Olivier Truffinet

Supervisors: Karim Ammar, Bertrand Bouriquet

DE LA RECHERCHE À L'INDUSTRIE

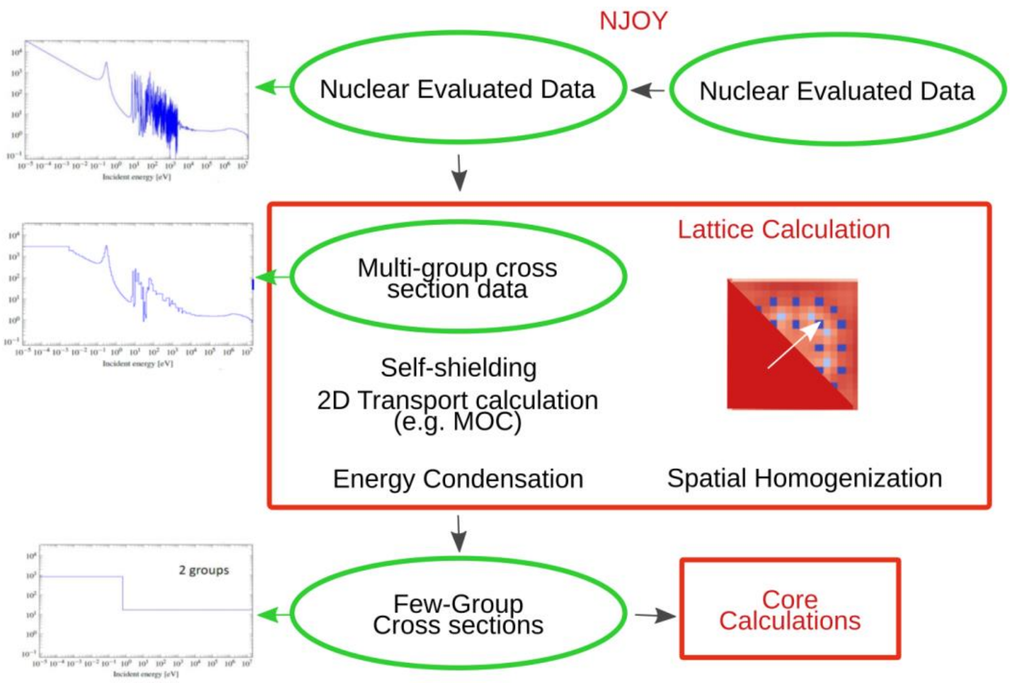
Compression and extrapolation of homogenized cross-sections by the EIM method

Sommaire

- Introduction: cross-sections and their reconstruction
- The Empirical Interpolation Method (EIM)
- The Big Data framework
- Experimental results
- Conclusion and perspectives

Introduction

Deterministic codes for neutronics calculation



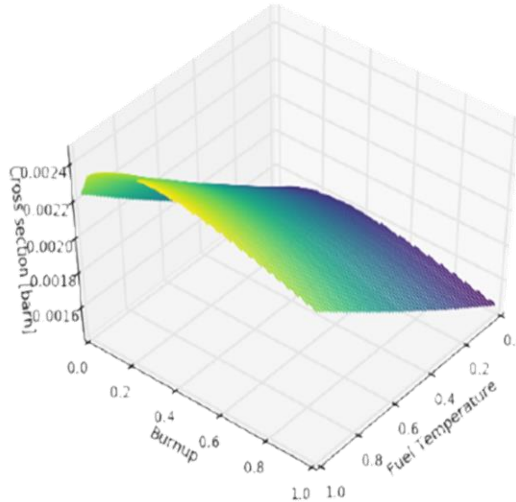
Source: E.Szames, Few group cross section modeling by machine learning for nuclear reactor

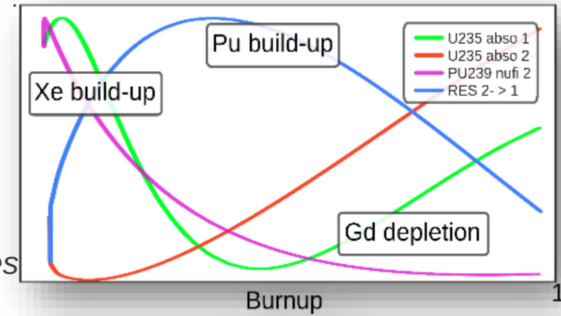
Homogenized cross-sections

- **Problem** : creating and storing a large number of <u>(strongly) correlated</u> multivariate surrogate models, one for each cross-section.

