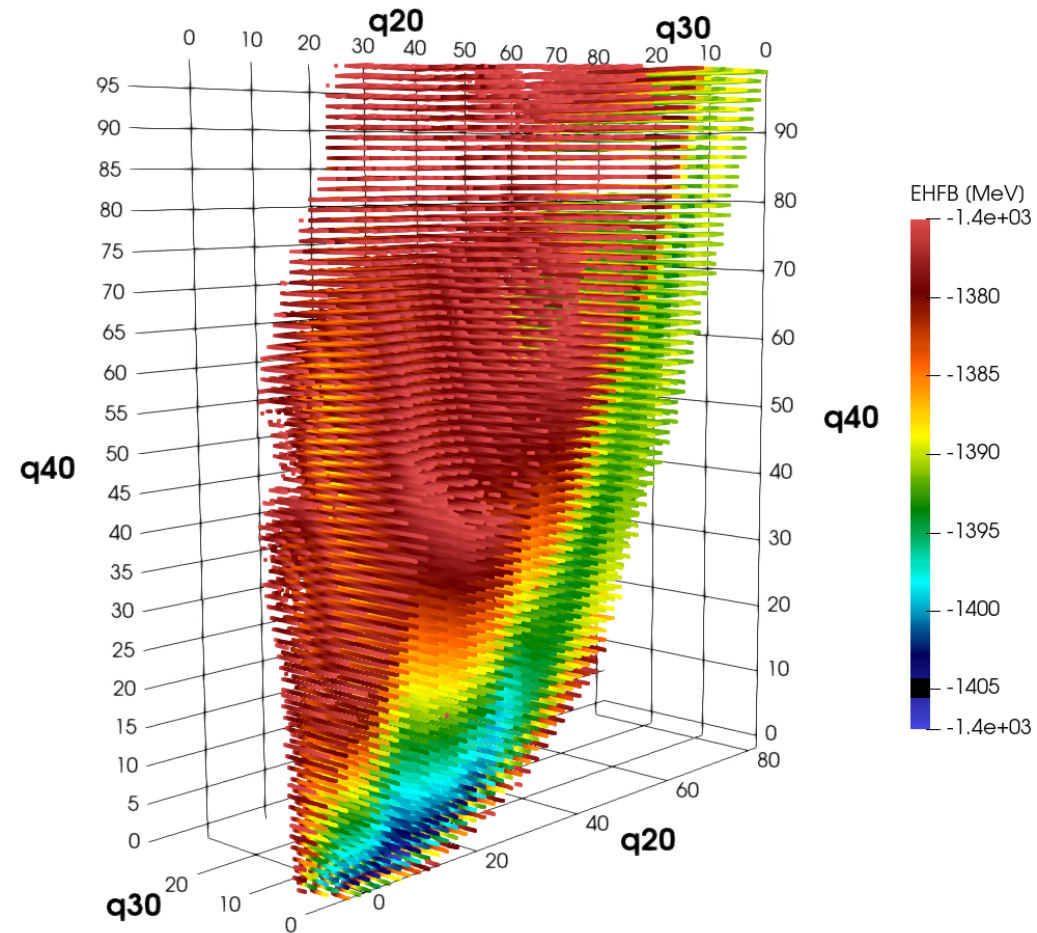
3D PES

^{180}Hg 3D PES

- 300 000 HFB calculations
- 12 500 HFB points in the PES
- 2ct optimized axial HO basis

