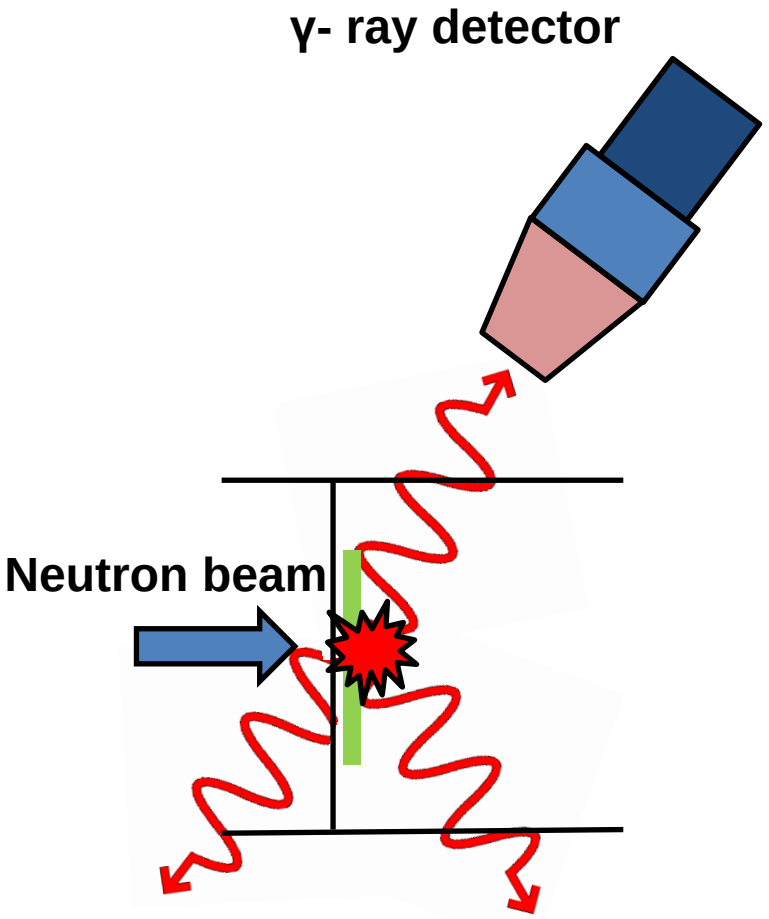
Machine learning applied to the modelling of nuclear deexcitation cascades

E. Mendoza, V. Alcayne, D. Cano-Ott
CIEMAT

Measurement of neutron capture cross sections at n_TOF (CERN)



To measure neutron capture cross sections we need:

- Neutron beam
- Sample
- Detection system

$$Y_{x,exp}(E_n) = \frac{C(E_n) - B(E_n)}{\varepsilon_x(E_n) \cdot \phi_n(E_n)}$$

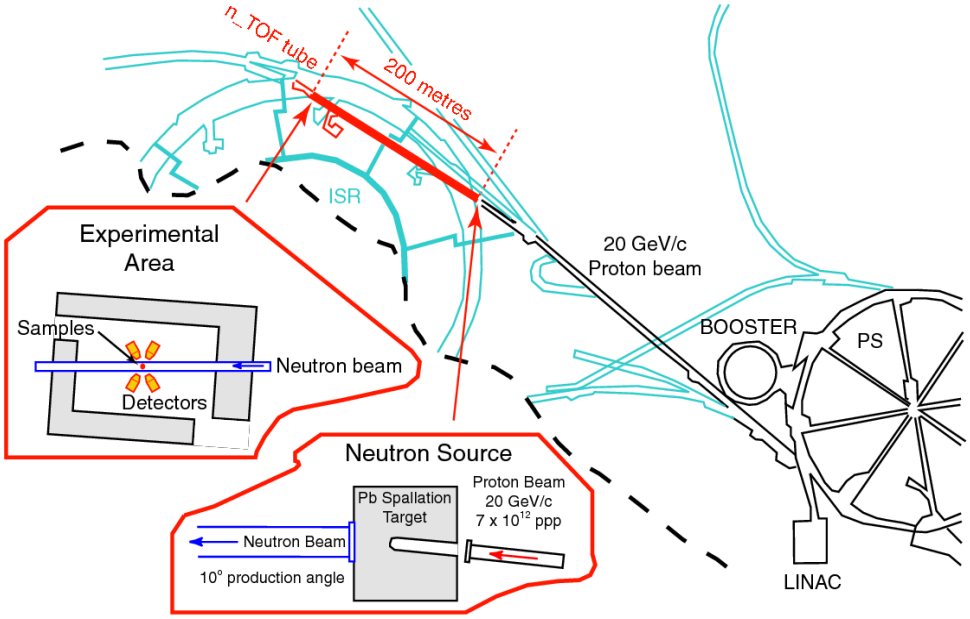
$$Y_{x,th}(E_n) \cong \frac{\sigma_x(E_n)}{\sigma_{tot}(E_n)}\left(1 - e^{-n\sigma_{tot}(E_n)}\right)$$

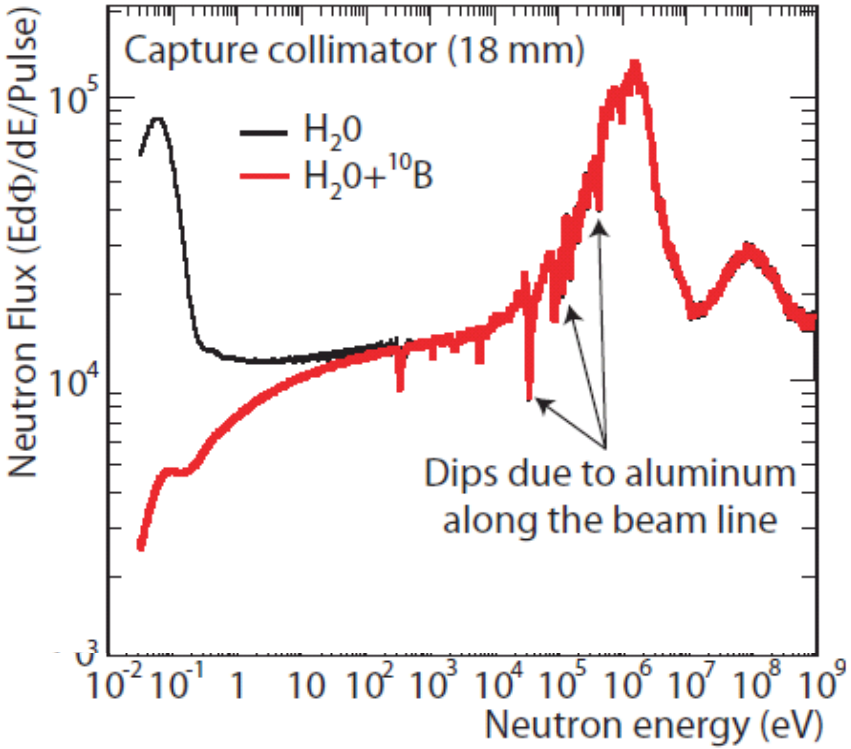
x → capture, fission …

$$\varepsilon_\gamma = \frac{\text{nº detected events}}{\text{nº captured events}}$$

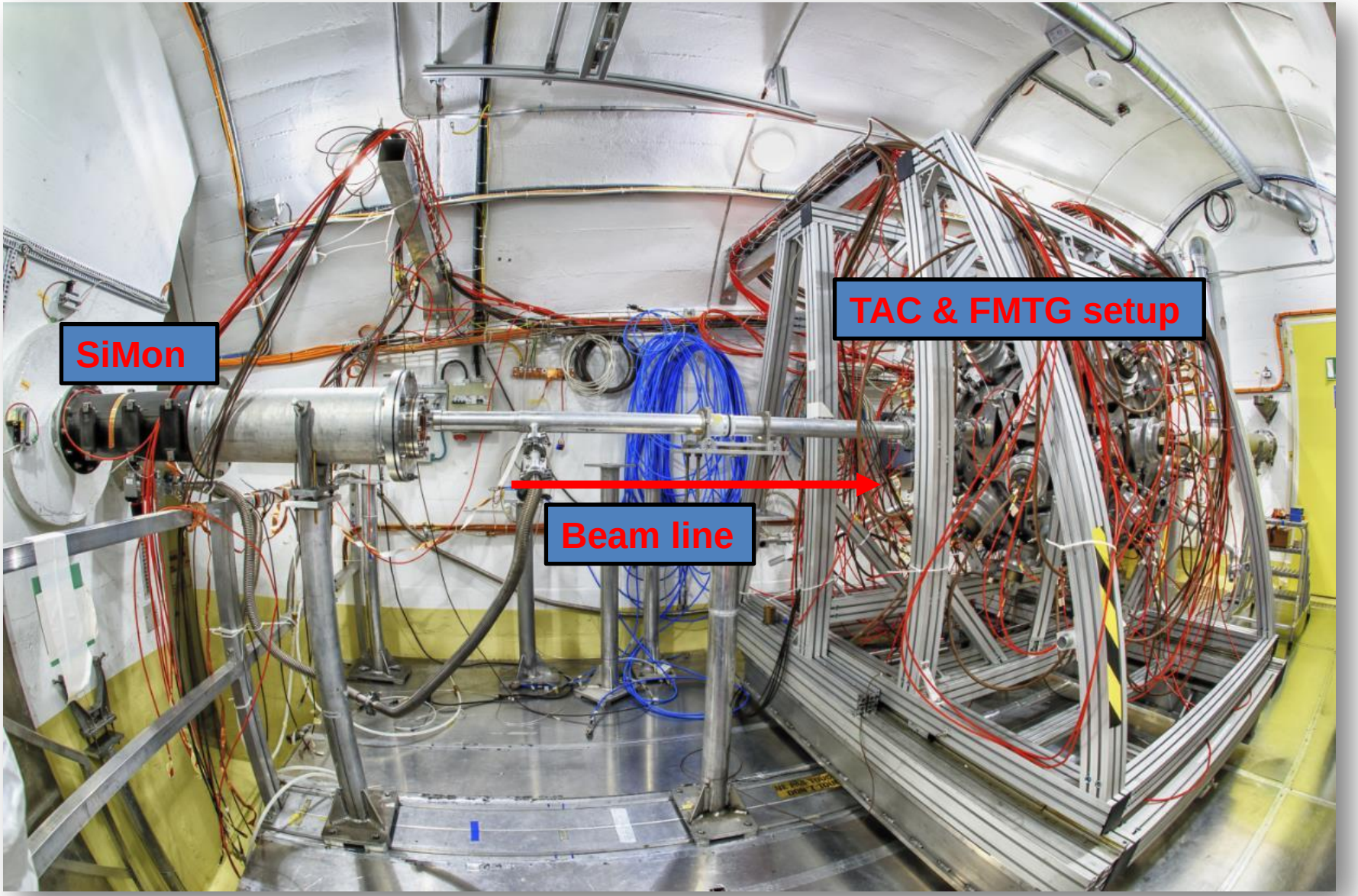
Measurement of neutron capture cross sections at n_TOF (CERN)