- Double constraint : very high precision ($\sim 10^{-3} - 10^{-4}$ relative error), low memory footprint

- Characteristics of the data:
  - Very little noise
  - Smooth, often monotonic
  - Low polynomial order with respect to most variables, one variable more erratic (burnup)

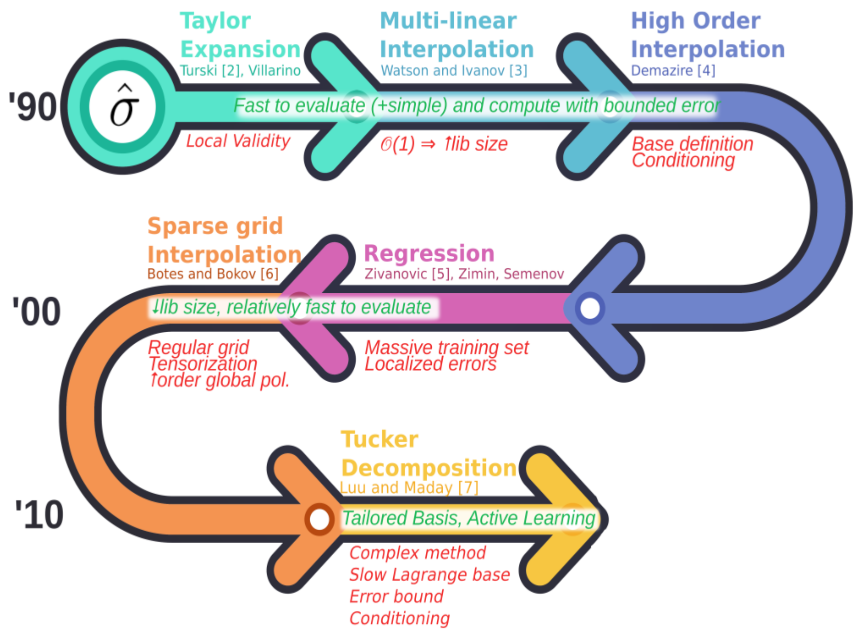




Source: E.Szames

State of the art

- Most common method: multilinear interpolation
- Strengths: simplicity, computation time, error control
- Problems: accuracy, number of coefficients, curse of dimensionality
- Consequence: the size of cross-sections libraries of is exploding (several hundreds of Gb)!



Source: E.Szames, *Few group cross section modeling by machine learning for nuclear reactor*

Empirical Interpolation Method (EIM)

Empirical interpolation method (EIM)

- Algorithm originally designed for approximation of nonlinear parametric functions in reduced basis methods for PDEs
- Base-coefficients decomposition : estimator $\hat{f}(\vec{x},\mu) \approx \sum_{i=1}^{r} c(\mu)\, B(\vec{x}) \approx f(\vec{x},\mu)$
- Coefficients $\mathrm{c}(\mu)$ are obtained by inversion of the interpolation system :

$$\hat{f}\left(\vec{x}_{\rho_i},\mu\right) = f\left(\vec{x}_{\rho_i},\mu\right)\ \forall\, i\ \in\ [\![1;r]\!]$$, where the ρ_i are the interpolation points

- Matrix form : $\boldsymbol{F} \approx \boldsymbol{C}\cdot\boldsymbol{B}$, where :
  - $\boldsymbol{F}$ is a matrix of values of the fonction f(in the format $\mu \otimes \vec{x}$), of size $n\ \times p$;
  - $\boldsymbol{C}$ is a matrix of coefficients, of size $n\ \times r$ with $\mathrm{r} \ll p$;
  - $\boldsymbol{B}$ is a matrix of basis vectors (non-orthogonals), of size r$\times\ p$.

Application to matrix compression

- We replace "values of a parametric function" by "matrix with strongly correlated lines". We're always looking for a decomposition $\boldsymbol{F} \approx \boldsymbol{C} \cdot \boldsymbol{B}$

- Interpolation : $\boldsymbol{C} = \boldsymbol{F}\boldsymbol{P}_{\rho}(\boldsymbol{B}\boldsymbol{P}_{\rho})^{-1}$, with $\boldsymbol{P}_{\rho}$ the projection matrix on the interpolation points (denoted ρ_i)

- Interest for matrix compression: the factorized form contains only $nr + rp = r(n+p)$ floating numbers (avec r$\ll n, p$) instead of np for the original matrix

- Compression rate : $\mathcal{R} = \frac{np}{r(n+p)} \in \left[\frac{\min(n,p)}{2r}, \frac{\min(n,p)}{r}\right]$

→Linear structure ⇒ very fast decompression, commutative with some operations (linear interpolation, linear combinations, slicing…)

Application to *surrogate modelling*

- We suppose that the matrix is made of independent outputs of a physics code: we can generate it column by column

- Let $\boldsymbol{f} \epsilon \mathbb{R}^p$ be a matrix row, $\tilde{\boldsymbol{f}} \epsilon \mathbb{R}^r$ its values at the points $\rho_1, \dots, \rho_r$.