Run time

- 3 days
 - 16 nodes (128 cpu per node)
- ➔ 150 000 cpu/h



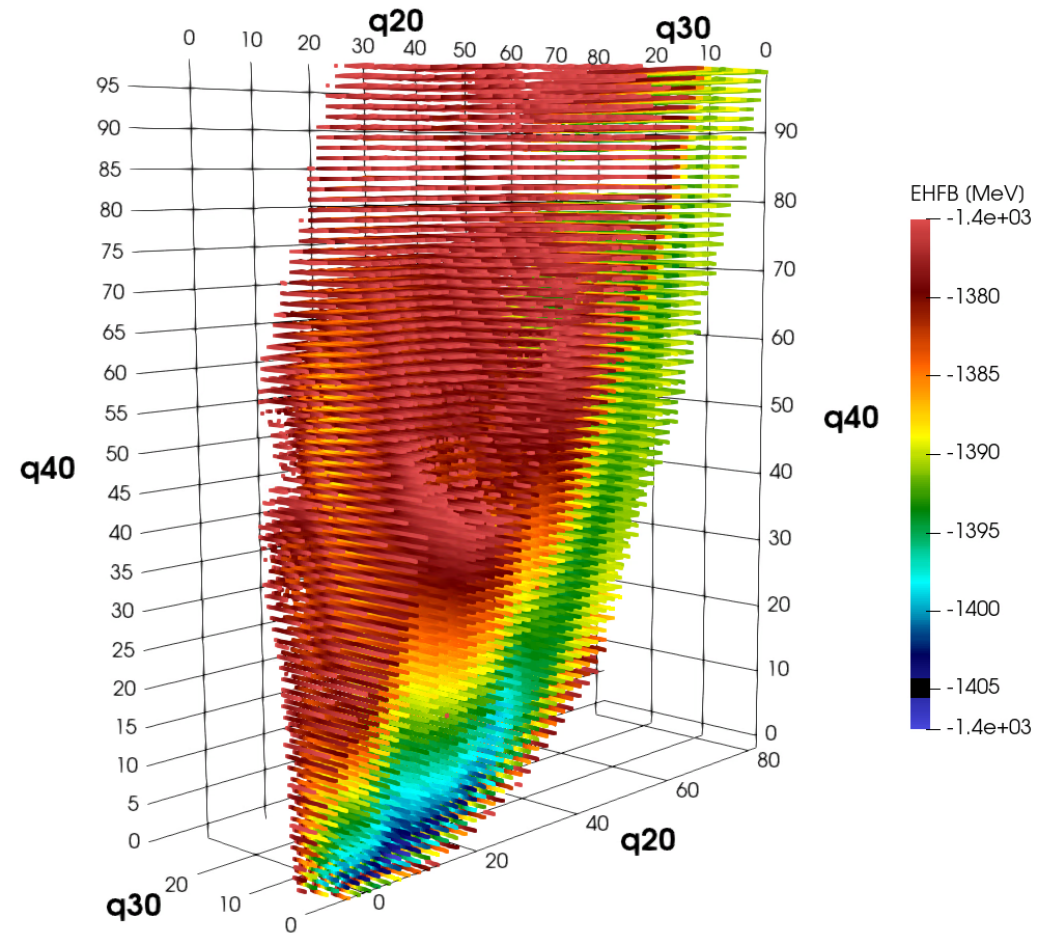
3D PES

^{180}Hg 3D PES

- 300 000 HFB calculations
- 12 500 HFB points in the PES
- 2ct optimized axial HO basis