The n_TOF facility is a high instantaneous intensity spallation neutron source driven by the CERN PS synchrotron (20 GeV/c with up to $8x10^{17}$ ppp)





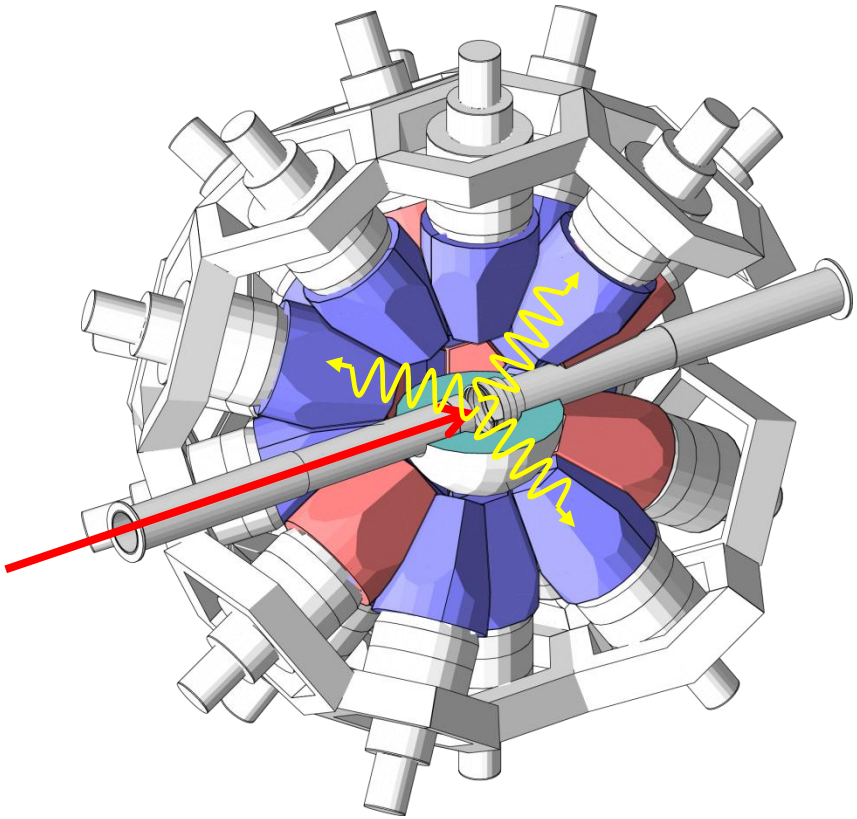
n_TOF EAR1



TAC & FTMG experimental setup

10 x $^{235}U_3O_8$ samples prepared by JRC Geel

Measurement of neutron capture cross sections at n_TOF (CERN)



To measure neutron capture cross sections we need:

- Neutron beam
- Sample
- Detection system

$$Y_{x,exp}(E_n) = \frac{C(E_n) - B(E_n)}{\varepsilon_x(E_n) \cdot \phi_n(E_n)}$$

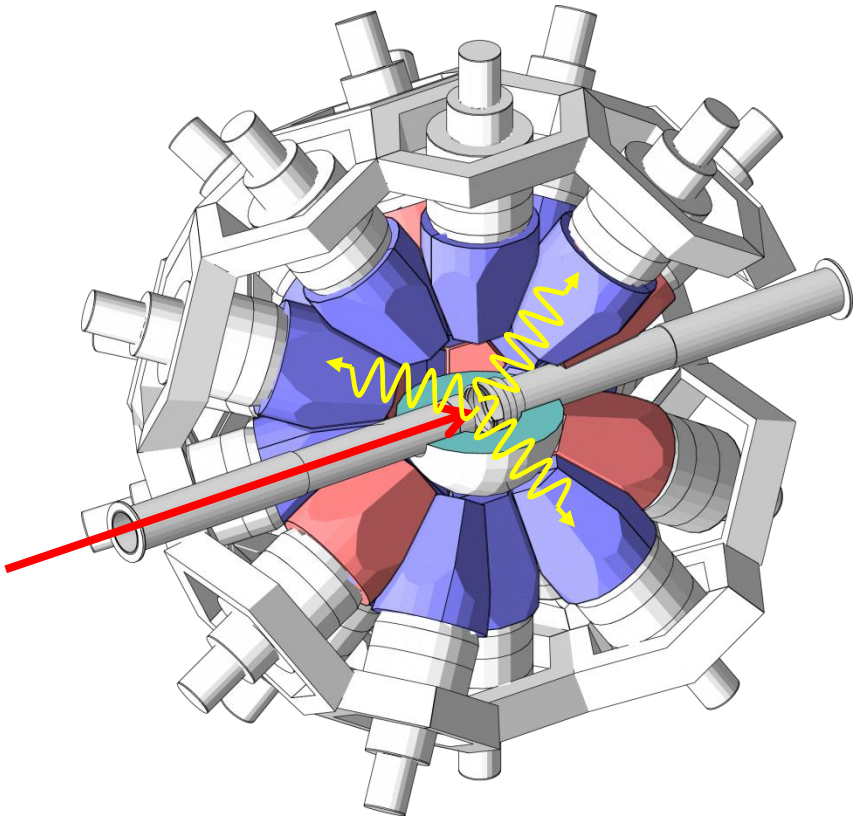
$$Y_{x,th}(E_n) \cong \frac{\sigma_x(E_n)}{\sigma_{tot}(E_n)}\left(1 - e^{-n\sigma_{tot}(E_n)}\right)$$

x → capture, fission …

Total Absorption Calorimeter: 40 BaF_2 crystals detecting the γ-ray cascades in coincidence.

$$\varepsilon_\gamma = \frac{\text{nº detected events}}{\text{nº captured events}}$$

Measurement of neutron capture cross sections at n_TOF (CERN)



Total Absorption Calorimeter: 40 BaF_2 crystals detecting the γ-ray cascades in coincidence.

We obtain the detection efficiency from Monte Carlo simulations:

1- Generate the γ-ray cascades emitted after neutron capture.

2- Transport the γ-rays through the TAC geometry (Geant4 MC toolkit).

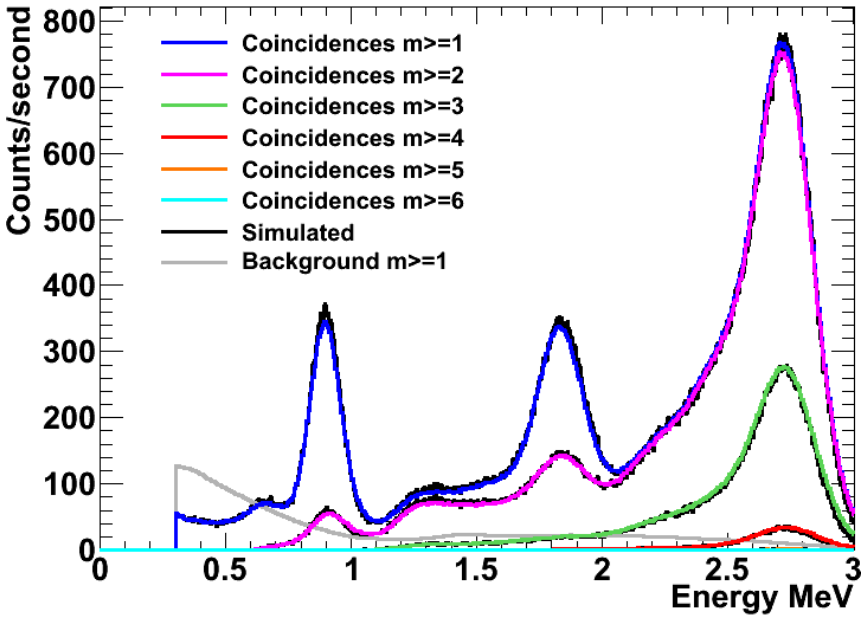
3- Reconstruct the detected events in the same way as it is done in the experiment.