We can compute the compressed coefficients of $\boldsymbol{f}$ by $\mathbf{c} = \boldsymbol{f}(\widetilde{\boldsymbol{BP}_{\boldsymbol{\rho}}})^{-1}$, then reconstruct the full vector $\boldsymbol{f}$: $\boldsymbol{f} \approx \mathbf{c} \cdot \boldsymbol{B}$ *(interpolation procedure)*

- In other words: if we already have a base and interpolation points, we can generate the physics code outputs at the points ρ_i only, and interpolate all the remaining values

/!\ The EIM method is discrete, restricted to the p support points : we cannot say anything outside these points ! /!\

Basis construction

- Greedy algorithm based on the infinite norm
- At each step, we add the line of data least well reproduced by the current model, and the point responsible for the largest error

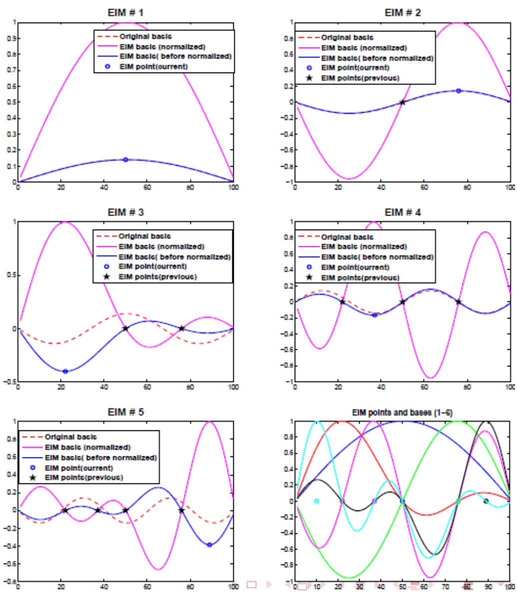


Fig.2: construction of the EIM base (source: private communication from S.Chaturantabut)

Algorithm 1 EIM algorithm

Input : matrix $\mathbf{F} = \begin{bmatrix} - & \mathbf{v}_1 & - \\ & \dots & \\ - & \mathbf{v}_n & - \end{bmatrix}$, factors rank r

Initialization :

$$\mathbf{v}_{max} = max_{\|\cdot\|_\infty}(\mathbf{v}_1, \dots, \mathbf{v}_n)$$
$$\rho_1 = argmax_{\|\cdot\|_\infty}(\mathbf{v}_1, \dots, \mathbf{v}_n)$$
$$\mathbf{b}_1 = \frac{\mathbf{v}_{max}}{\|\mathbf{v}_{max}\|_\infty}$$
$$\mathbf{P}_\rho = [\mathbf{e}_{\rho_1}]$$
$$\mathbf{B} = \begin{bmatrix} - & b_1 & - \end{bmatrix}$$

Iteration :
for $i = 1, \dots, r$ **do**

$$\mathbf{C} = \mathbf{F}\mathbf{P}_\rho(\mathbf{B}\mathbf{P}_\rho)^{-1}$$
$$\mathbf{R} = \begin{bmatrix} - & \mathbf{r}_1 & - \\ & \dots & \\ - & \mathbf{r}_n & - \end{bmatrix} = \mathbf{F} - \mathbf{C}\mathbf{B}$$
$$\mathbf{v}_{max} = max_{\|\cdot\|_\infty}(\mathbf{r}_1, \dots, \mathbf{r}_n)$$
$$\rho_i = argmax_{\|\cdot\|_\infty}(\mathbf{r}_1, \dots, \mathbf{r}_n)$$
$$\mathbf{b}_i = \frac{\mathbf{v}_{max}}{\|\mathbf{v}_{max}\|_\infty}$$
$$\mathbf{P}_\rho = [\mathbf{e}_{\rho_1}, \dots, \mathbf{e}_{\rho_i}]$$
$$\mathbf{B} = \begin{bmatrix} - & \mathbf{b}_1 & - \\ & \dots & \\ - & \mathbf{b}_i & - \end{bmatrix}$$

end for

Output : basis $\mathbf{B}$, magic points $\mathbf{P}_\rho$

Big Data Framework

Description of the data

- Use of a widely-used benchmark to operate on representative data. Set of 12 fuel assemblies sufficient to perform a core calculation.
- Resulting matrix : 1.2 million rows, 4704 columns → 60 Go in HDF5 format. Doesn't easily fit in RAM…

