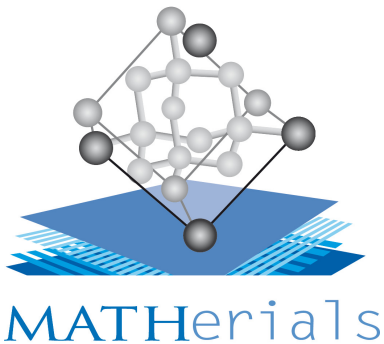
Numerical methods for high-dimensional problems: examples from materials science applications

Virginie Ehrlacher[1,2]

[1] Ecole des Ponts ParisTech

[2] INRIA, MATHERIALS project-team



20 ans du groupe Calcul, 3rd June 2024

High-dimensional approximation

Goal: find an approximation of a **high-dimensional function**

$$u(x_1, \ldots, x_d)$$

or a **set of high-dimensional functions**

$$K$$

by some simple functions (i.e. easy to evaluate) depending only on **a few parameters**

High-dimensional approximation

Goal: find an approximation of a **high-dimensional function**

$$u(x_1, \ldots, x_d)$$

or a **set of high-dimensional functions**

$$K$$

by some simple functions (i.e. easy to evaluate) depending only on **a few parameters**

High-dimensional approximation

Goal: find an approximation of a **high-dimensional function**

$$u(x_1, \ldots, x_d)$$

or a **set of high-dimensional functions**

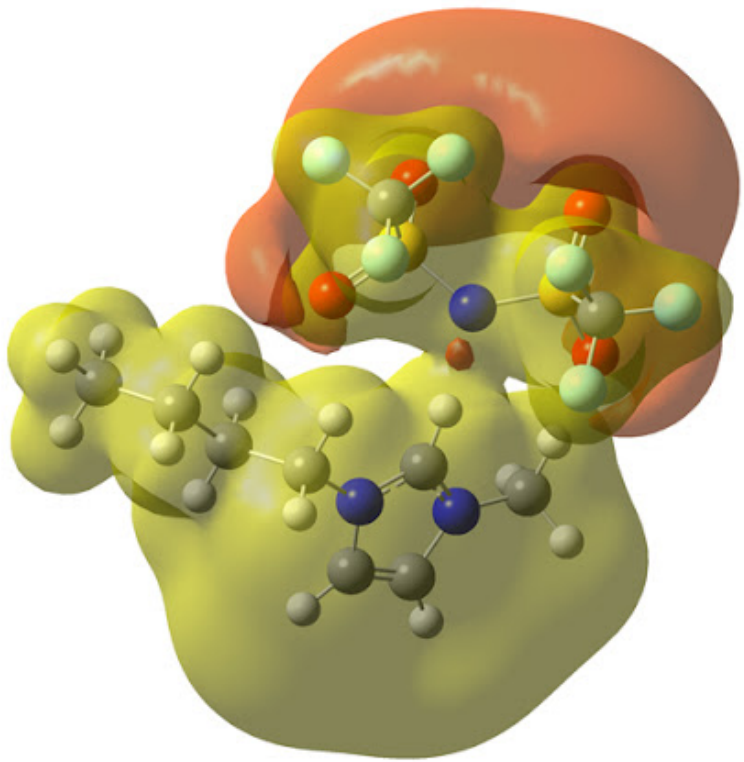
$$K$$

by some simple functions (i.e. easy to evaluate) depending only on **a few parameters**

Some high-dimensional problems in materials science: electronic structure calculations

- **Schrödinger equation:** $\Psi(x_1, \ldots, x_d, t)$

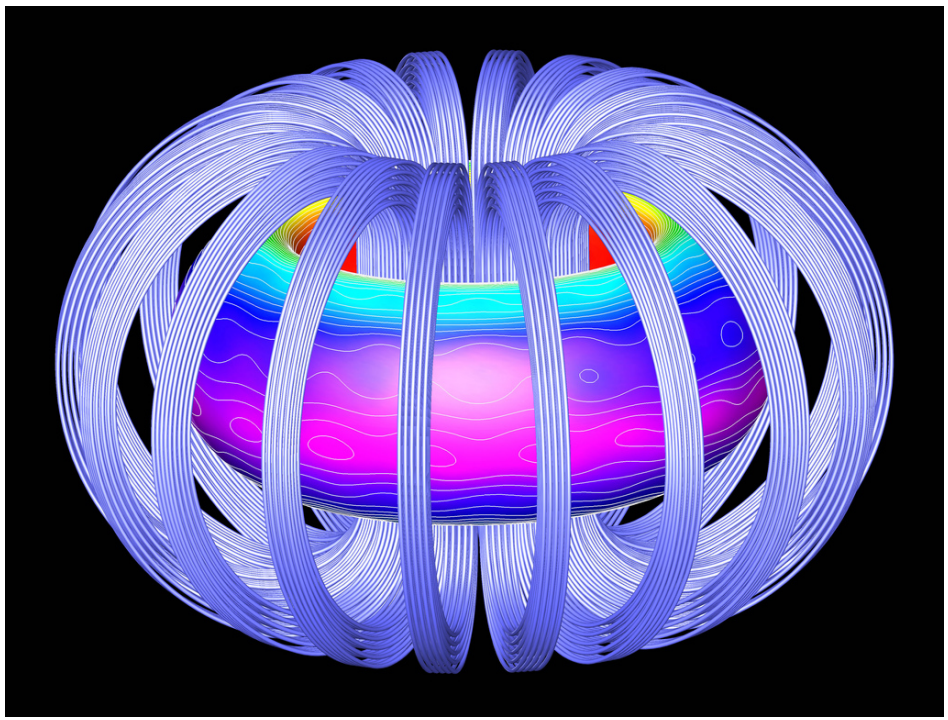
$$i\hbar\partial_t\Psi = -\frac{\hbar}{2m}\sum_{i=1}^{d}\Delta_{x_i}\Psi + V\Psi$$



Some high-dimensional problems in materials science: kinetic equations

- **Boltzmann equation:** $p(x_1, \ldots, x_d, v_1, \ldots, v_d, t)$

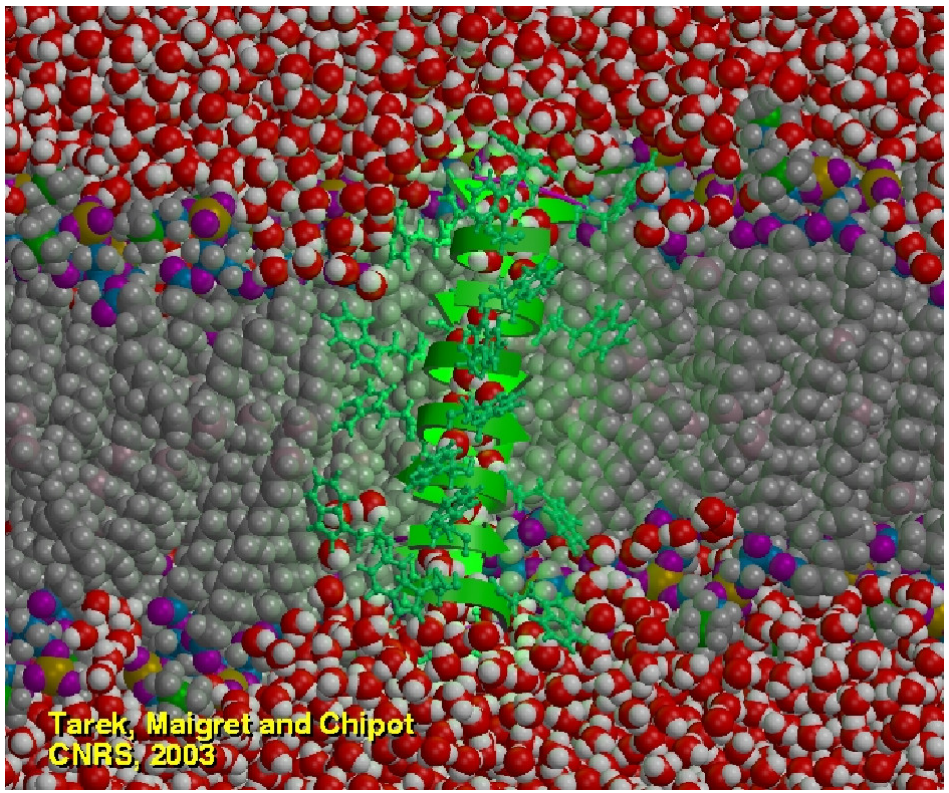
$$\partial_t p + \sum_{i=1}^{d} v_i \, \partial_{x_i} p + \sum_{i=1}^{d} F_i \, \partial_{v_i} p = H(p, p)$$



Some high-dimensional problems in materials science: molecular dynamics

- **Fokker-Planck equation:** $p(x_1, \ldots, x_d, t)$

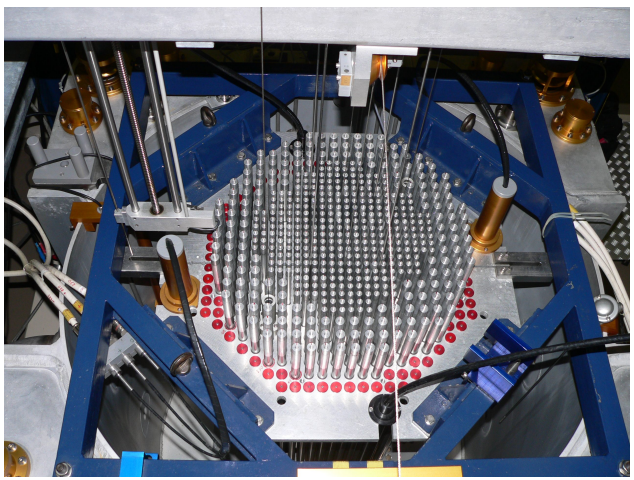
$$\partial_t p + \sum_{i=1}^{d} \partial_{x_i}(a_i p) - \frac{1}{2} \sum_{i=1}^{d} \sum_{j=1}^{d} \partial^2_{x_i x_j}(b_{ij} p) = 0.$$