→ Compare the results of the simulations with the experimental results (E_{dep} spectra).

$$\varepsilon_\gamma = \frac{nº\ detect\ \ event}{nº\ capt^{u}\ \ event}$$

Measurement of neutron capture cross sections at n_TOF (CERN)

^{88}Y calibration source



validation

We obtain the detection efficiency from Monte Carlo simulations:

1- Generate the γ-ray cascades emitted after neutron capture.

2- Transport the γ-rays through the TAC geometry (Geant4 MC toolkit).

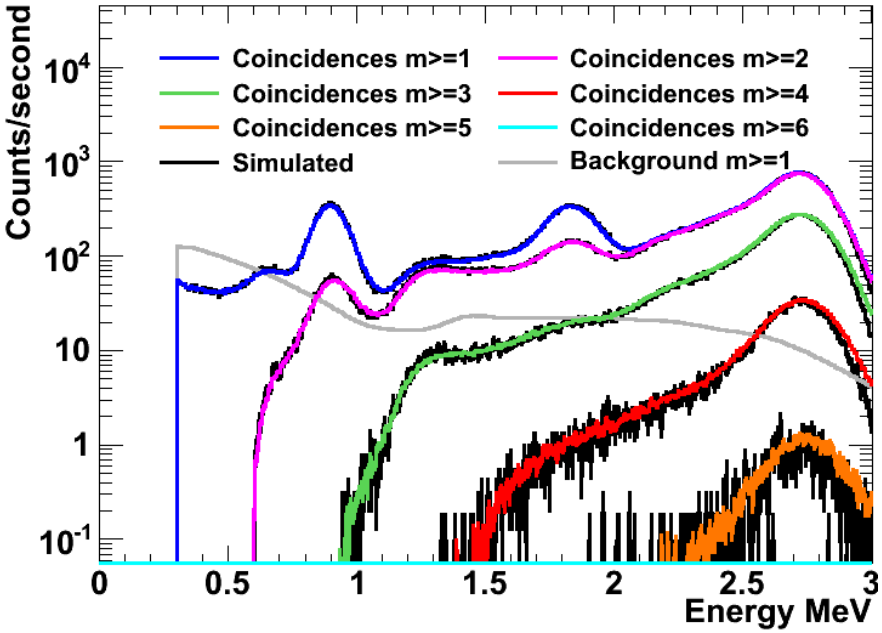
3- Reconstruct the detected events in the same way as it is done in the experiment.

→ Compare the results of the simulations with the experimental results (E_{dep} spectra).

$$\varepsilon_\gamma = \frac{nº\ detect\quad event}{nº\ capt^{u}\quad event}$$

Measurement of neutron capture cross sections at n_TOF (CERN)

^{88}Y calibration source



validation

We obtain the detection efficiency from Monte Carlo simulations:

1- Generate the γ-ray cascades emitted after neutron capture.

2- Transport the γ-rays through the TAC geometry (Geant4 MC toolkit).

3- Reconstruct the detected events in the same way as it is done in the experiment.

→ Compare the results of the simulations with the experimental results (E_{dep} spectra).

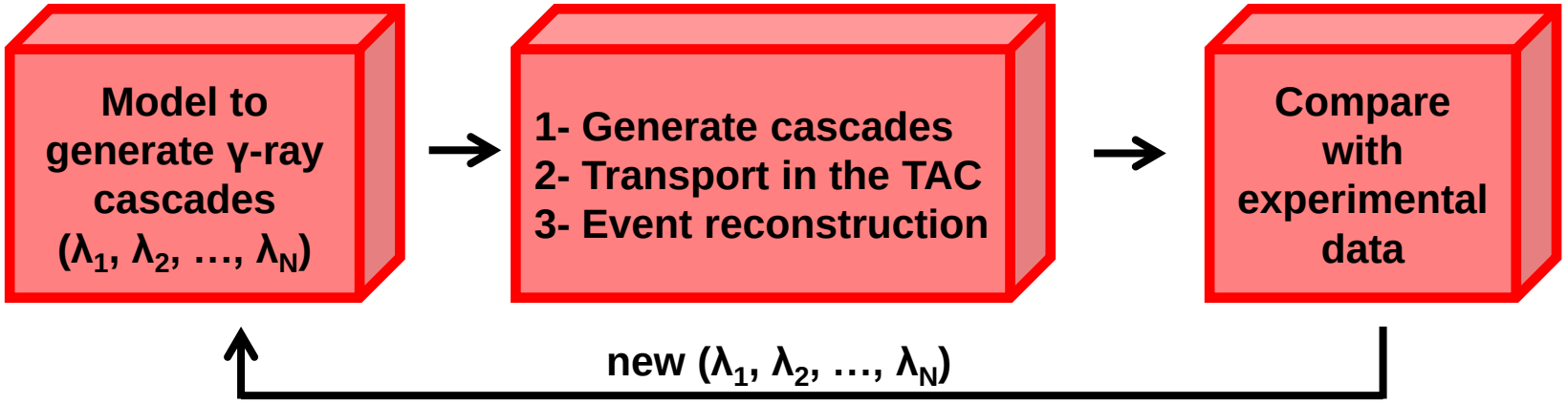
$$\varepsilon_\gamma = \frac{nº\ detect\quad event}{nº\ capt^{u}\quad event}$$

Generation of the γ-ray cascades

It is not straightforward how to generate reliable γ-ray cascades emitted after neutron capture.

How we proceeded in the past:

- Use a model to generate the γ-ray cascades.
- The model depends on some parameters: $\lambda_1, \lambda_2, \ldots, \lambda_N$.
- Adjust the parameters “by hand” to reproduce the experimental spectra.

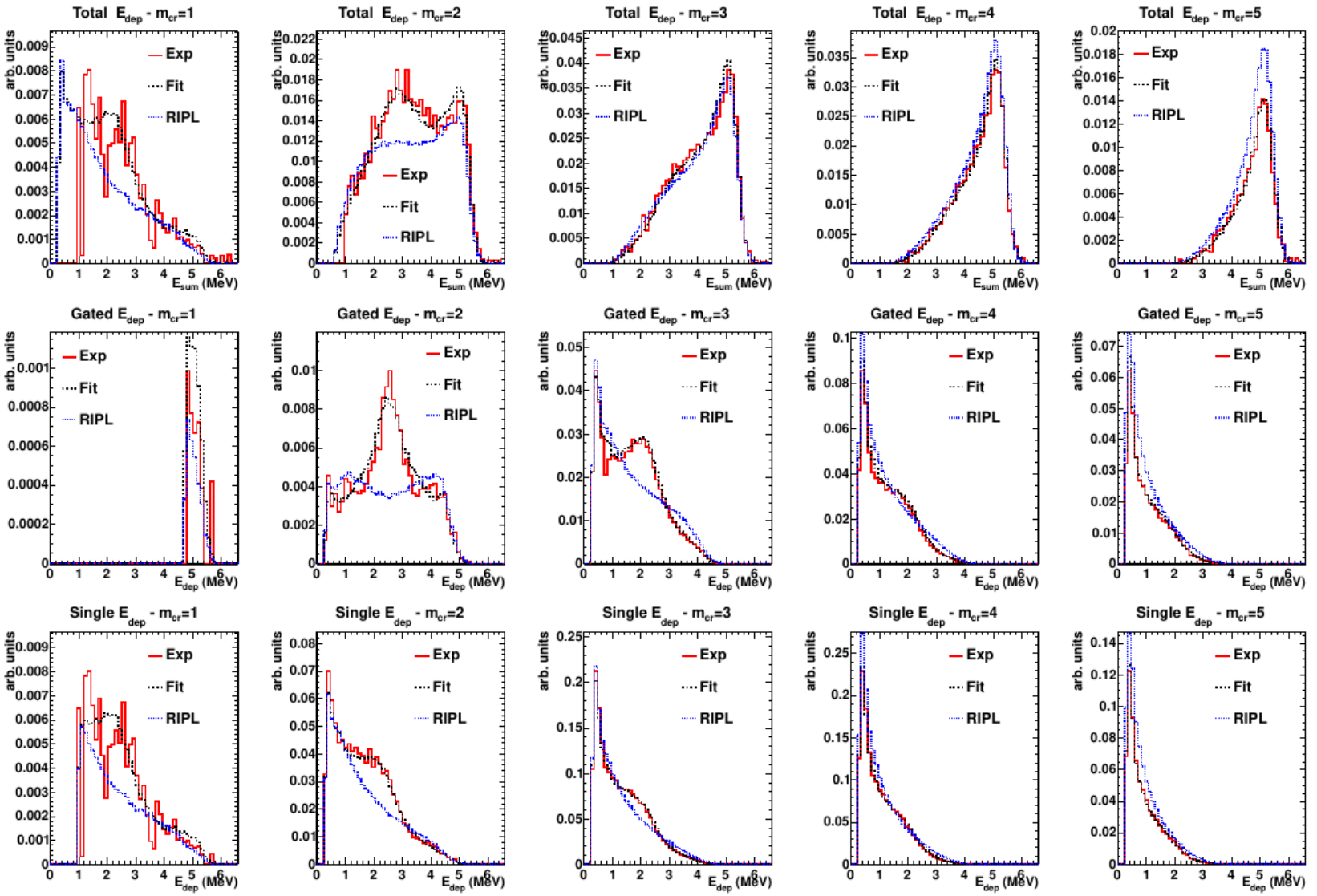


What we are doing/trying to do:

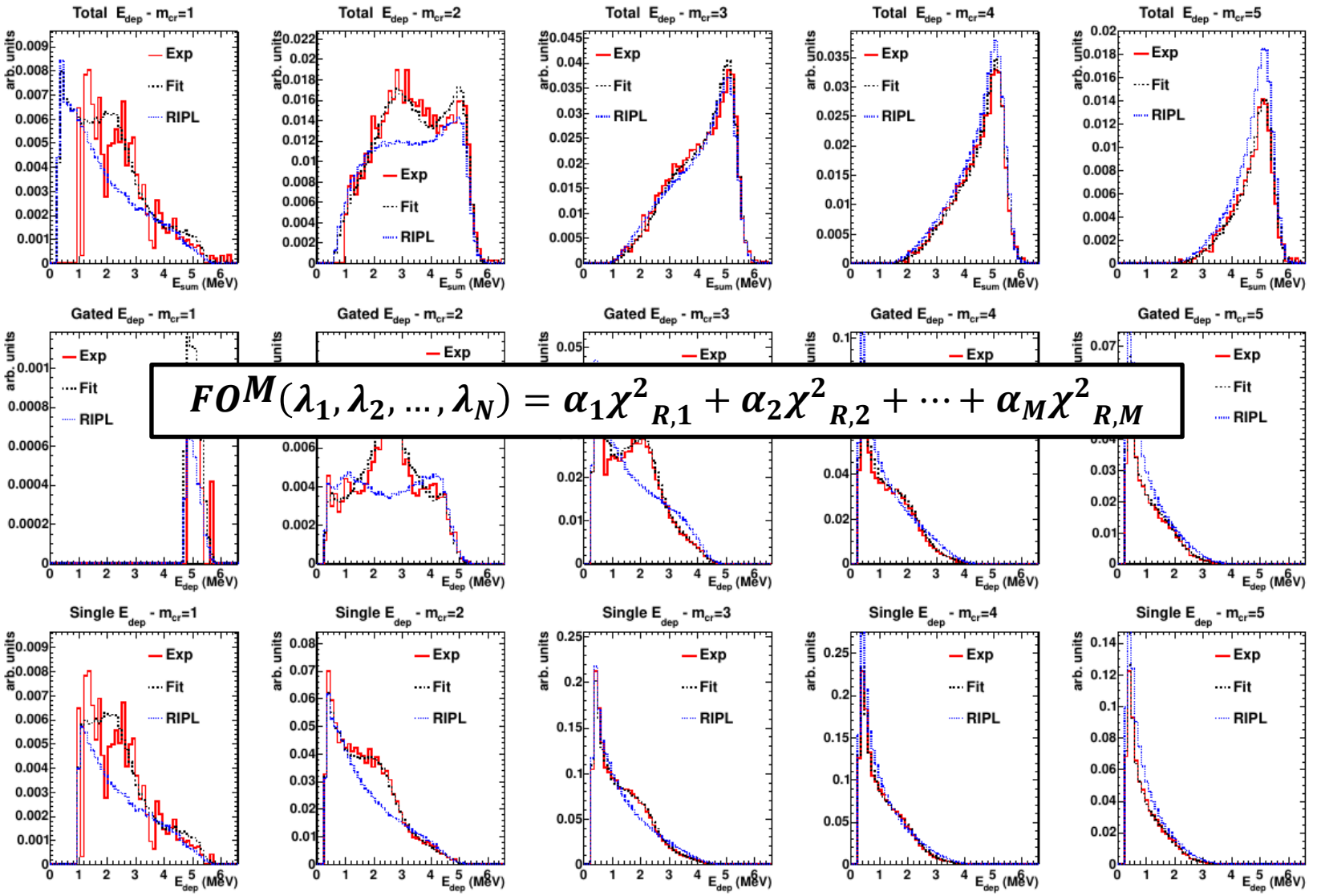
- Make a **fit** of the $\lambda_1, \lambda_2, \ldots, \lambda_N$ parameters to the experimental data.

To make a fit we try to minimize a figure of merit: FOM=FOM($\lambda_1, \lambda_2, \ldots, \lambda_N$)

Generation of the γ-ray cascades



Generation of the γ-ray cascades



$$FO^{M}(\lambda_1, \lambda_2, \ldots, \lambda_N) = \alpha_1 \chi^2_{R,1} + \alpha_2 \chi^2_{R,2} + \cdots + \alpha_M \chi^2_{R,M}$$

Generation of the γ-ray cascades

Goal: find the minimum of FOM(λ_1, λ_2, …, λ_N)

Limitations of the fitting process:

- It takes some time (~10-20 min.) to evaluate the FOM function in one (λ_1, λ_2, …, λ_N) point (generate cascades + MC transport + event reconstruction).
- The evaluation of the FOM function is not performed with 100% precision → uncertainty due to counting statistics in the MC → no derivatives.

Differential evolution algorithm:

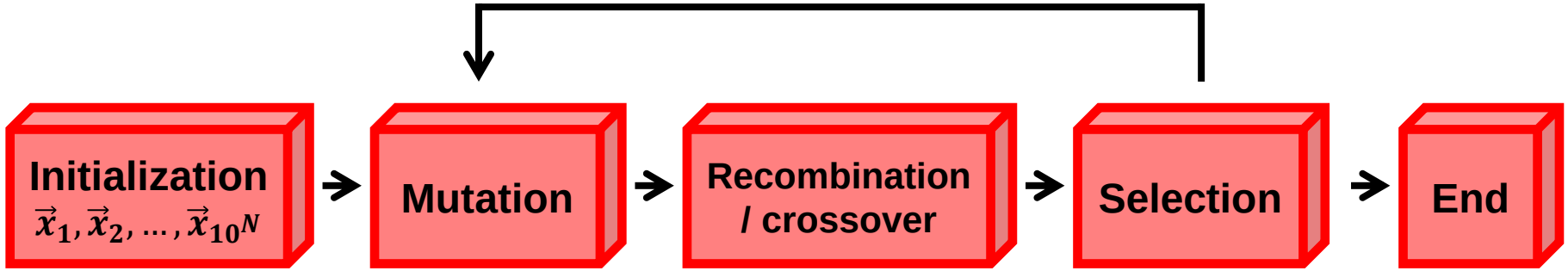
- no derivatives ($f(\lambda_1, \lambda_2, \ldots, \lambda_N)$ doesn't need to be even continuous → *good* for noisy functions)
- robust
- very easy to implement
- easy to use in parallel computing

References:

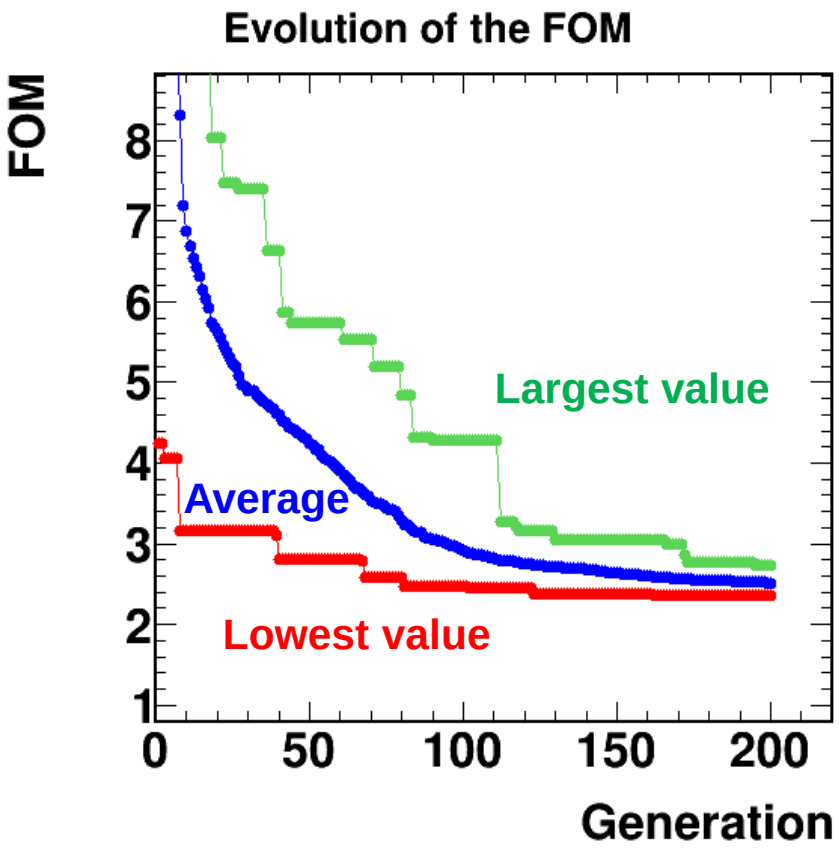
- R. Storn, K. Price, Journal of Global Optimization 11, 341 (1997)
- S. Das, P.N. Suganthan, IEEE Trans. on Evolutionary Computation 15, 4 (2011)
- S. Das, S.S. Mullick, P. Suganthan, Swarm and Evol. Computation 27, 1 (2016)