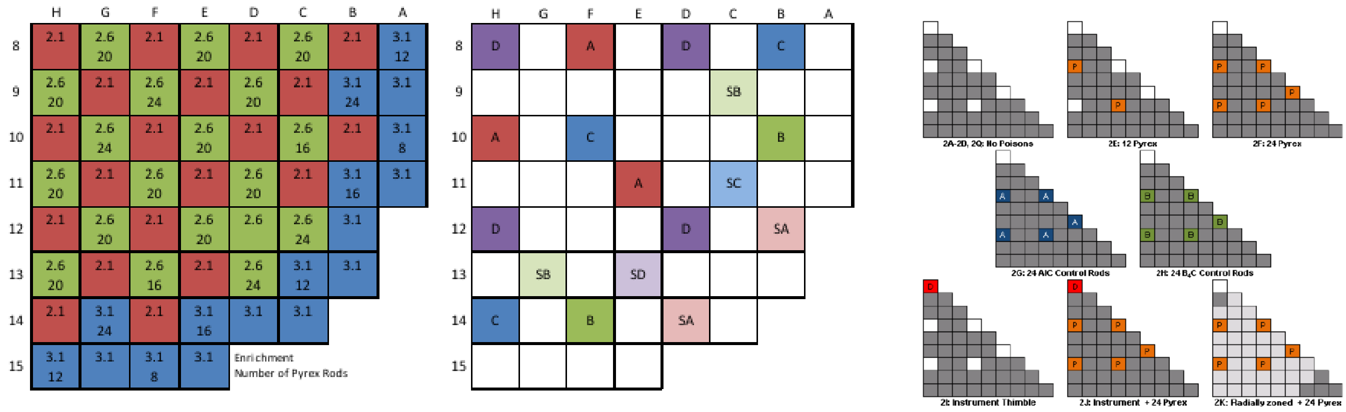
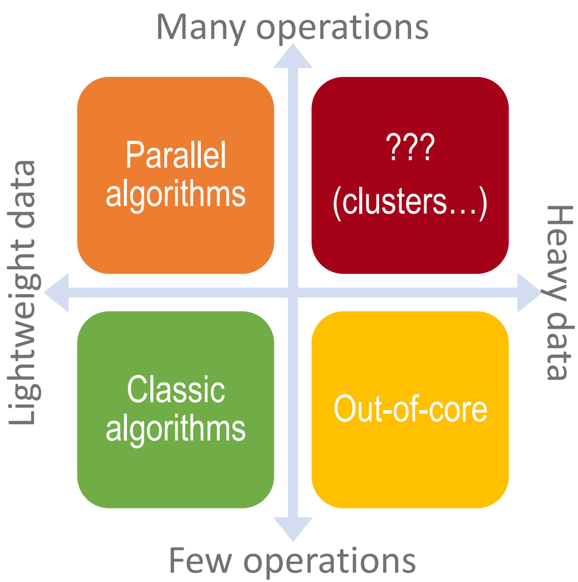


Fig.3: Description of the VERA benchmark (problem n°5)

Parallelism and out-of-core computation

- Out-of-core = computing environment where the allocated RAM space is too small to perform the calculation
- In this case, one must:
    - Read the data directly from slow bulk memory;
    - Minimize the number of passes on the data;
    - Minimize the memory footprint (no large volumes of intermediate data);
    - Allocate RAM intelligently
- Counter-intuitively, parallelism is of little use in this case, since performance is capped by the slow read (I/O) anyway

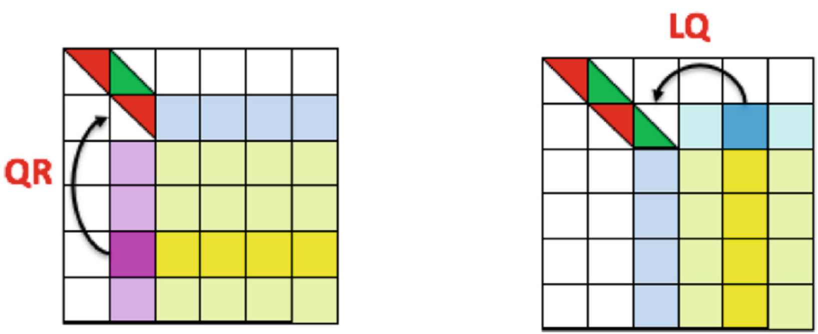


COMPRESSION AND EXTRAPOLATION OF HOMOGENIZED CROSS-SECTIONS BY THE EIM METHOD

- Algorithms exist for out-of-core SVD, but they are recent, complex and not well implemented in accessible libraries

- **EIM adapts readily to out-of-core and parallel computation** : the computation of residues (central step of the database construction) is done independently line by line, and its output is a single value that is easy to store

- A Python library for the management of very large data: the `Dask` library (dynamic task scheduling)

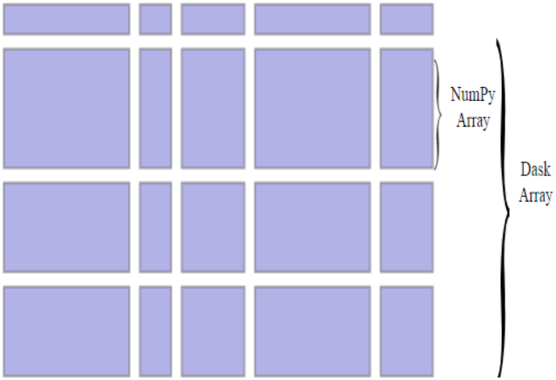


(a) QR factorization of tile $A_{2,2}$ (b) LQ factorization of tile $A_{2,3}$

Fig.5: Representation of an exact out-of-core SVD algorithm (source : K.Kabir & al, *A Framework for Out of Memory SVD Algorithms)*