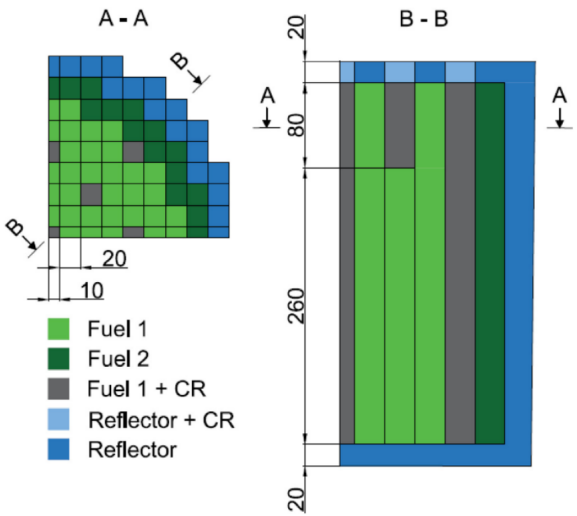


Some high-dimensional problems in materials science: neutronics

- **Parametrized equation:** $u(\mu; x) \quad \mu = (\mu_1, \ldots, \mu_p)$

$$\mathcal{A}(\mu; u(\mu; \cdot)) = f(\mu; \cdot)$$





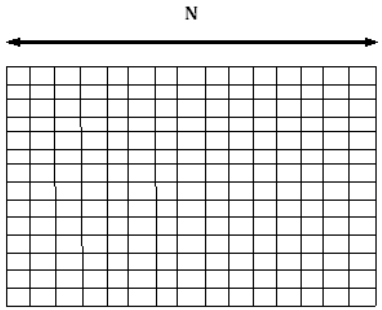
Curse of dimensionality: first intuition

Goal: find an approximation of a **high-dimensional function**

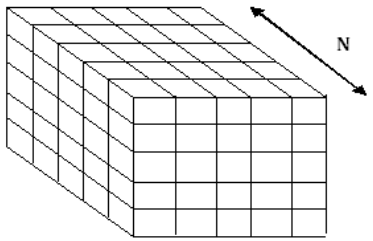
$$u(x_1, \ldots, x_d)$$

Standard discretization approaches (such as classical finite element methods) suffer from the so-called **curse of dimensionality** [Bellman,1961]

$COMP = N^2$



$COMP = N^3$



$COMP = N^d$

High-dimensional approximation

Let us assume that u is an element of a given set of functions V (a normed vector space for instance).

- For a certain subset of functions $Z_n \subset V$ described by n parameters, the error of **best approximation** of u by elements of Z_n is defined by

$$e_{Z_n}(u) = \inf_{v \in Z_n} \|u - v\|_V$$

- A sequence of subsets $(Z_n)_{n \geq 1}$ is called an **approximation tool**.

  Two types of approaches:
  - linear approximation: Z_n are linear spaces
  - nonlinear approximation: Z_n are **not** linear spaces

High-dimensional approximation

Let us assume that u is an element of a given set of functions V (a normed vector space for instance).

- For a certain subset of functions $Z_n \subset V$ described by n parameters, the error of **best approximation** of u by elements of Z_n is defined by

$$e_{Z_n}(u) = \inf_{v \in Z_n} \|u - v\|_V$$

- A sequence of subsets $(Z_n)_{n \geq 1}$ is called an **approximation tool**.

  Two types of approaches:
  - linear approximation: Z_n are linear spaces
  - nonlinear approximation: Z_n are **not** linear spaces

High-dimensional approximation

Let us assume that u is an element of a given set of functions V (a normed vector space for instance).

- For a certain subset of functions $Z_n \subset V$ described by n parameters, the error of **best approximation** of u by elements of Z_n is defined by

$$e_{Z_n}(u) = \inf_{v \in Z_n} \|u - v\|_V$$

- A sequence of subsets $(Z_n)_{n \geq 1}$ is called an **approximation tool**.

Two types of approaches:

- linear approximation: Z_n are linear spaces
- nonlinear approximation: Z_n are **not** linear spaces

High-dimensional approximation

Let us assume that u is an element of a given set of functions V (a normed vector space for instance).

- For a certain subset of functions $Z_n \subset V$ described by n parameters, the error of **best approximation** of u by elements of Z_n is defined by

$$e_{Z_n}(u) = \inf_{v \in Z_n} \|u - v\|_V$$

- A sequence of subsets $(Z_n)_{n \geq 1}$ is called an **approximation tool**.

  Two types of approaches:
  - linear approximation: Z_n are linear spaces
  - nonlinear approximation: Z_n are **not** linear spaces

High-dimensional approximation

Fundamental problems are:

- to determine if and how fast $e_{Z_n}(u)$ goes to 0 for a certain function u (or a set of functions K) and a certain approximation tool $(Z_n)_{n\geq 1}$;
- to provide algorithms which (hopefully) produce approximations $u_n \in Z_n$ of u such that

$$\|u - u_n\|_V \leq C e_{Z_n}(u),$$

with

(i) either C independent of n;
(ii) or $C(n)e_{Z_n}(u) \underset{n\to+\infty}{\longrightarrow} 0$.

- to provide a posteriori error estimators to estimate the error $\|u - u_n\|_V$ in practice.

Fundamental problems are:

- to determine if and how fast $e_{Z_n}(u)$ goes to 0 for a certain function u (or a set of functions K) and a certain approximation tool $(Z_n)_{n\geq 1}$;
- to provide algorithms which (hopefully) produce approximations $u_n \in Z_n$ of u such that

$$\|u - u_n\|_V \leq C e_{Z_n}(u),$$

with

(i) either C independent of n;
(ii) or $C(n)e_{Z_n}(u) \underset{n\to+\infty}{\longrightarrow} 0$.

- to provide a posteriori error estimators to estimate the error $\|u - u_n\|_V$ in practice.

Fundamental problems are:

- to determine if and how fast $e_{Z_n}(u)$ goes to 0 for a certain function u (or a set of functions K) and a certain approximation tool $(Z_n)_{n\geq 1}$;
- to provide algorithms which (hopefully) produce approximations $u_n \in Z_n$ of u such that

$$\|u - u_n\|_V \leq C e_{Z_n}(u),$$

with

(i) either C independent of n;
(ii) or $C(n)e_{Z_n}(u) \underset{n\to+\infty}{\longrightarrow} 0$.

- to provide a posteriori error estimators to estimate the error $\|u - u_n\|_V$ in practice.

Fundamental problems are:

- to determine if and how fast $e_{Z_n}(u)$ goes to 0 for a certain function u (or a set of functions K) and a certain approximation tool $(Z_n)_{n\geq 1}$;
- to provide algorithms which (hopefully) produce approximations $u_n \in Z_n$ of u such that

$$\|u - u_n\|_V \leq C e_{Z_n}(u),$$

with

(i) either C independent of n;
(ii) or $C(n)e_{Z_n}(u) \underset{n\to+\infty}{\longrightarrow} 0$.

- to provide a posteriori error estimators to estimate the error $\|u - u_n\|_V$ in practice.

Linear approximations: Kolmogorov n-width

For a set of functions K in a normed vector space V, the **Kolmogorov n-width** of K is defined as

$$d_n(K) = \inf_{Z_n \subset V} \sup_{u \in K} \inf_{v \in Z_n} \|u - v\|_V$$

where the first infimum is taken over all linear subspaces Z_n of V of dimension n.

The Kolmogorov width $d_n(K)$ measures how well functions belonging to the set K can be approximated by an n-dimensional linear space. It measures the **ideal performance** that we can expect from **linear approximation methods**.

Example: Let $V = L^p((0,1)^d)$ and K the unit ball of $W^{k,p}((0,1)^d)$.
Then, we have

$$d_n(K) \sim n^{-k/d}$$

Linear approximations: Kolmogorov n-width

For a set of functions K in a normed vector space V, the **Kolmogorov n-width** of K is defined as

$$d_n(K) = \inf_{Z_n \subset V} \sup_{u \in K} \inf_{v \in Z_n} \|u - v\|_V$$

where the first infimum is taken over all linear subspaces Z_n of V of dimension n.

The Kolmogorov width $d_n(K)$ measures how well functions belonging to the set K can be approximated by an n-dimensional linear space. It measures the **ideal performance** that we can expect from **linear approximation methods**.

Example: Let $V = L^p((0,1)^d)$ and K the unit ball of $W^{k,p}((0,1)^d)$.
Then, we have

$$d_n(K) \sim n^{-k/d}$$

Linear approximations: Kolmogorov n-width

For a set of functions K in a normed vector space V, the **Kolmogorov n-width** of K is defined as

$$d_n(K) = \inf_{Z_n \subset V} \sup_{u \in K} \inf_{v \in Z_n} \|u - v\|_V$$

where the first infimum is taken over all linear subspaces Z_n of V of dimension n.

The Kolmogorov width $d_n(K)$ measures how well functions belonging to the set K can be approximated by an n-dimensional linear space. It measures the **ideal performance** that we can expect from **linear approximation methods**.

Example: Let $V = L^p((0,1)^d)$ and K the unit ball of $W^{k,p}((0,1)^d)$.
Then, we have

$$d_n(K) \sim n^{-k/d}$$

How to beat the curse of dimensionality?

The key is to consider **sets of functions with specific low-dimensional structures** and to propose approximation tools (**formats**) which exploit these structures (**application-dependent**).

Possible approaches:

- build a sequence of linear approximation spaces $(Z_n)_{n\geq 1}$ specifically taylored to the targeted application (reduced basis methods...)
- nonlinear approximation tools (tensor methods, neural networks...)
- Combine both worlds!

How to beat the curse of dimensionality?

The key is to consider **sets of functions with specific low-dimensional structures** and to propose approximation tools (**formats**) which exploit these structures (**application-dependent**).

Possible approaches:

- build a sequence of linear approximation spaces $(Z_n)_{n\geq 1}$ specifically taylored to the targeted application (reduced basis methods...)
- nonlinear approximation tools (tensor methods, neural networks...)
- Combine both worlds!