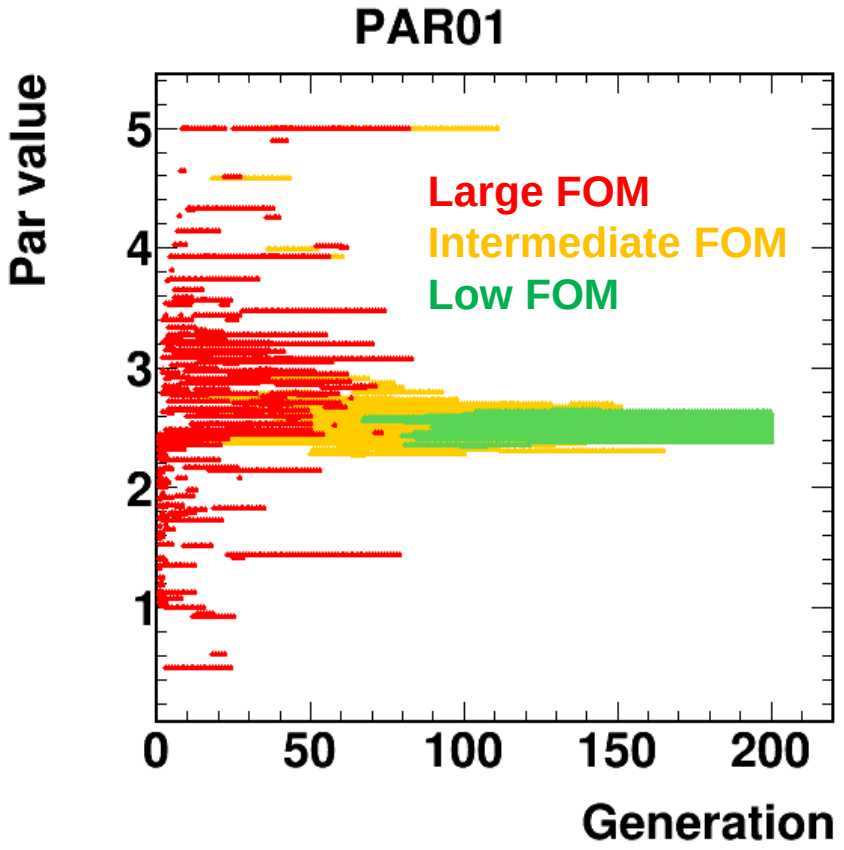
The differential evolution algorithm

- There are some candidate solution vectors (~10xN), called *agents*: $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{10N}$, with $\vec{x}_1 \in \mathbb{R}^N$, $(\lambda_1, \lambda_2, \ldots, \lambda_N)$.
- At each iteration, a new candidate vectors for the new generation of agents is constructed using perturbations or linear combinations of the previous generation agents (**mutation**) + **recombination** of the old and new agents.
- The function is evaluated with the new candidate agents. Those candidates which improve the solution replace the ones of the previous generation → new generation.



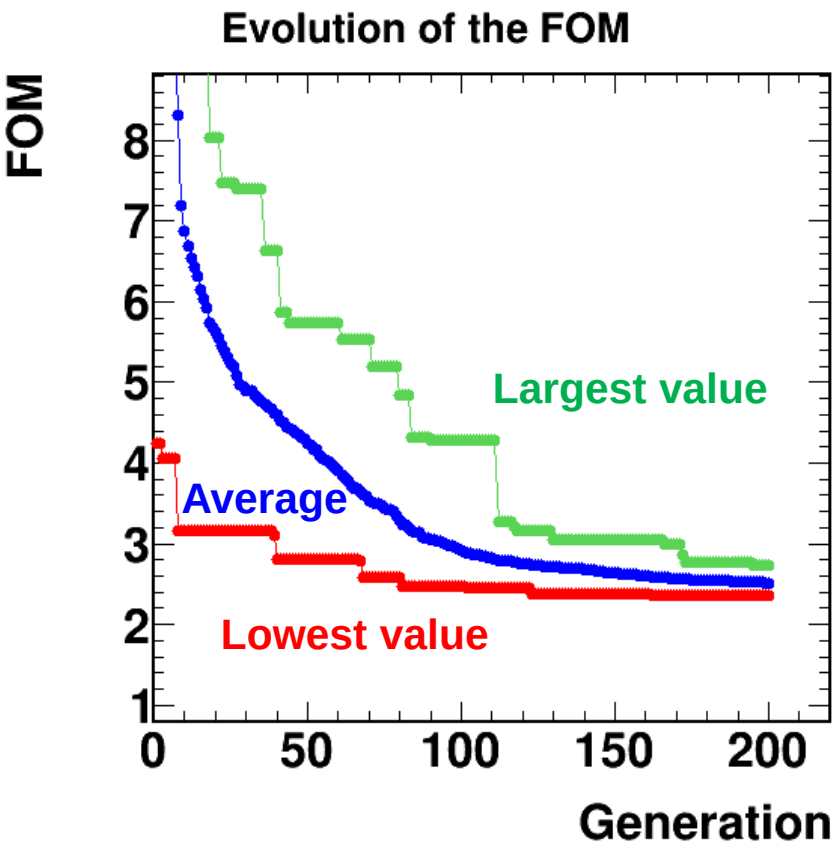
Convergence of the method

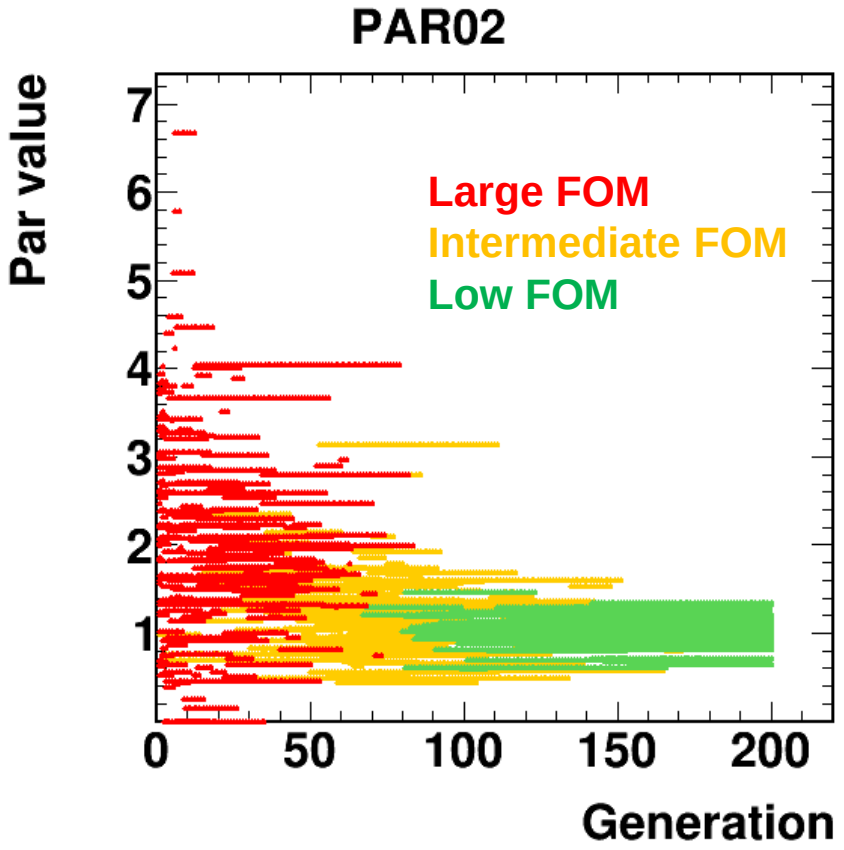




Left: Evolution of the FOM with the generation: largest and lowest FOM value, and average over all agents.
Right: evolution of the value of parameter 1.