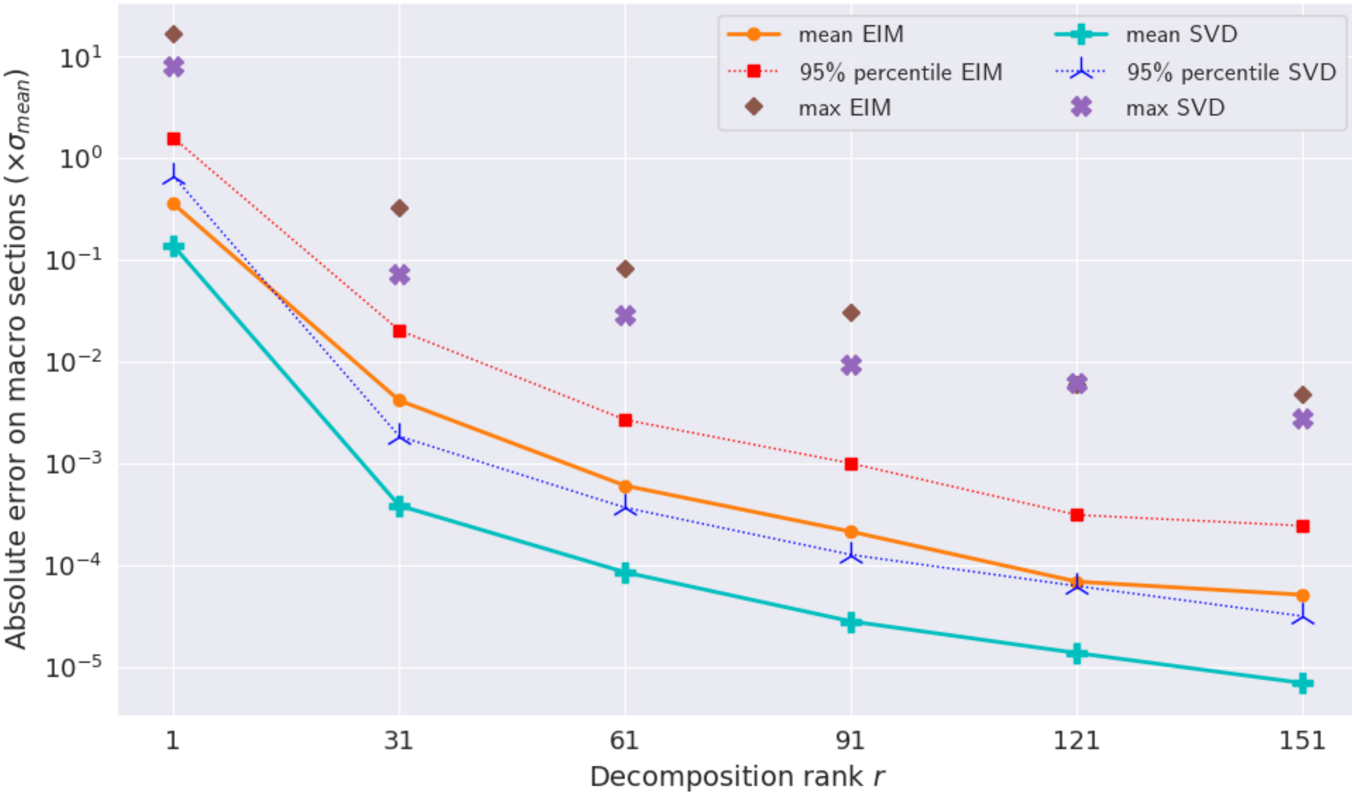
Stochastic EIM

- Even if compression time is not a decisive criterion of performance (offline phase), compressing the whole dataset can be long (1-2h)
- Cause : EIM processes the whole dataset at each iteration, even though it is extremely redundant
- One could rather choose to read only a subset of rows at each iteration and pick the future basis function among these. If the data is already chunked, these subsets can be horizontal groups of chunks
- Very efficient scheme (tested in the next part) !