How to beat the curse of dimensionality?

The key is to consider **sets of functions with specific low-dimensional structures** and to propose approximation tools (**formats**) which exploit these structures (**application-dependent**).

Possible approaches:

- build a sequence of linear approximation spaces $(Z_n)_{n\geq 1}$ specifically taylored to the targeted application (reduced basis methods...)
- nonlinear approximation tools (tensor methods, neural networks...)
- Combine both worlds!

How to beat the curse of dimensionality?

The key is to consider **sets of functions with specific low-dimensional structures** and to propose approximation tools (**formats**) which exploit these structures (**application-dependent**).

Possible approaches:

- build a sequence of linear approximation spaces $(Z_n)_{n\geq 1}$ specifically taylored to the targeted application (reduced basis methods...)
- nonlinear approximation tools (tensor methods, neural networks...)
- Combine both worlds!

Outline of the talk

1. Reduced basis methods

2. Tensors and neural networks

Outline of the talk

Parametrized PDEs

- The behaviour of many systems can be described by the solutions of a system of Partial Differential Equations.
- These equations can depend on one or several parameters $\mu = (\mu_1, \cdots, \mu_p)$ with $p \in \mathbb{N}^*$ which can take values in a set denoted by $\mathcal{P} \subset \mathbb{R}^p$.
  In this case, for one particular value $\mu \in \mathcal{P}$ of this vector of parameters, the associated solution to the PDE system is a function u_μ solution of

$$\mathcal{A}(u_\mu; \mu) = 0,$$

where $\mathcal{A}(\cdot; \mu)$ is some differential operator depending on the parameter vector μ.

Here, the set of functions one wishes to consider is the **set of solutions to the parametric PDE**:

$$K = \{u_\mu, \ \mu \in \mathcal{P}\}$$

Parametrized PDEs

- The behaviour of many systems can be described by the solutions of a system of Partial Differential Equations.
- These equations can depend on one or several parameters $\mu = (\mu_1, \cdots, \mu_p)$ with $p \in \mathbb{N}^*$ which can take values in a set denoted by $\mathcal{P} \subset \mathbb{R}^p$.
  In this case, for one particular value $\mu \in \mathcal{P}$ of this vector of parameters, the associated solution to the PDE system is a function u_μ solution of

$$\mathcal{A}(u_\mu; \mu) = 0,$$

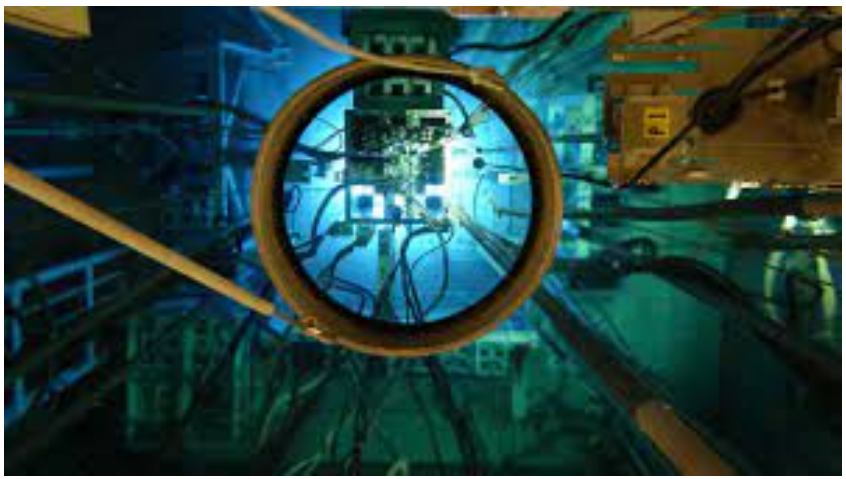
  where $\mathcal{A}(\cdot; \mu)$ is some differential operator depending on the parameter vector μ.

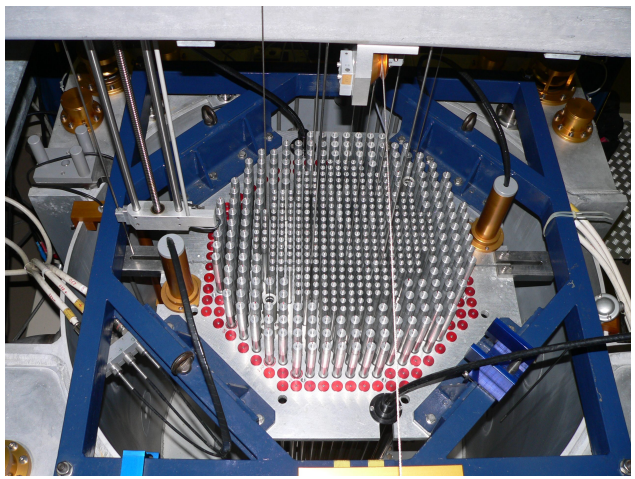
Here, the set of functions one wishes to consider is the **set of solutions to the parametric PDE**:

$$K = \{u_\mu, \ \mu \in \mathcal{P}\}$$

Research nuclear core reactor

joint work with Yonah Conjugo-Taumhas, Geneviève Dussson, Tony Lelièvre, François Madiot





Criticity calculation in a nuclear core reactor

Simple example: Two-group diffusion model

[Coste-Delclaux, Diop, Nicolas, Bonin, 2013], [Mula, 2014], [Giret, 2018], [Allaire, Blanc, Desprès, Golse, 2019]

- **Spatial domain** $\Omega \subset \mathbb{R}^d$ occupied by the nuclear core reactor
- Neutrons are assumed to be separated into **2 groups** according to their energy: $E = \{E_1, E_2\}$ $(E_1 > E_2)$
- $\mu \in \mathcal{P}$: vector of **parameters** of the problem, which encodes the values of the **physical properties** of the nuclear core

Problem of interest: Find

- $u_\mu = (u_{1,\mu}, u_{2,\mu}) : \Omega \to \mathbb{R}^2$: **neutron scalar fluxes**;
- $\lambda_\mu > 0$ eigenvalue with **smallest modulus**;

solution to the non-symmetric eigenvalue problem

$$\mathcal{A}_\mu u_\mu = \lambda_\mu \mathcal{B}_\mu = \frac{1}{k_\mu} \mathcal{B}_\mu u_\mu$$

where $\mathcal{A}_\mu$ and $\mathcal{B}_\mu$ are linear operators such that $\mathcal{A}_\mu^{-1}\mathcal{B}_\mu$ satisfies the assumptions of the **Krein-Rutman theorem**.

Criticity calculation in a nuclear core reactor

Simple example: Two-group diffusion model

[Coste-Delclaux, Diop, Nicolas, Bonin, 2013], [Mula, 2014], [Giret, 2018], [Allaire, Blanc, Desprès, Golse, 2019]

- **Spatial domain** $\Omega \subset \mathbb{R}^d$ occupied by the nuclear core reactor
- Neutrons are assumed to be separated into **2 groups** according to their energy: $E = \{E_1, E_2\}$ $(E_1 > E_2)$
- $\mu \in \mathcal{P}$: vector of **parameters** of the problem, which encodes the values of the **physical properties** of the nuclear core

Problem of interest: Find

- $u_\mu = (u_{1,\mu}, u_{2,\mu}) : \Omega \to \mathbb{R}^2$: **neutron scalar fluxes**;
- $\lambda_\mu > 0$ eigenvalue with **smallest modulus**;

solution to the non-symmetric eigenvalue problem

$$\mathcal{A}_\mu u_\mu = \lambda_\mu \mathcal{B}_\mu = \frac{1}{k_\mu} \mathcal{B}_\mu u_\mu$$

where $\mathcal{A}_\mu$ and $\mathcal{B}_\mu$ are linear operators such that $\mathcal{A}_\mu^{-1}\mathcal{B}_\mu$ satisfies the assumptions of the **Krein-Rutman theorem**.

Criticity calculation in a nuclear core reactor

Simple example: Two-group diffusion model

[Coste-Delclaux, Diop, Nicolas, Bonin, 2013], [Mula, 2014], [Giret, 2018], [Allaire, Blanc, Desprès, Golse, 2019]

- **Spatial domain** $\Omega \subset \mathbb{R}^d$ occupied by the nuclear core reactor
- Neutrons are assumed to be separated into **2 groups** according to their energy: $E = \{E_1, E_2\}$ $(E_1 > E_2)$
- $\mu \in \mathcal{P}$: vector of **parameters** of the problem, which encodes the values of the **physical properties** of the nuclear core

Problem of interest: Find

- $u_\mu = (u_{1,\mu}, u_{2,\mu}) : \Omega \to \mathbb{R}^2$: **neutron scalar fluxes**;
- $\lambda_\mu > 0$ eigenvalue with **smallest modulus**;

solution to the non-symmetric eigenvalue problem

$$\mathcal{A}_\mu u_\mu = \lambda_\mu \mathcal{B}_\mu = \frac{1}{k_\mu} \mathcal{B}_\mu u_\mu$$

where $\mathcal{A}_\mu$ and $\mathcal{B}_\mu$ are linear operators such that $\mathcal{A}_\mu^{-1}\mathcal{B}_\mu$ satisfies the assumptions of the **Krein-Rutman theorem**.

Criticity calculation in a nuclear core reactor

Simple example: Two-group diffusion model

[Coste-Delclaux, Diop, Nicolas, Bonin, 2013], [Mula, 2014], [Giret, 2018], [Allaire, Blanc, Desprès, Golse, 2019]

- **Spatial domain** $\Omega \subset \mathbb{R}^d$ occupied by the nuclear core reactor
- Neutrons are assumed to be separated into **2 groups** according to their energy: $E = \{E_1, E_2\}$ $(E_1 > E_2)$
- $\mu \in \mathcal{P}$: vector of **parameters** of the problem, which encodes the values of the **physical properties** of the nuclear core

Problem of interest: Find

- $u_\mu = (u_{1,\mu}, u_{2,\mu}) : \Omega \to \mathbb{R}^2$: **neutron scalar fluxes**;
- $\lambda_\mu > 0$ eigenvalue with **smallest modulus**;

solution to the non-symmetric eigenvalue problem

$$\mathcal{A}_\mu u_\mu = \lambda_\mu \mathcal{B}_\mu = \frac{1}{k_\mu} \mathcal{B}_\mu u_\mu$$

where $\mathcal{A}_\mu$ and $\mathcal{B}_\mu$ are linear operators such that $\mathcal{A}_\mu^{-1}\mathcal{B}_\mu$ satisfies the assumptions of the **Krein-Rutman theorem**.