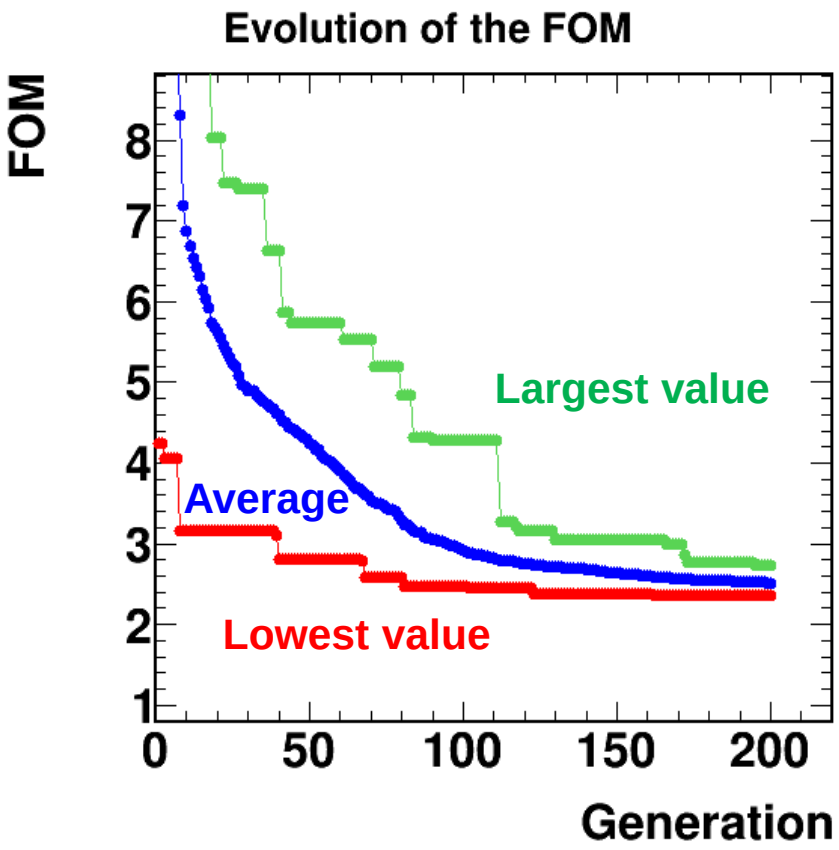
Convergence of the method

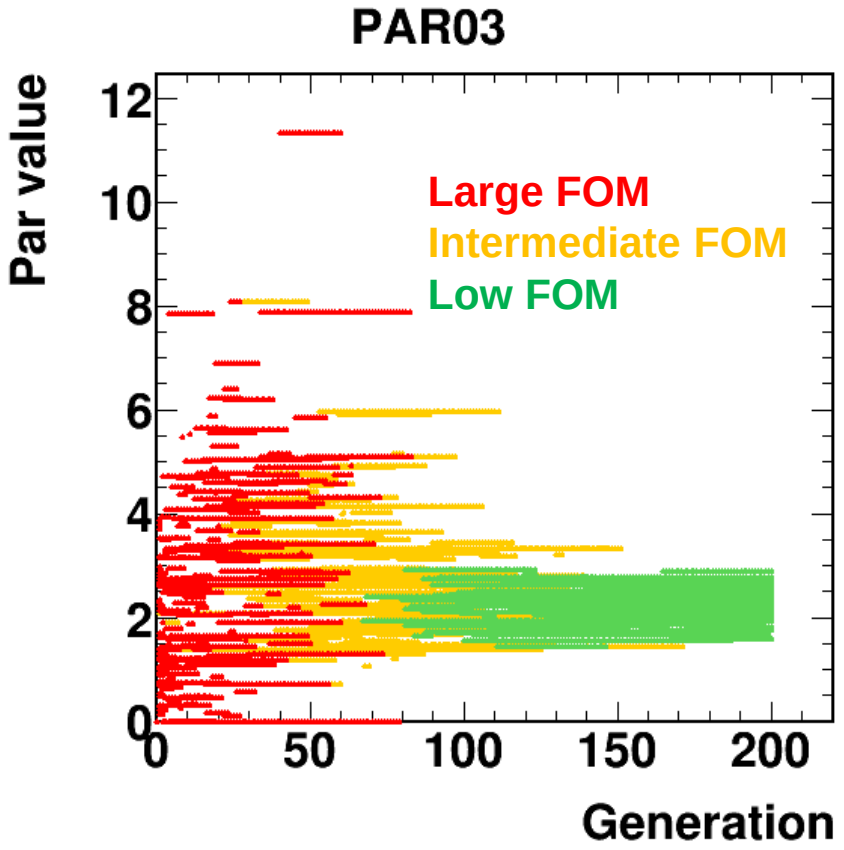




Left: Evolution of the FOM with the generation: largest and lowest FOM value, and average over all agents.
Right: evolution of the value of parameter 2.

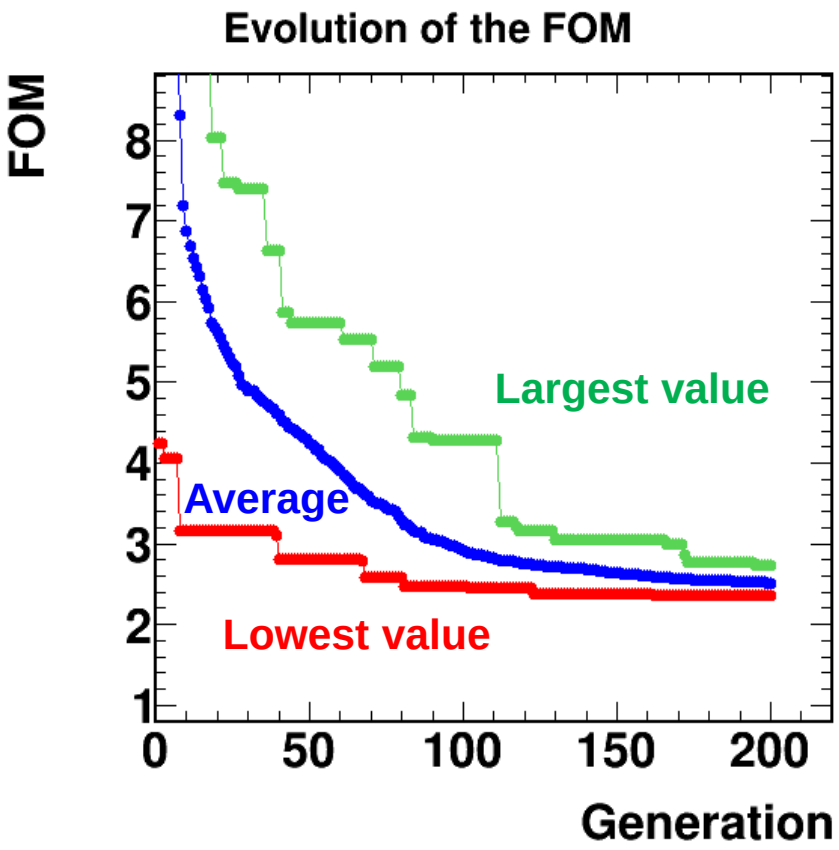
Convergence of the method

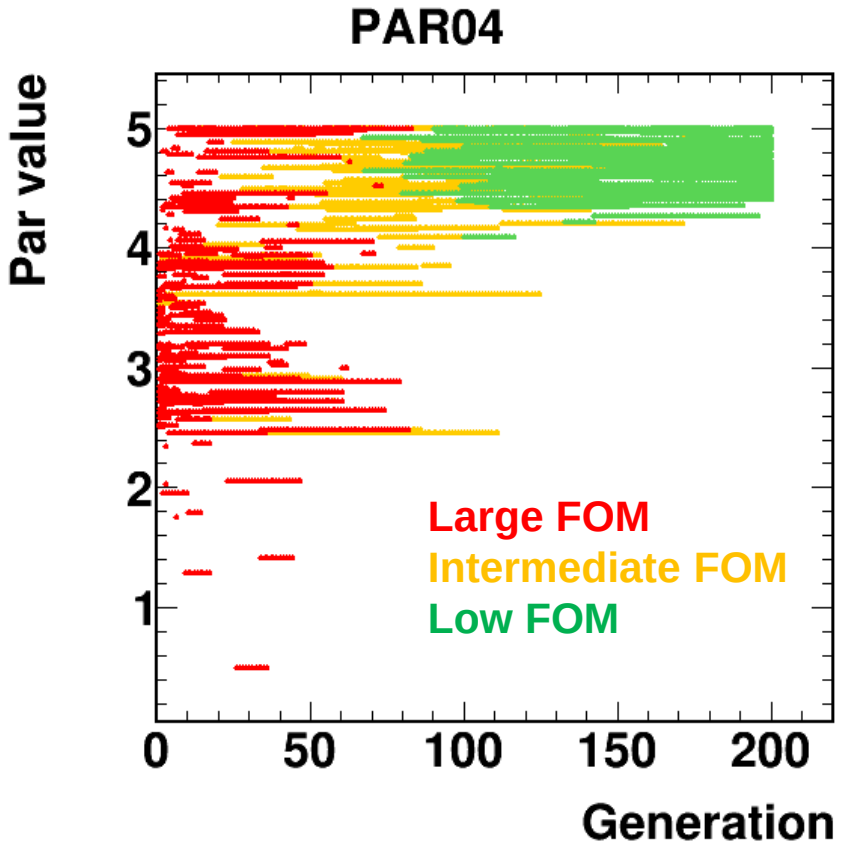




Left: Evolution of the FOM with the generation: largest and lowest FOM value, and average over all agents.
Right: evolution of the value of parameter 3.