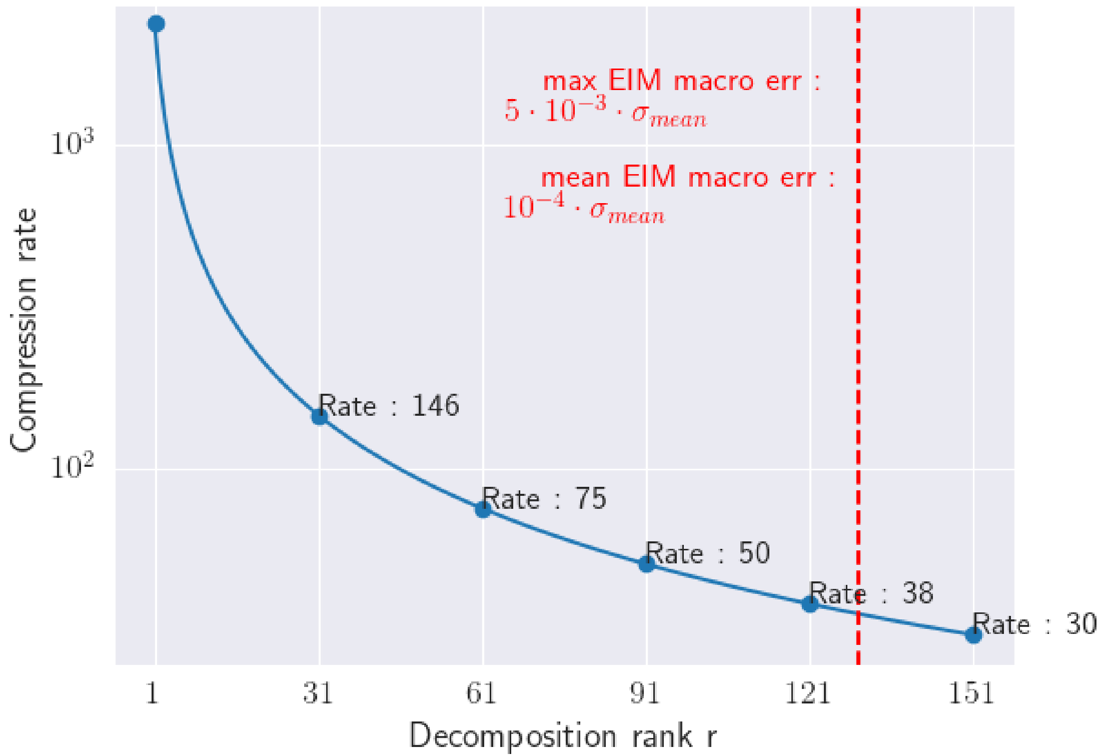


Numerical experiments

COMPRESSION AND EXTRAPOLATION OF HOMOGENIZED CROSS-SECTIONS BY THE EIM METHOD

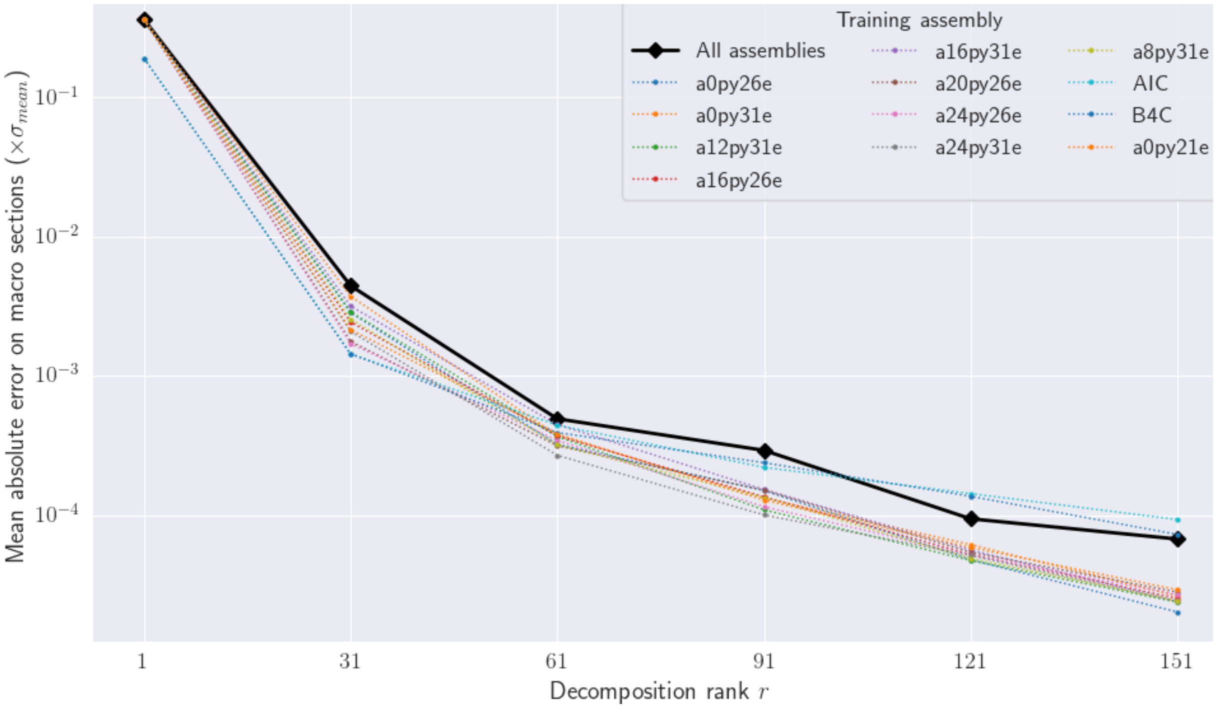


- For compression only, EIM worse than SVD by a small factor (<10)
- Polynomial decay of the error on this data