Criticity calculation in a nuclear core reactor

Simple example: Two-group diffusion model

[Coste-Delclaux, Diop, Nicolas, Bonin, 2013], [Mula, 2014], [Giret, 2018], [Allaire, Blanc, Desprès, Golse, 2019]

- **Spatial domain** $\Omega \subset \mathbb{R}^d$ occupied by the nuclear core reactor
- Neutrons are assumed to be separated into **2 groups** according to their energy: $E = \{E_1, E_2\}$ $(E_1 > E_2)$
- $\mu \in \mathcal{P}$: vector of **parameters** of the problem, which encodes the values of the **physical properties** of the nuclear core

Problem of interest: Find

- $u_\mu = (u_{1,\mu}, u_{2,\mu}) : \Omega \to \mathbb{R}^2$: **neutron scalar fluxes**;
- $\lambda_\mu > 0$ eigenvalue with **smallest modulus**;

solution to the non-symmetric eigenvalue problem

$$\mathcal{A}_\mu u_\mu = \lambda_\mu \mathcal{B}_\mu = \frac{1}{k_\mu} \mathcal{B}_\mu u_\mu$$

where $\mathcal{A}_\mu$ and $\mathcal{B}_\mu$ are linear operators such that $\mathcal{A}_\mu^{-1}\mathcal{B}_\mu$ satisfies the assumptions of the **Krein-Rutman theorem**.

Effective multiplication factor

- $k_\mu < 1$: the fission reaction is not the prevailing phenomenon, then the total mean number of neutrons tends towards zero along time;
  the reactor is said to be **subcritical**
- $k_\mu = 1$: both creation and absorption of neutrons take as much place as the other inside the system;
  the reactor is said to be **critical**
- $k_\mu > 1$: the fission dominates the absorption phenomenon, therefore a chain reaction phenomenon takes place inside the system, and the total mean number of neutrons increases at an exponential rate, the system then tends to collapse;
  the reactor is said to be **supercritical**

Two-Group Diffusion Equation

$$\mathcal{A}_\mu u_\mu = \lambda_\mu \mathcal{B}_\mu u_\mu$$

Two-group Diffusion Equation

$$\begin{aligned} &-\nabla \cdot (D_{1,\mu} \nabla u_{1,\mu}) + \Sigma_{11,\mu} u_{1,\mu} + \Sigma_{12,\mu} u_{2,\mu} \\ &= \lambda_\mu \left[\chi_{1,\mu} \left((\nu\Sigma_f)_{1,\mu} u_{1,\mu} + (\nu\Sigma_f)_{2,\mu} u_{2,\mu}\right)\right] \\ \\ &-\nabla . (D_{2,\mu} \nabla u_{2,\mu}) + \Sigma_{22,\mu} u_{2,\mu} + \Sigma_{21,\mu} u_{1,\mu} \\ &= \lambda_\mu \left[\chi_{2,\mu} \left((\nu\Sigma_f)_{1,\mu} u_{1,\mu} + (\nu\Sigma_f)_{2,\mu} u_{2,\mu}\right)\right] \end{aligned} \tag{1}$$

- $\Sigma_{ii} = \Sigma_{ti} - \Sigma_{s,ii}$;
- Σ_{ti}: total cross-section of group i;
- $\Sigma_{s,ij}$: scattering cross-section from group i to group j;
- $\Sigma_{ij} = -\Sigma_{s,ij}$;
- $D_i = \frac{1}{3\Sigma_{ti}}$: diffusion coefficient of group i;
- Σ_{fi}: fission cross-section of group i;
- ν_i: average number of neutrons of groupd i emitted per fission;
- χ_i: fission spectrum of group i

Parameters of the problem

- $\mu \in \mathcal{P}$ represents the **physical** properties of the core and its configuration.
- The spatial domain of calculation Ω is split into a structured grid that defines K regions. On each region Ω_k, μ^k represents the set of material parameters inside the domain Ω_k, so that $\mu = (\mu^1, \ldots, \mu^K) \in \mathcal{P}$.

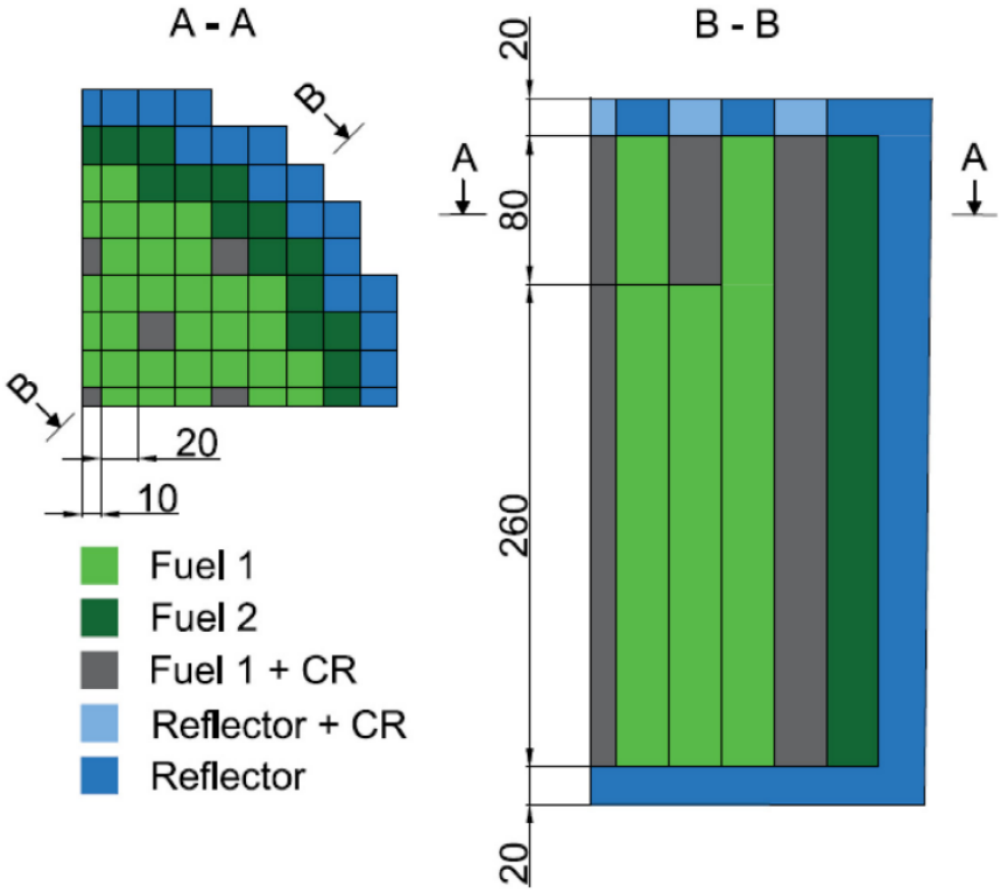


Figure: Cross-sectional view of the BSS-11 nuclear core reactor

Motivation of model-order reduction methods

- For a particular value of $\mu \in \mathcal{P}$, a numerical approximation of the solution u_μ is computed by some numerical scheme (for instance with a finite element code), the resolution of which may be very costly from a computational point of view.
- There exist a wide variety of contexts in which it is necessary to perform **parametric studies** of the problem at hand, i.e. to compute (a numerical approximation of) the solution u_μ for a very large number of values of the parameter vector μ as quickly as possible!

Examples:

- Design optimization
- Inverse problems
- Real-time control
- Uncertainty quantification

In such contexts, naive parametric studies using a standard finite element code may be extremely expensive from a computational point of view and time-consuming!

Motivation of model-order reduction methods

- For a particular value of $\mu \in \mathcal{P}$, a numerical approximation of the solution u_μ is computed by some numerical scheme (for instance with a finite element code), the resolution of which may be very costly from a computational point of view.
- There exist a wide variety of contexts in which it is necessary to perform **parametric studies** of the problem at hand, i.e. to compute (a numerical approximation of) the solution u_μ for a very large number of values of the parameter vector μ as quickly as possible!

Examples:

- Design optimization
- Inverse problems
- Real-time control
- Uncertainty quantification

In such contexts, naive parametric studies using a standard finite element code may be extremely expensive from a computational point of view and time-consuming!

Motivation of model-order reduction methods

- For a particular value of $\mu \in \mathcal{P}$, a numerical approximation of the solution u_μ is computed by some numerical scheme (for instance with a finite element code), the resolution of which may be very costly from a computational point of view.
- There exist a wide variety of contexts in which it is necessary to perform **parametric studies** of the problem at hand, i.e. to compute (a numerical approximation of) the solution u_μ for a very large number of values of the parameter vector μ as quickly as possible!

Examples:

- Design optimization
- Inverse problems
- Real-time control
- Uncertainty quantification

In such contexts, naive parametric studies using a standard finite element code may be extremely expensive from a computational point of view and time-consuming!

Principle of model-order reduction

Model-order reduction methods have been developped to circumvent this difficulty. The principle of these methods is the following:

- **Offline stage:** Compute u_μ with a standard numerical scheme (for instance finite elements) for a **small** number of well-chosen values of the parameter vector μ; this stage can be quite **expensive** from a computational point of view.
- Build another model, a **reduced model** from these few (expensive) computations in order to compute numerical approximations of u_μ for many other values of μ, but at a computational cost which is **much cheaper** than the initial (finite element) scheme.
- **Online stage:** Use the reduced model (instead of the original finite element code) in order to compute much faster u_μ for a large number of values of μ.