Run time

- 3 days
 - 16 nodes (128 cpu per node)
- ➔ 150 000 cpu/h



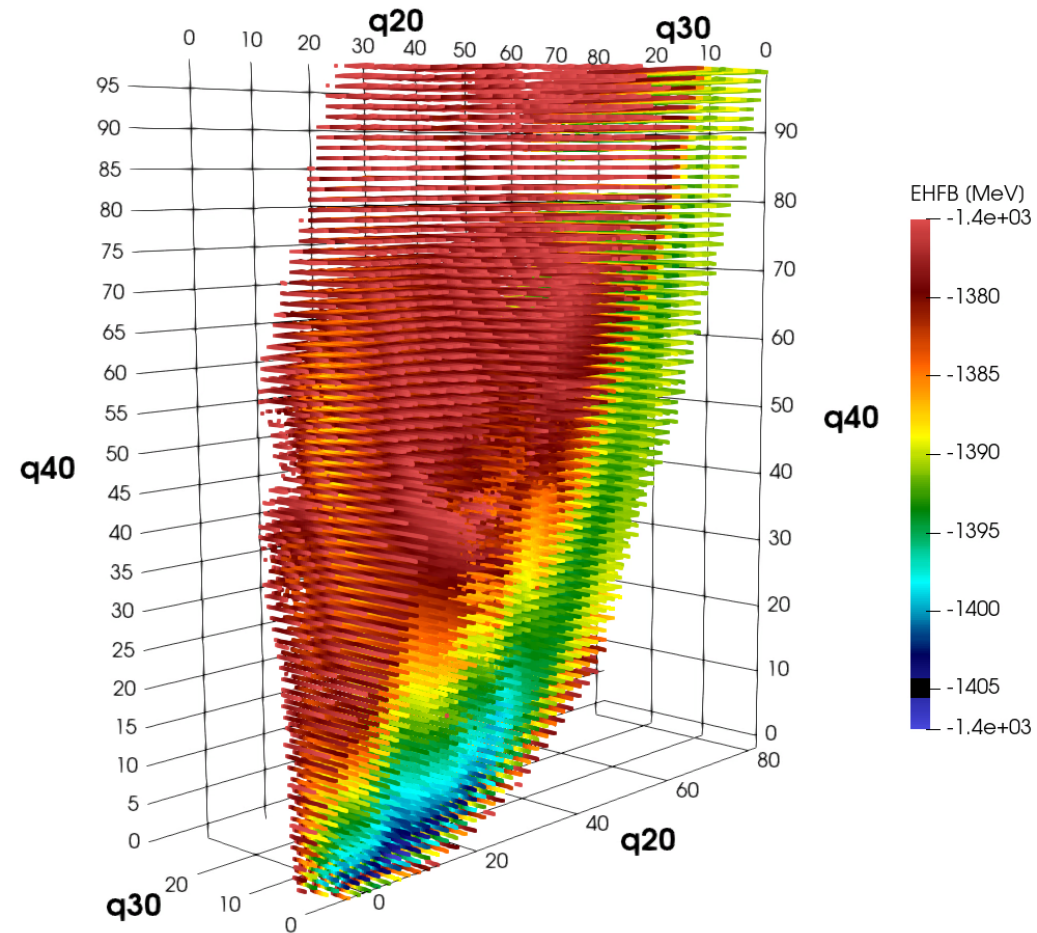
3D PES

^{180}Hg 3D PES

- 300 000 HFB calculations
- 12 500 HFB points in the PES
- 2ct optimized axial HO basis

Run time

- 3 days
 - 16 nodes (128 cpu per node)
- ➔ 150 000 cpu/h



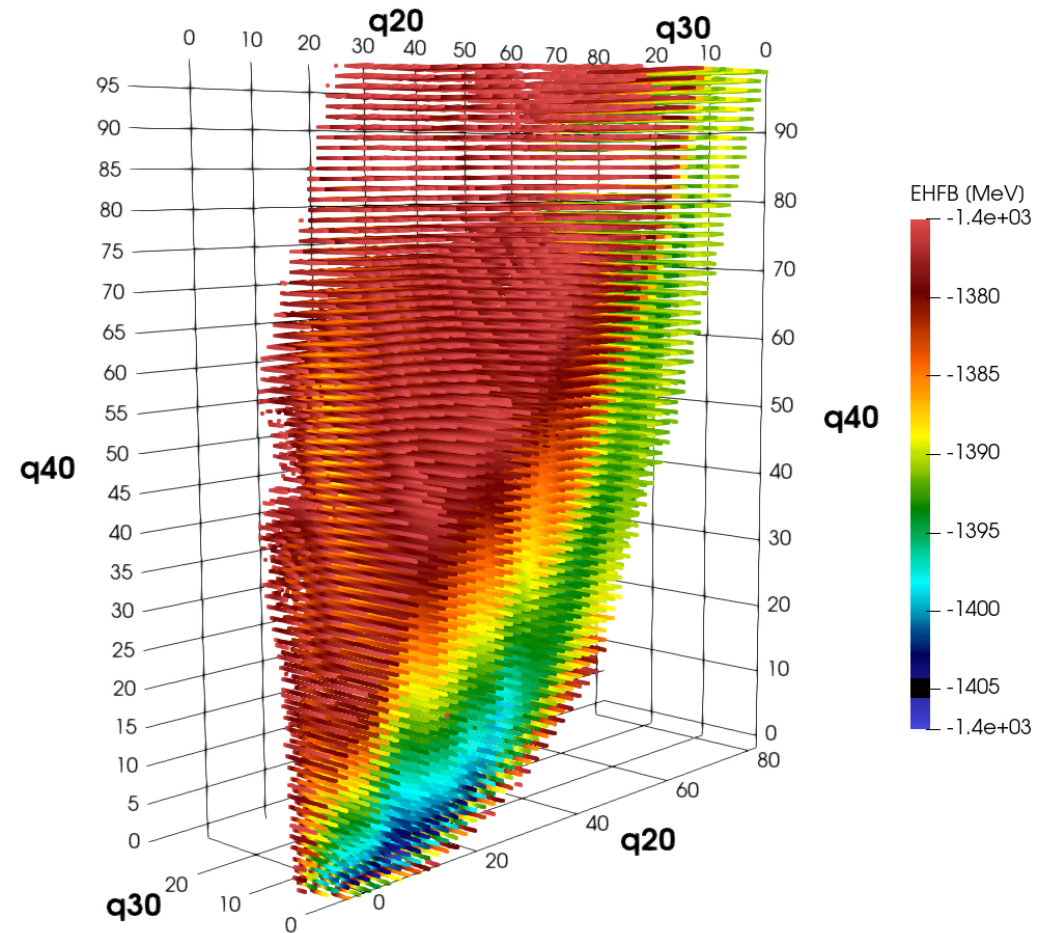
3D PES

^{180}Hg 3D PES

- 300 000 HFB calculations
- 12 500 HFB points in the PES
- 2ct optimized axial HO basis