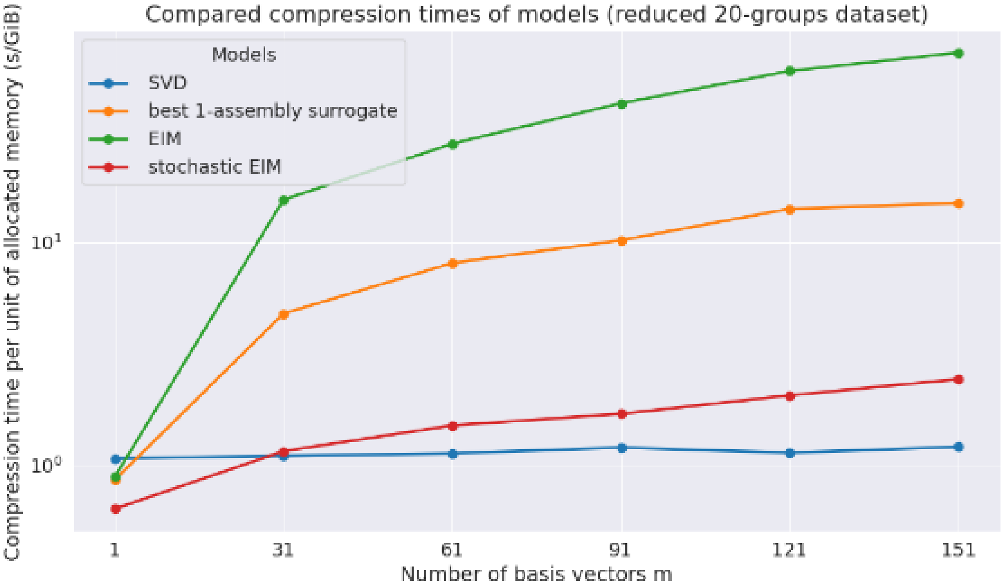
- Compression rates of several dozens achieved.
- The limiting factor is the small side of the matrix → use of tensor methods ?

COMPRESSION AND EXTRAPOLATION OF HOMOGENIZED CROSS-SECTIONS BY THE EIM METHOD



- Impressive extrapolation performance : for most metrics, extrapolated data is more accurate than compressed data !
- Robust to the choice of the training set in our case (except for pathological cases)
- With extrapolation, 88% of overall physics computation was spared

Compression time performance



- Classical EIM is handicapped by its numerous passes on the data
- Stochastic EIM is 40 times faster, and just as accurate
- Surrogate EIM is faster because it works on less data

Performance recap

	r=150, $\mathcal{R}$=30			
Model	Err_{mean}^{rel} $(\cdot 10^{-5})$	Err_{max}^{abs} $(\cdot 10^{-5}\sigma_{moy})$	T_{comp} (s/GiB)	T_{tot} (s)
SVD	2	284	1	9E4
EIM	10	481	71	9E4
Stochastic EIM	8	439	1.1	9E4
Surrogate	8	247	17	1E4

Disgression : Least Squares Extrapolation

- Data extrapolation relies on the fact that with EIM, dimension reduction makes use of a subset of the data matrix columns only
- Same could be applied to Least Squares Regression (itself linked to SVD) if we could find relevant column indices to sample at
- Methods to perform such sampling have been developed recently: see Cohen and Migliorati, *Optimal weighted least-squares methods.* Only pitfall : the number of sampling points must be greater than the rank (typically, $3r$ points are needed for stability)
- I started comparing both methods… The game is close !

Conclusion et ouvertures

Conclusion

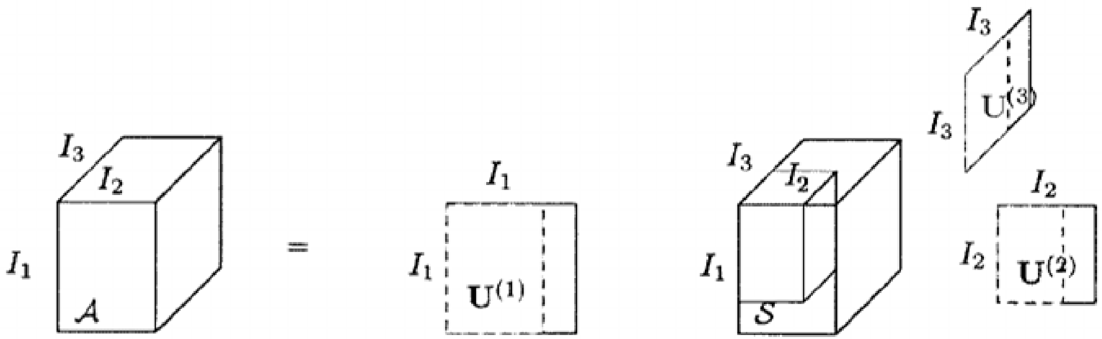
- New use of EIM for compression and extrapolation of physical data
- Simple, linear method, easy to implement even in out-of-core or massively parallel contexts
- Linear decompression commutative with some post-processing operations (slicing, linear combinations, interpolation), allowing for very efficient routines
- Experimental results on a challenging dataset (60 GB):
  - Memory savings of a factor 30 for an average relative error $< 10^{-5}$ and a maximum error $< 5 \cdot 10^{-3}$
  - Reduction of the number of code calls by a factor of 8

Looking for new application cases !