Principle of model-order reduction

Model-order reduction methods have been developped to circumvent this difficulty. The principle of these methods is the following:

- **Offline stage:** Compute u_μ with a standard numerical scheme (for instance finite elements) for a **small** number of well-chosen values of the parameter vector μ; this stage can be quite **expensive** from a computational point of view.
- Build another model, a **reduced model** from these few (expensive) computations in order to compute numerical approximations of u_μ for many other values of μ, but at a computational cost which is **much cheaper** than the initial (finite element) scheme.
- **Online stage:** Use the reduced model (instead of the original finite element code) in order to compute much faster u_μ for a large number of values of μ.

Principle of model-order reduction

Model-order reduction methods have been developped to circumvent this difficulty. The principle of these methods is the following:

- **Offline stage:** Compute u_μ with a standard numerical scheme (for instance finite elements) for a **small** number of well-chosen values of the parameter vector μ; this stage can be quite **expensive** from a computational point of view.
- Build another model, a **reduced model** from these few (expensive) computations in order to compute numerical approximations of u_μ for many other values of μ, but at a computational cost which is **much cheaper** than the initial (finite element) scheme.
- **Online stage:** Use the reduced model (instead of the original finite element code) in order to compute much faster u_μ for a large number of values of μ.

Principle of model-order reduction

Model-order reduction methods have been developped to circumvent this difficulty. The principle of these methods is the following:

- **Offline stage:** Compute u_μ with a standard numerical scheme (for instance finite elements) for a **small** number of well-chosen values of the parameter vector μ; this stage can be quite **expensive** from a computational point of view.
- Build another model, a **reduced model** from these few (expensive) computations in order to compute numerical approximations of u_μ for many other values of μ, but at a computational cost which is **much cheaper** than the initial (finite element) scheme.
- **Online stage:** Use the reduced model (instead of the original finite element code) in order to compute much faster u_μ for a large number of values of μ.

Reduced-basis method

A few seminal references:

- Cohen, Dahmen, DeVore, Maday, Patera...
- *Reduced Basis Methods for Partial Differential Equations: An Introduction*, Alfio Quarteroni, Andrea Manzoni, Federico Negri
- *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*, Jan S Hesthaven, Gianluigi Rozza, Benjamin Stamm

In this talk: **Reduced Basis method** for accelerating the resolution of parametrized **generalized non-symmetric eigenvalue problems**, with a view to accelerating parametric studies for criticity calculations.

Two-Group Diffusion Equation (discrete formulation)

- Discretization of the spatial domain Ω with $P1$ crossed-triangular **finite elements** over a rectangle mesh
- The solution u_μ is approximated by an element $u_{\mu,h}$ belonging to a finite-dimensional subspace V_h of dimension N_h (number of DoFs): $V_h = \mathrm{Span}\{(\varphi_i)_{i=1,N_h}\}$

Weak formulation of the problem

$$\text{Find } (u_{\mu,h}, u^*_{\mu,h}, k_{\mu,h}) \in V_h \times V_h \times \mathbb{R}^*_+ \text{ such that}$$

$$\forall v_h \in V_h, \quad a_{\mu,h}(u_{\mu,h}, v_h) = \frac{1}{k_{\mu,h}} b_{\mu,h}(u_{\mu,h}, v_h).$$

Adjoint problem $\quad \forall v_h \in V_h, \quad a_{\mu,h}(v_h, u^*_{\mu,h}) = \frac{1}{k_{\mu,h}} b_{\mu,h}(v_h, u^*_{\mu,h}).$

Two-Group Diffusion Equation (discrete formulation)

- Discretization of the spatial domain Ω with $P1$ crossed-triangular **finite elements** over a rectangle mesh
- The solution u_μ is approximated by an element $u_{\mu,h}$ belonging to a finite-dimensional subspace V_h of dimension N_h (number of DoFs): $V_h = \mathrm{Span}\{(\varphi_i)_{i=1,N_h}\}$

Weak formulation of the problem

$$\text{Find } \left(u_{\mu,h}, u^*_{\mu,h}, k_{\mu,h}\right) \in V_h \times V_h \times \mathbb{R}^*_+ \text{ such that}$$

$$\forall v_h \in V_h, \quad a_{\mu,h}(u_{\mu,h}, v_h) = \frac{1}{k_{\mu,h}} b_{\mu,h}(u_{\mu,h}, v_h).$$

Adjoint problem $\quad \forall v_h \in V_h, \quad a_{\mu,h}(v_h, u^*_{\mu,h}) = \frac{1}{k_{\mu,h}} b_{\mu,h}(v_h, u^*_{\mu,h}).$

Two-Group Diffusion Equation (matrix form)

$$u_{\mu,h} = \sum_{i=1}^{N_h} (U_{\mu,h})_i \varphi_i, \quad u^*_{\mu,h} = \sum_{i=1}^{N_h} (U^*_{\mu,h})_i \varphi_i \tag{2}$$

Matrix form of the problem

Find $\left(U_{\mu,h}, U^*_{\mu,h}, k_{\mu,h}\right) \in \mathbb{R}^{N_h} \times \mathbb{R}^{N_h} \times \mathbb{R}^*_+$ such that

$$A_{\mu,h} U_{\mu,h} = \frac{1}{k_{\mu,h}} B_{\mu,h} U_{\mu,h} \tag{3}$$

Adjoint problem

$$A^T_{\mu,h} U^*_{\mu,h} = \frac{1}{k_{\mu,h}} B^T_{\mu,h} U^*_{\mu,h}$$

- **Generalized eigenvalue problem**
- $A_{\mu,h} \in \mathbb{R}^{N_h \times N_h}$ is **non-symmetric** and invertible
- $B_{\mu,h} \in \mathbb{R}^{N_h \times N_h}$ is **non-symmetric**, not invertible and positive

$\longrightarrow$ **High-fidelity** problem

Reduced basis method

- The resolution of the high-fidelity problem for a large number of values of the parameter vector $\mu \in \mathcal{P}$ may be very costly from a computational point of view because N_h is large!
- The principle of the reduced basis method is to approximate the solution $(u_{\mu,h}, u^*_{\mu,h}, k_{\mu,h})$ by a Galerkin approximation associated to a linear subspace $V_N \subset V_h$ of dimension at most $2N$ with N much smaller than N_h.
- The **reduced space** V_N is chosen such that

$$V_N = \mathrm{Vect}\left\{u_{\mu_1,h}, u^*_{\mu_1,h}, \cdots, u_{\mu_N,h}, u^*_{\mu_N,h}\right\},$$

  where $\mu_1, \cdots, \mu_N$ are N particular well-chosen values of the parameter vector μ.
- In the **offline stage**, the high-fidelity problem is only solved for this N values of the parameter vector.

Reduced basis method

- The resolution of the high-fidelity problem for a large number of values of the parameter vector $\mu \in \mathcal{P}$ may be very costly from a computational point of view because N_h is large!
- The principle of the reduced basis method is to approximate the solution $\left(u_{\mu,h}, u^*_{\mu,h}, k_{\mu,h}\right)$ by a Galerkin approximation associated to a linear subspace $V_N \subset V_h$ of dimension at most $2N$ with N much smaller than N_h.
- The **reduced space** V_N is chosen such that

$$V_N = \text{Vect}\left\{u_{\mu_1,h}, u^*_{\mu_1,h}, \cdots, u_{\mu_N,h}, u^*_{\mu_N,h}\right\},$$

where $\mu_1, \cdots, \mu_N$ are N particular well-chosen values of the parameter vector μ.

- In the **offline stage**, the high-fidelity problem is only solved for this N values of the parameter vector.

Reduced basis method

- The resolution of the high-fidelity problem for a large number of values of the parameter vector $\mu \in \mathcal{P}$ may be very costly from a computational point of view because N_h is large!
- The principle of the reduced basis method is to approximate the solution $\left(u_{\mu,h}, u^*_{\mu,h}, k_{\mu,h}\right)$ by a Galerkin approximation associated to a linear subspace $V_N \subset V_h$ of dimension at most $2N$ with N much smaller than N_h.
- The **reduced space** V_N is chosen such that

$$V_N = \mathrm{Vect}\left\{u_{\mu_1,h}, u^*_{\mu_1,h}, \cdots, u_{\mu_N,h}, u^*_{\mu_N,h}\right\},$$

  where $\mu_1, \cdots, \mu_N$ are N particular well-chosen values of the parameter vector μ.
- In the **offline stage**, the high-fidelity problem is only solved for this N values of the parameter vector.

Reduced basis method

- The resolution of the high-fidelity problem for a large number of values of the parameter vector $\mu \in \mathcal{P}$ may be very costly from a computational point of view because N_h is large!
- The principle of the reduced basis method is to approximate the solution $\left(u_{\mu,h}, u^*_{\mu,h}, k_{\mu,h}\right)$ by a Galerkin approximation associated to a linear subspace $V_N \subset V_h$ of dimension at most $2N$ with N much smaller than N_h.
- The **reduced space** V_N is chosen such that

$$V_N = \text{Vect}\left\{u_{\mu_1,h}, u^*_{\mu_1,h}, \cdots, u_{\mu_N,h}, u^*_{\mu_N,h}\right\},$$

  where $\mu_1, \cdots, \mu_N$ are N particular well-chosen values of the parameter vector μ.
- In the **offline stage**, the high-fidelity problem is only solved for this N values of the parameter vector.