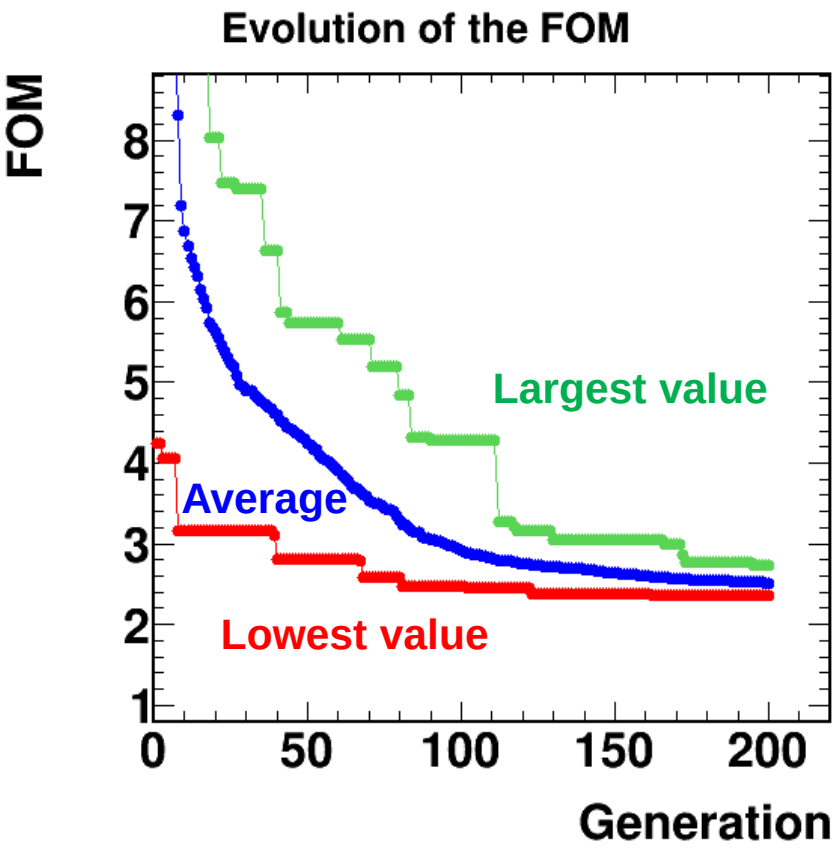
Convergence of the method

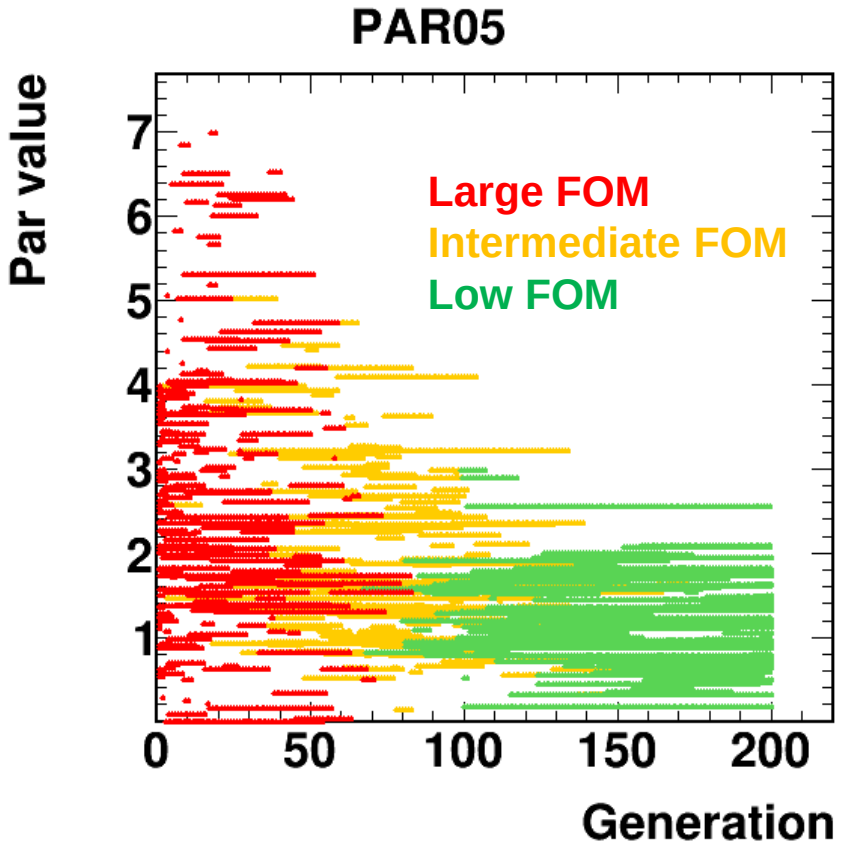




Left: Evolution of the FOM with the generation: largest and lowest FOM value, and average over all agents.
Right: evolution of the value of parameter 4.

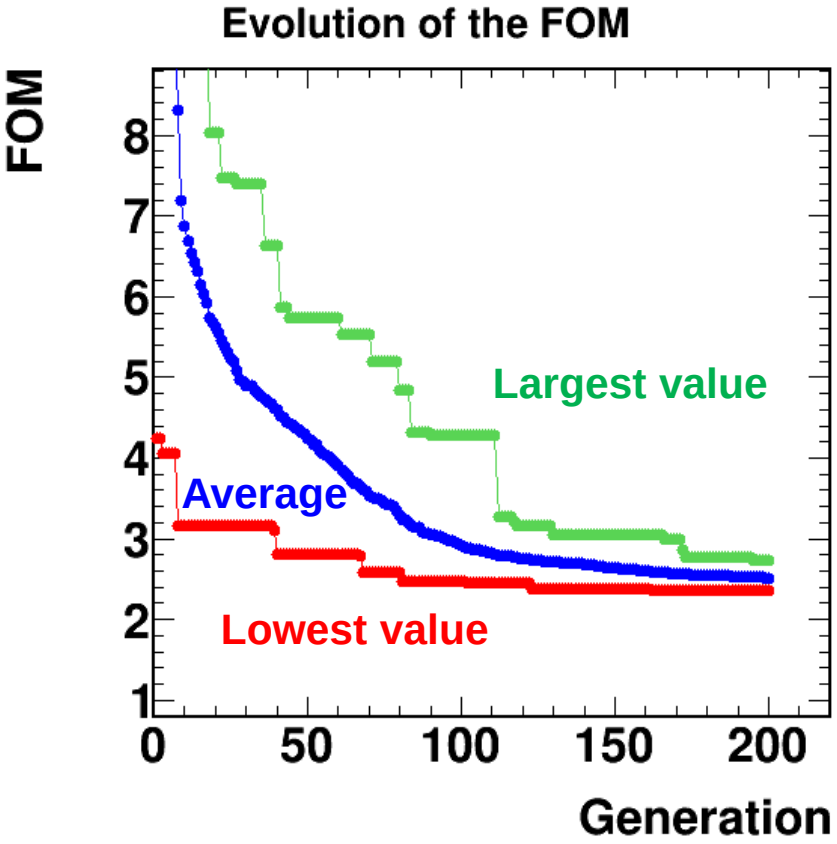
Convergence of the method

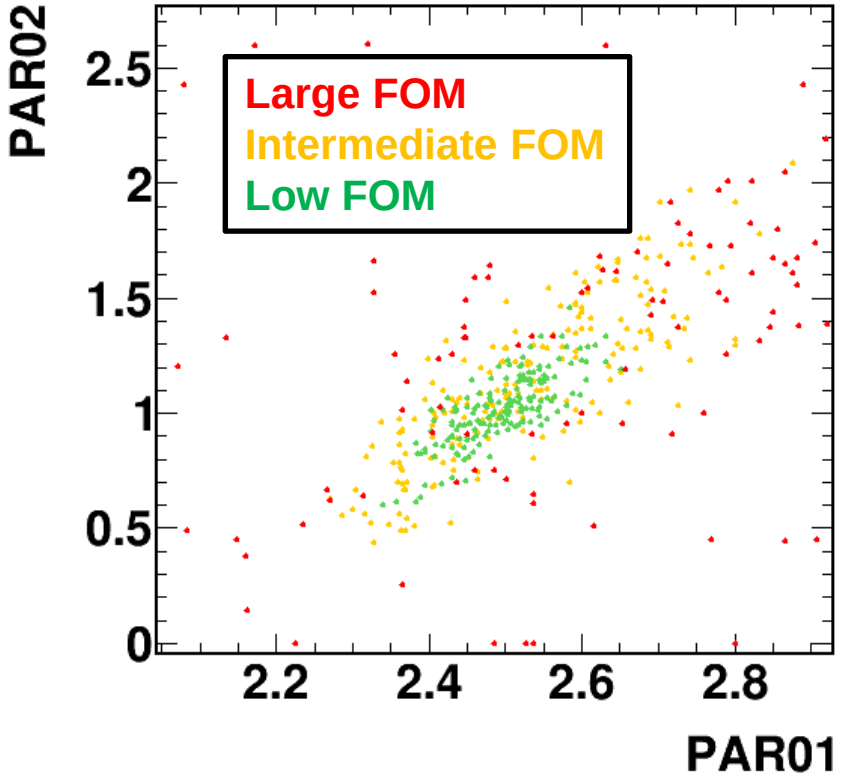




Left: Evolution of the FOM with the generation: largest and lowest FOM value, and average over all agents.
Right: evolution of the value of parameter 5.