- Other physical problems on which to apply this methodology ?
- Desired characteristics :
  - Costly simulation to run for a large number of parameters values (not necessarily a grid)
  - One parameter (continuous or discrete) with many values can be isolated from the other. This is the one that will be extrapolated
  - It’s okay for the resulting snapshots to be correlated
  - Ideally, compression of the data is of interest

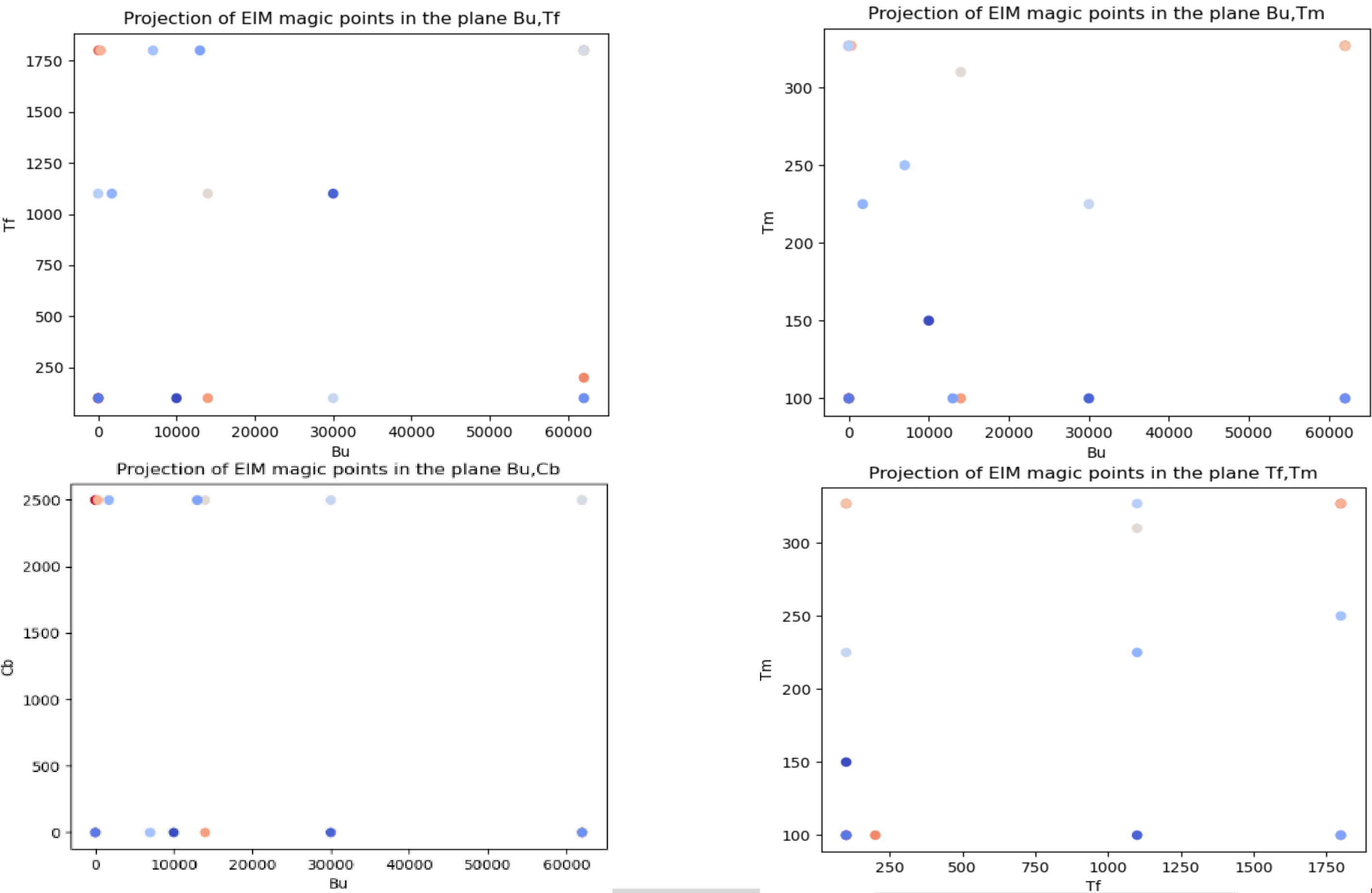
Development: HOOI and its EIM adaptation

- To improve performance even more, it could be interesting to compress the data along several distinct axes (data in tensor form)
- Optimal algorithm for this : Higher Order Orthogonal Iteration. Same problems as SVD for out-of-core. Slower decompression
- First encouraging results: compression ratio x5 with the same average precision!
- Adaptable to the EIM ; creation of a HOEIM? Convergence not guaranteed... Use of the aforementioned least-squares sampling method ?



MERCI POUR VOTRE ATTENTION !

COMPRESSION AND EXTRAPOLATION OF HOMOGENIZED CROSS-SECTIONS BY THE EIM METHOD



Un critère d'arrêt empirique en loi de puissance ?

Quelques lectures sur l'EIM

- M. &. al, **«An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations»,** *Compte-rendus de l'académie des Sciences*, Paris, 2004.

- S. Kristoffersen, **«The Empirical Interpolation Method»,** 2013.

- H.Gong, **«Data assimilation with reduced basis and noisy measurement : Applications to nuclear reactor cores»**, 2018.

(application à la reconstruction du champ neutronique)