Online stage of the Reduced Basis method

Galerkin approximation of the eigenvalue problem in V_N

Weak formulation of the reduced problem

$$\text{Find } (u_{\mu,N}, u^*_{\mu,N}, k_{\mu,N}) \in V_N \times V_N \times \mathbb{R}^*_+ \text{ such that}$$

$$\forall v_N \in V_N, \quad a_{\mu,h}(u_{\mu,N}, v_N) = \frac{1}{k_{\mu,N}} b_{\mu,h}(u_{\mu,N}, v_N).$$

Adjoint problem $\quad \forall v_N \in V_N, \quad a_{\mu,h}(v_N, u^*_{\mu,N}) = \frac{1}{k_{\mu,N}} b_{\mu,h}(v_N, u^*_{\mu,N}).$

Online stage of the reduced basis method

- In the **online stage**, for each new value of $\mu \in \mathcal{P}$, an atmost $2N$-dimensional matrix eigenvalue problem is solved. When $N \ll N_h$, the resolution of the reduced problem is much cheaper from a computational point of view than the resolution of the original high-fidelity problem!
- **Reduced basis**: Let $n := \dim V_N$ and $(\theta_1, \cdots, \theta_n)$ be an orthonormal basis of V_N. Denoting by

$$\Theta_N := (\theta_1 | \cdots | \theta_n) \in \mathbb{R}^{N_h \times n},$$

We define the $n \times n$ reduced matrices:

$$\begin{cases} A_{\mu,N} = \Theta_N^T A_{\mu,h} \Theta_N \\ B_{\mu,N} = \Theta_N^T B_{\mu,h} \Theta_N \end{cases}.$$

Approximate solutions given by the reduced basis method

Reduced problem

Find $\left(c_{\mu,N}, c^*_{\mu,N}, k_{\mu,N}\right) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^*_+$ such that

$$A_{\mu,N} c_{\mu,N} = \frac{1}{k_{\mu,N}} B_{\mu,N} c_{\mu,N} \quad \text{and} \quad U_{\mu,N} = \Theta_N c_{\mu,N}$$

$$A^T_{\mu,N} c^*_{\mu,N} = \frac{1}{k_{\mu,N}} B^T_{\mu,N} c^*_{\mu,N} \quad \text{and} \quad U^*_{\mu,N} = \Theta_N c^*_{\mu,N}$$

$$u_{\mu,N} := \sum_{i=1}^{N_h} \left(U_{\mu,N}\right)_i \varphi_i, \quad u^*_{\mu,N} := \sum_{i=1}^{N_h} \left(U^*_{\mu,N}\right)_i \varphi_i.$$

How to build V_N ?

This is usually done via an iterative algorithm called a **greedy algorithm**.

Need to choose a finite subset $\mathcal{P}_{\text{train}} \subset \mathcal{P}$, called **training set**.

Naive Greedy algorithm

- Choose randomly $\mu_1 \in \mathcal{P}_{\text{train}}$.

$$V_1 = \text{Vect}\left\{u_{\mu_1,h}, u^*_{\mu_1,h}\right\}$$

- **Iteration** N**:** Choose $\mu_N \in \mathcal{P}_{\text{train}}$ such that

$$\mu_N \in \underset{\mu \in \mathcal{P}_{\text{train}}}{\text{argmax}} \, |k_{\mu,h} - k_{\mu,N-1}|$$

$$V_N = \text{Vect}\left\{u_{\mu_1,h}, u^*_{\mu_1,h}, \cdots, u_{\mu_N,h}, u^*_{\mu_N,h}\right\}$$

A naive version of the Greedy algorithm requires to evaluate $k_{\mu,h}$, for all $\mu \in \Lambda_{\text{train}}$
$\rightarrow$ too expensive...

Practical algorithm:
Replace $e^k_{N-1}(\mu) := |k_{\mu,h} - k_{\mu,N-1}|$ by an easy-to-compute **a posteriori error estimator** $\Delta^k_{N-1}(\mu)$.

Error on the eigenvalue

- **Residuals**:

$$R_{\mu,N} = (B_{\mu,h} - k_{\mu,N} A_{\mu,h})\, u_{\mu,N}$$
$$R^*_{\mu,N} = \left(B^T_{\mu,h} - k_{\mu,N} A^T_{\mu,h}\right) u^*_{\mu,N} \tag{4}$$

Proposition. *A posteriori* error estimator

There exists a positive constant $C^k(\mu) > 0$ (called the **prefactor**) such that for all $\mu \in \mathcal{P}$,

$$e^k_N(\mu) = |k_{\mu,h} - k_{\mu,N}| \leqslant C^k(\mu) \frac{\|R_{\mu,N}\| \|R^*_{\mu,N}\|}{\langle c^*_{\mu,N}, A_{\mu,N} c_{\mu,N} \rangle} = C^k(\mu) \eta^k_N(\mu) \tag{5}$$

with $\eta^k_N(\mu) := \dfrac{\|R_{\mu,N}\| \|R^*_{\mu,N}\|}{\langle c^*_{\mu,N}, A_{\mu,N} c_{\mu,N} \rangle}$.

How to build V_N ?

- **Practical *a posteriori* error estimator**:

$$\Delta_N^k(\mu) = \overline{C}_N^k \frac{\|R_{\mu,N}\| \|R^*_{\mu,N}\|}{\langle c^*_{\mu,N}, A_{\mu,N} c_{\mu,N} \rangle} = \overline{C}_N^k \eta_N^k(\mu)$$

  where $\overline{C}_N^k$ is a heuristic estimation of the prefactor $C^k(\mu)$
- $\Delta_N^k(\mu)$ can be efficiently computed with complexity $\mathcal{O}(n^2)$ if the data of the problem is separated.

How to build V_N ?

Actual Greedy algorithm

- Choose randomly $\mu_1 \in \mathcal{P}_{\text{train}}$.

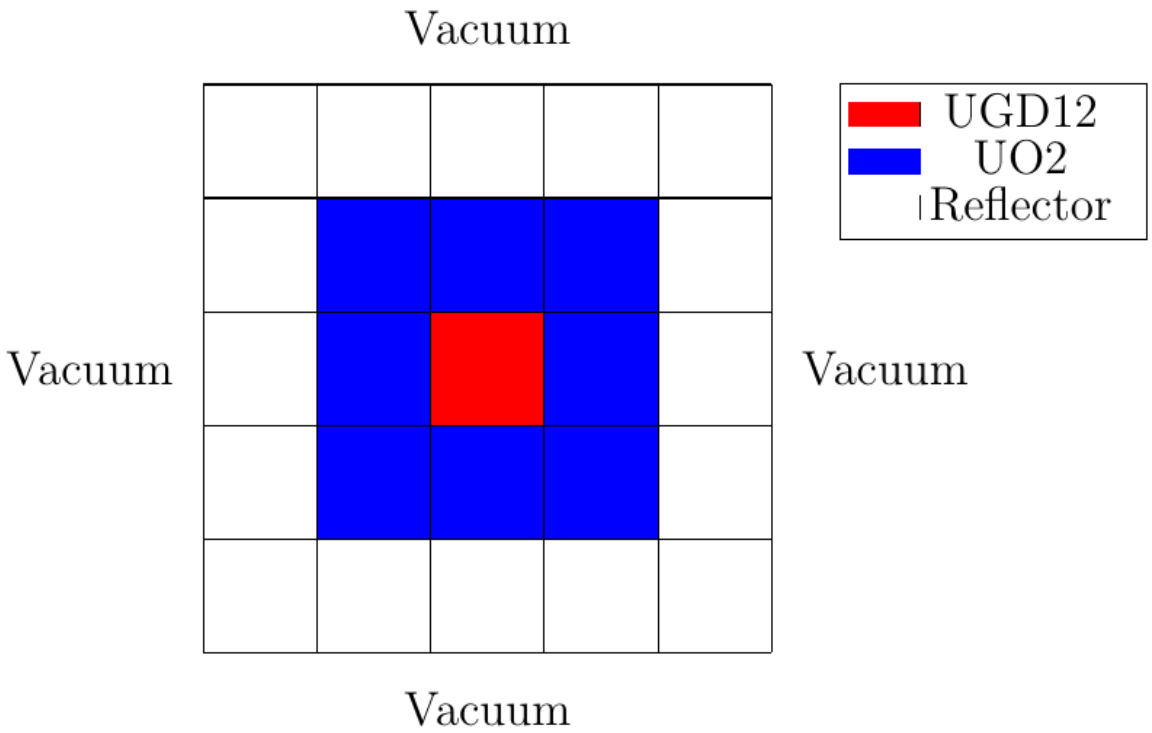
$$V_1 = \text{Vect}\left\{u_{\mu_1,h}, u^*_{\mu_1,h}\right\}$$

- **Iteration** N**:** Choose $\mu_N \in \mathcal{P}_{\text{train}}$ such that

$$\mu_N \in \underset{\mu \in \mathcal{P}_{\text{train}}}{\text{argmax}}\ \Delta^k_{N-1}(\mu).$$

$$V_N = \text{Vect}\left\{u_{\mu_1,h}, u^*_{\mu_1,h}, \cdots, u_{\mu_N,h}, u^*_{\mu_N,h}\right\}$$

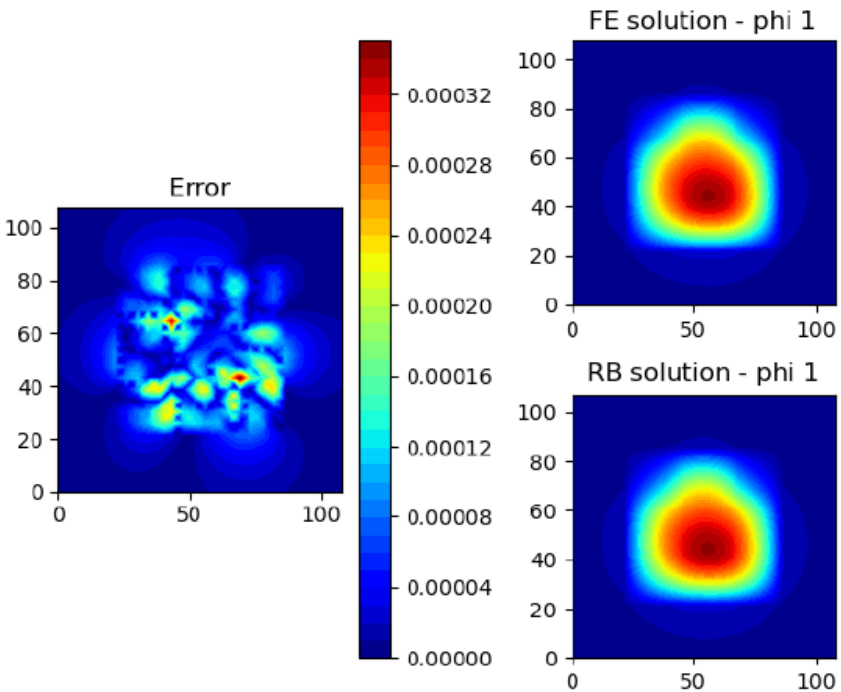
First toy test case: the MiniCore problem