Run time

- 3 days
 - 16 nodes (128 cpu per node)
- ➔ 150 000 cpu/h



3D PES

^{180}Hg 3D PES

- 300 000 HFB calculations
- 12 500 HFB points in the PES
- 2ct optimized axial HO basis

Run time

- 3 days
 - 16 nodes (128 cpu per node)
- ➔ 150 000 cpu/h

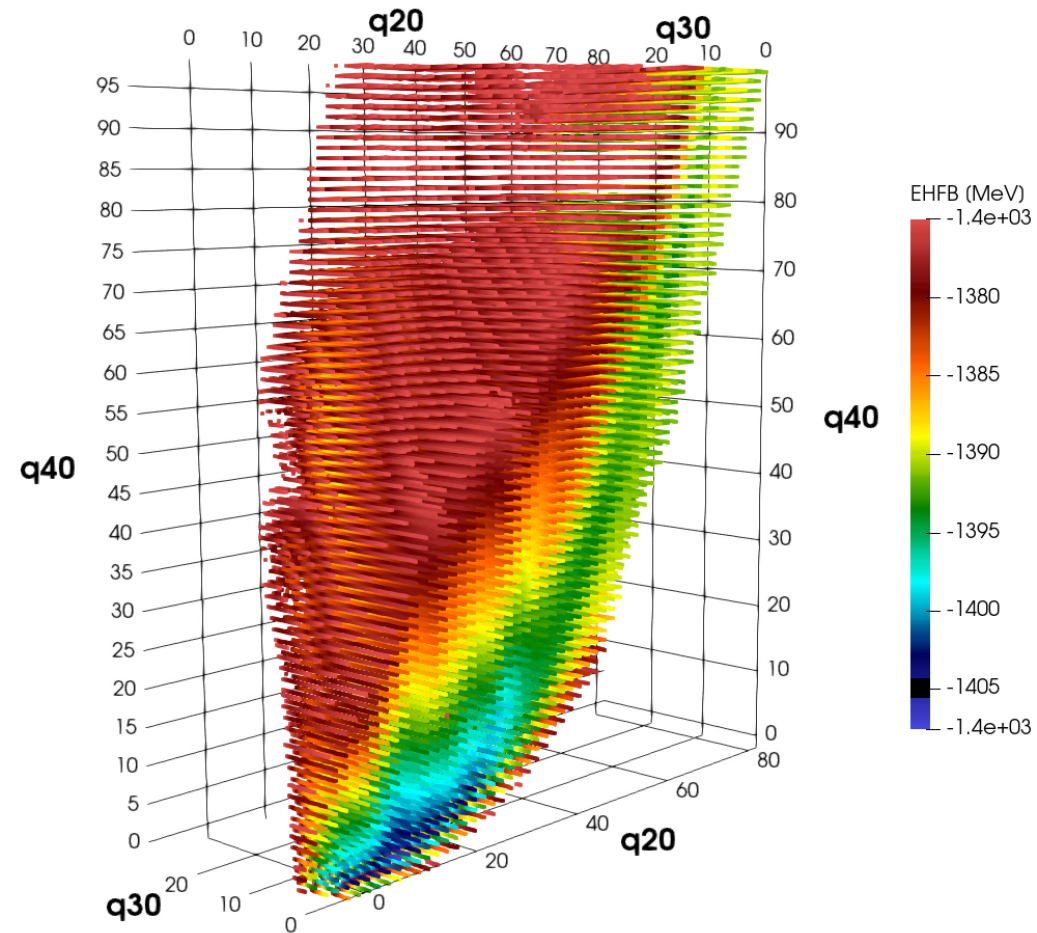


Table of contents

I. HFB Solver - **HFB3**

II. PES Generation - **CHICON**

III. [Odd nuclei](#)



Odd nuclei



Odd nuclei are not time reversal invariant !

1. Blocking + Equal Filing approximation (EFA)

Recently implemented in the HFB3 solver

2. Explicit time-reversal symmetry breaking

- What happens for **fission** (heavy system at high deformation) specifically ?
- **All fields** must be implemented without using time reversal symmetry
 -  All DG fields are implemented
-  It remains to adapt the solver

Conclusion

HFB3

- Solver is efficient and stable
- Accepted for publication in EPJA



PES generation

- Systematic 1000 PES production
 - **Soon available:** Fission data on a CEA website
- 3D PES production is in development



Odd nuclei

- All DG fields with time-reversal symmetry breaking are implemented
- **Soon available:** HFB solver breaking time-reversal symmetry
- **PhD Goal:** Results comparing EFA to symmetry breaking for odd actinide fission