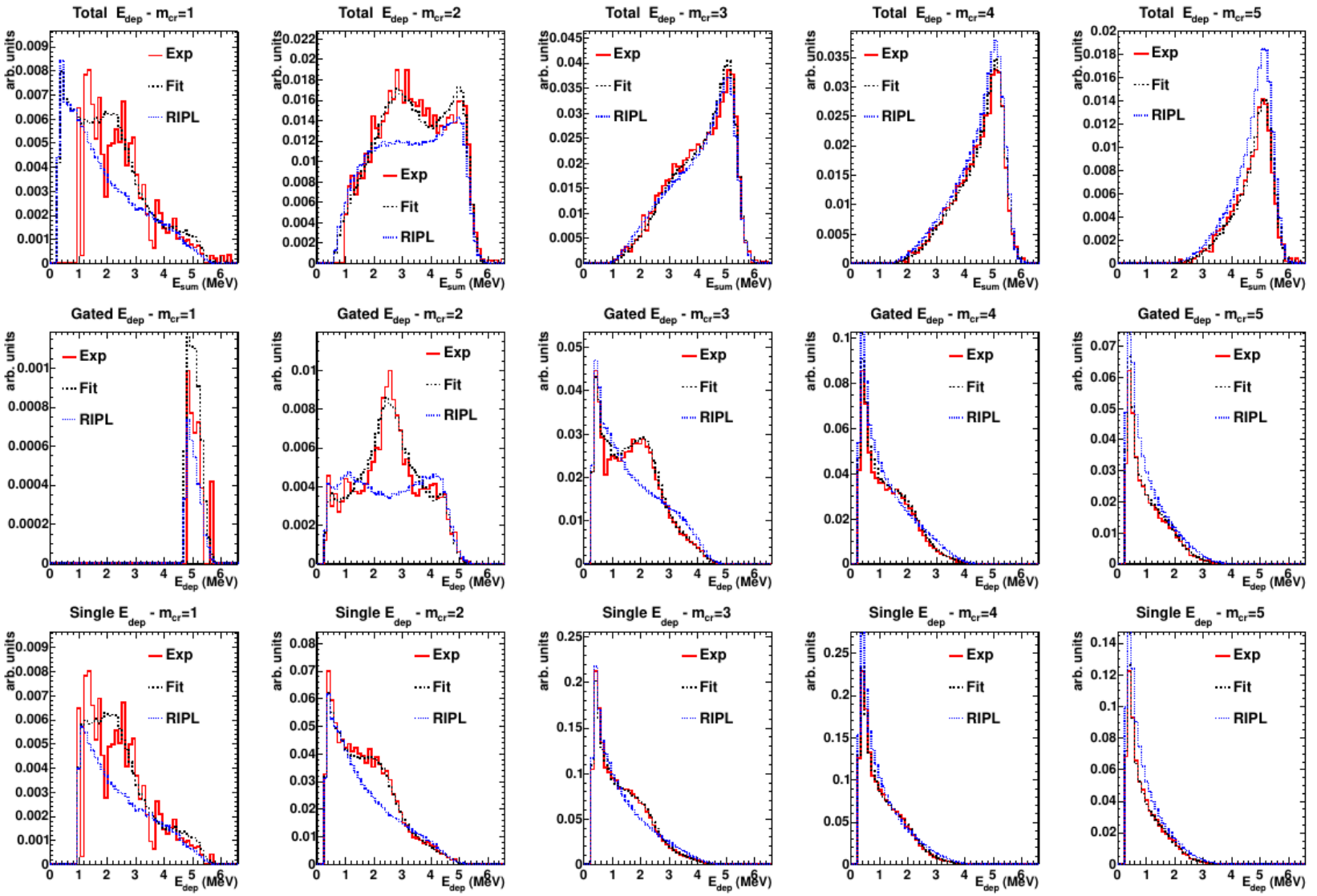
Convergence of the method



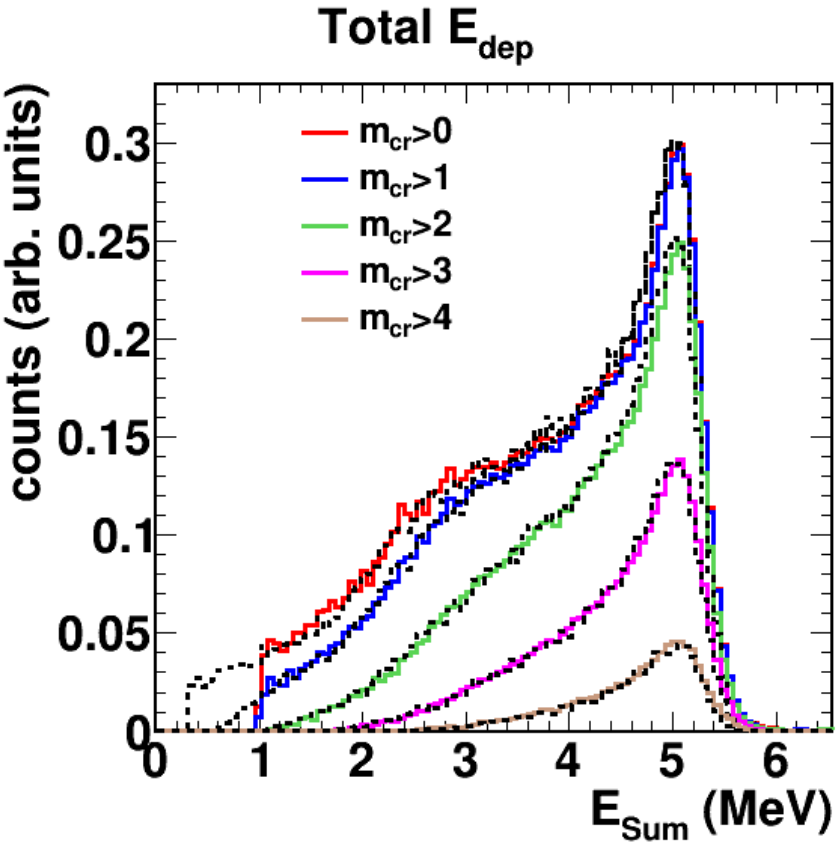


Left: Evolution of the FOM with the generation: largest and lowest FOM value, and average over all agents.
Right: scatter plot of the values of parameters 1 and 2.

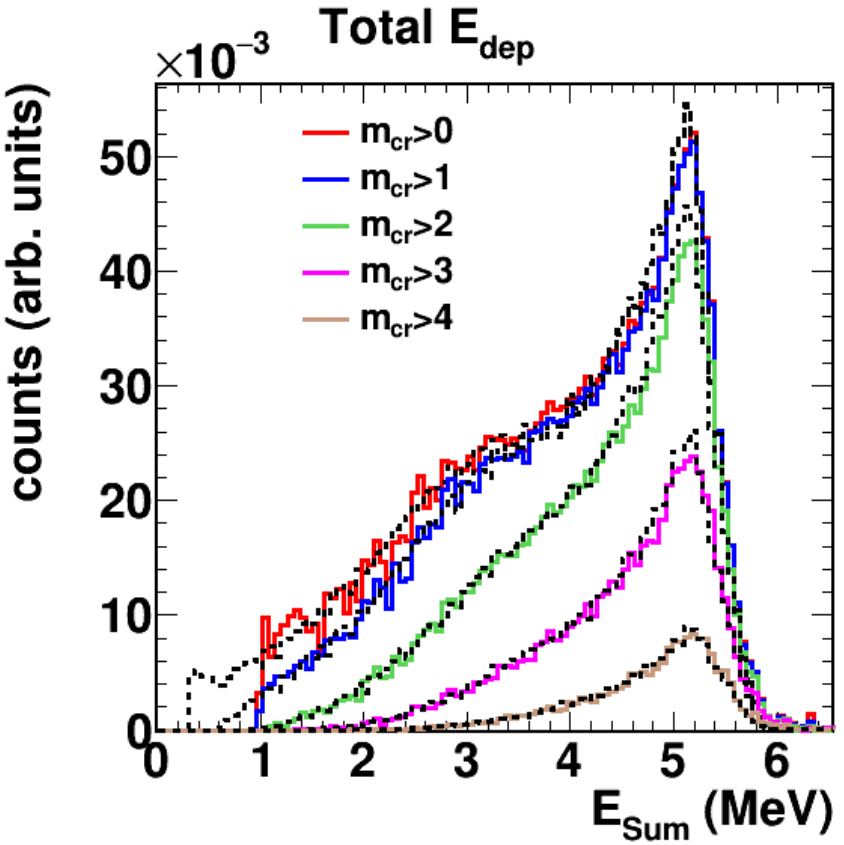
Results from the fit – ^{244}Cm(n,γ)



Convergence of the method



^{240}Pu(n,γ)



^{244}Cm(n,γ)

The methodology has been used to calculate the detection efficiency of the TAC to ^{240}Pu(n,γ) and ^{244}Cm(n,γ) cascades.

Conclusions

We have used the differential evolution algorithm to fit the parameters of a model capable of generating neutron capture cascades.

The purpose of this work is:

1. To calculate the detection efficiency of γ-ray detectors ✓
2. To perform nuclear structure studies → work in progress

Differential evolution algorithm:

- no derivatives ($f(\lambda_1, \lambda_2, \ldots, \lambda_N)$ doesn't need to be even continuous → *good* for noisy functions)
- robust
- very easy to implement
- easy to use in parallel computing

Refs. of our work:

E. Mendoza et al., EPJ Web Conf. 239, 01015 (2020), https://doi.org/10.1051/epjconf/202023901015

V. Alcayne et al., EPJ Web Conf. 239, 01034 (2020), https://doi.org/10.1051/epjconf/202023901034