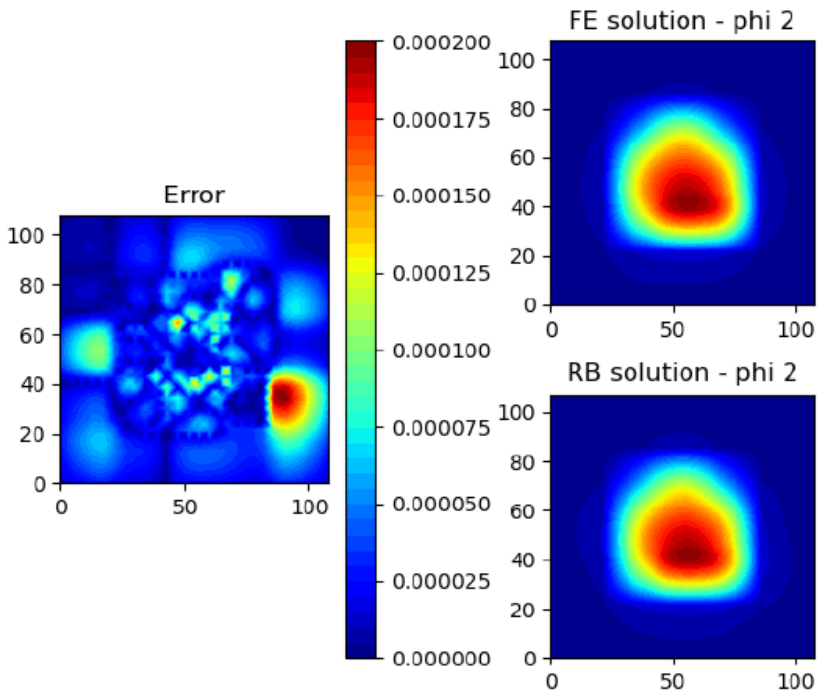


- 25 spatial regions
- $L = 107.52\ cm$
- UGD12: mix of uranium dioxyde and Galinium oxyde
- UO2: uranium dioxyde
- BC: $u_\mu(x) = 0, \quad x \in \partial\Omega$
- $N_h = 2602$ DoFs per group
- Training set of parameters $\mathcal{P}_{\text{train}}$ of cardinality 1000 generated **randomly**

High-fidelity and reduced solver

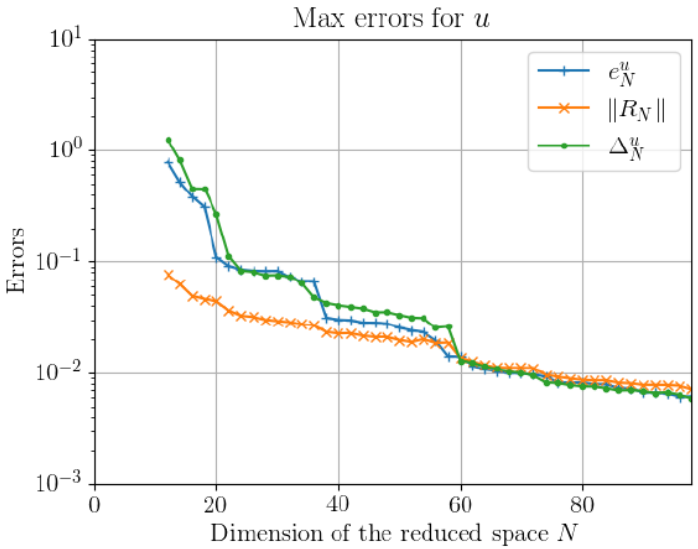
Reduced-order model obtained with $N = 100$

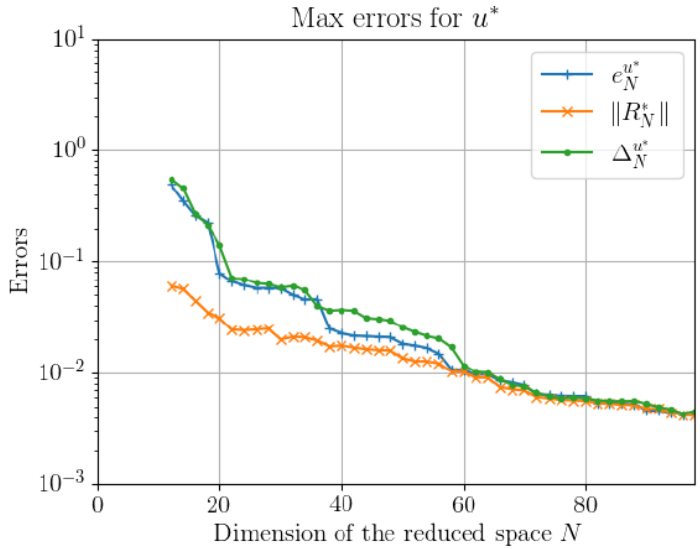


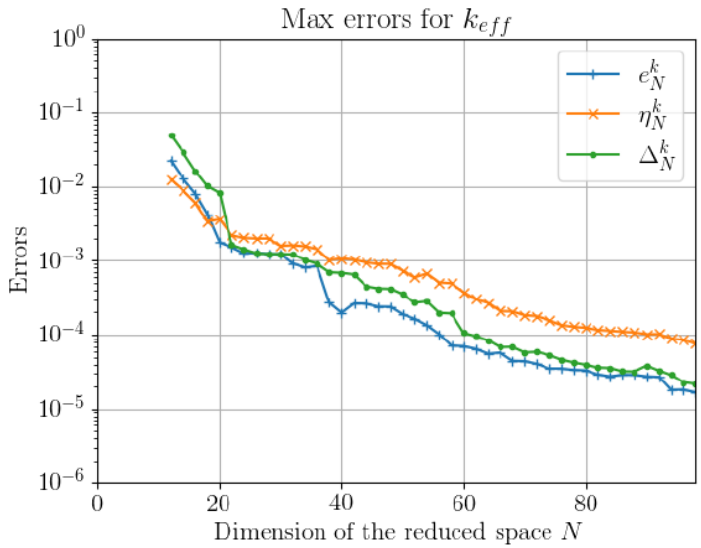


Convergence of the reduced basis : mean relative errors over $\mathcal{P}_{\text{test}}$

- $\mathcal{P}_{\text{test}} \subset \mathcal{P}$ with cardinality 50 (test set)
- $\mathcal{P}_{\text{pref}} \subset \mathcal{P}$ with cardinality 10 (prefactor set)







Parametric variability of the prefactor

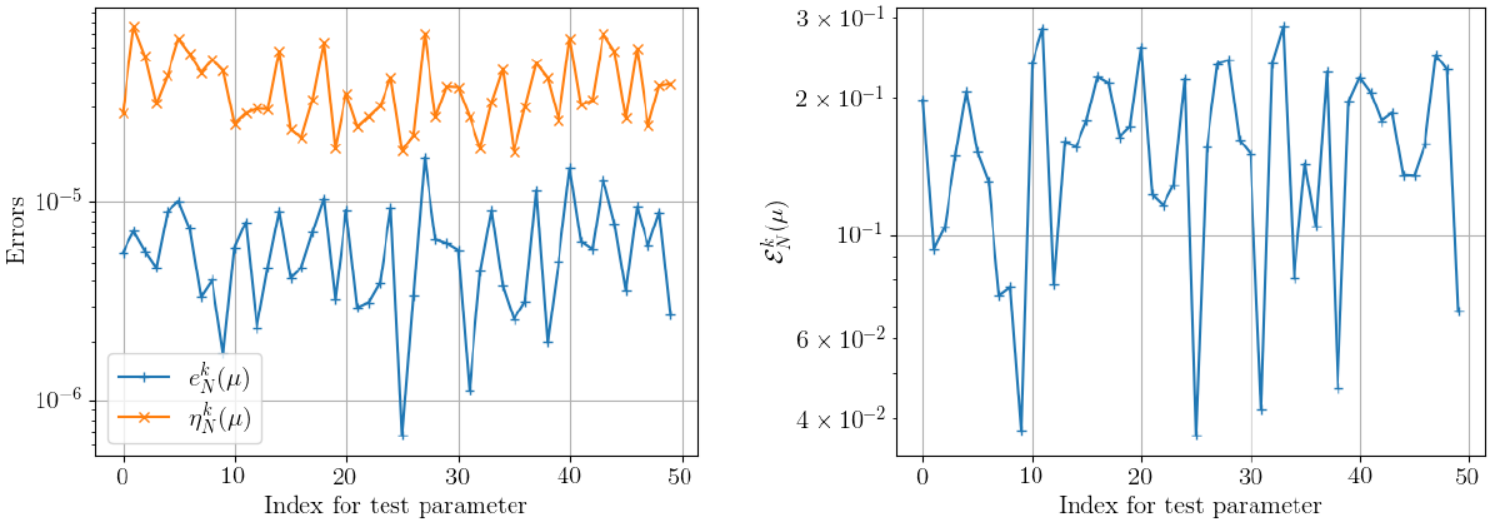


Figure: Parametric variability of the prefactor

Gain in computational time

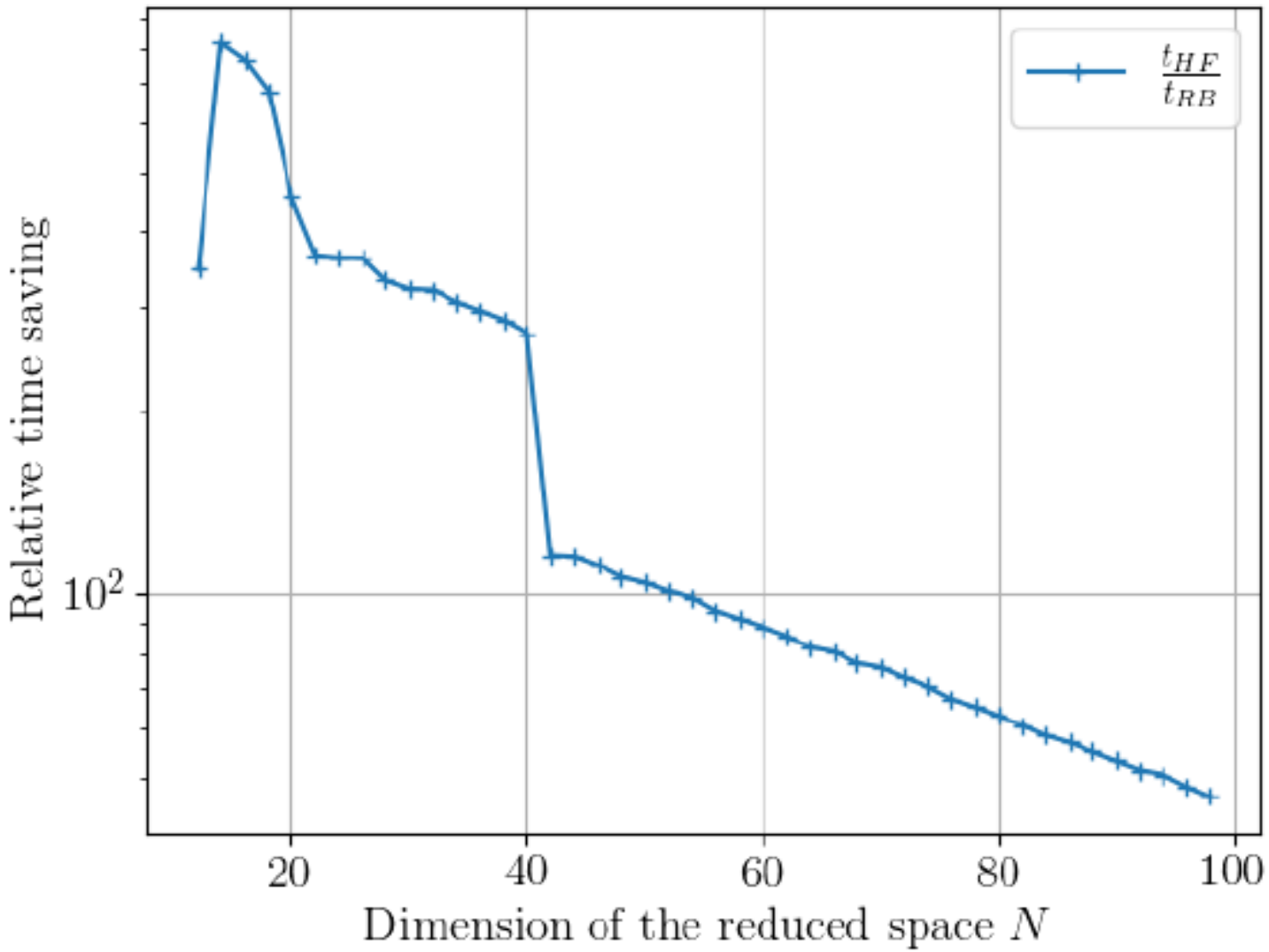


Figure: Relative time saving of the reduced solver

3D test case in APOLLO3 code (MINARET solver)

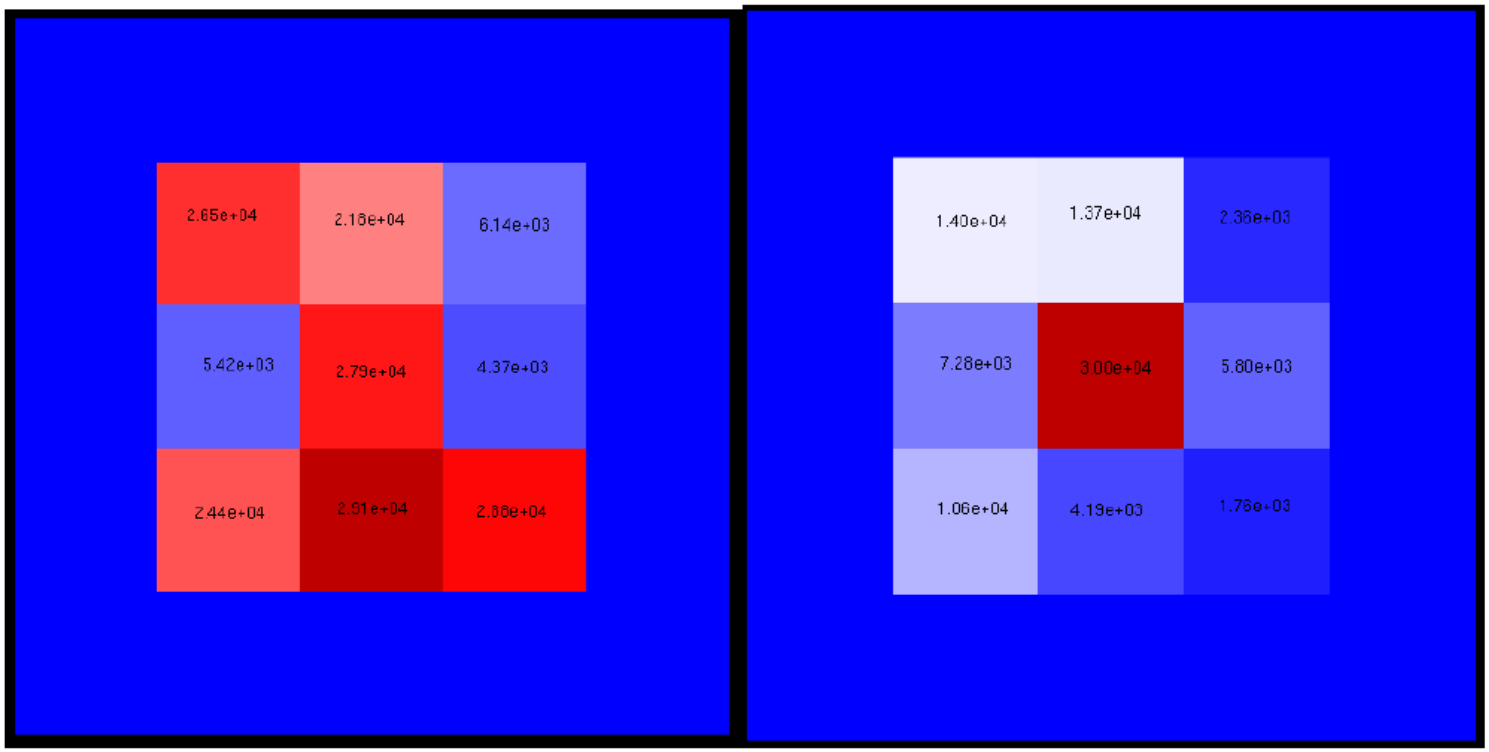
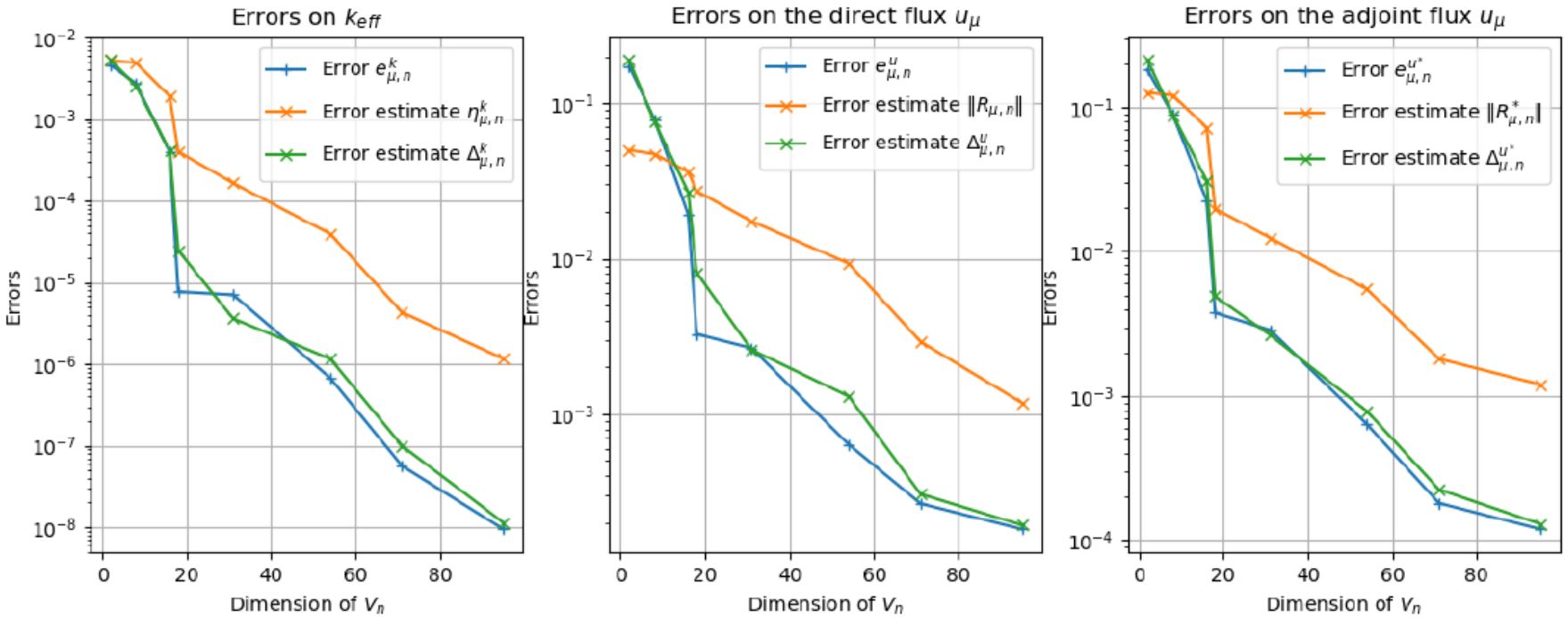


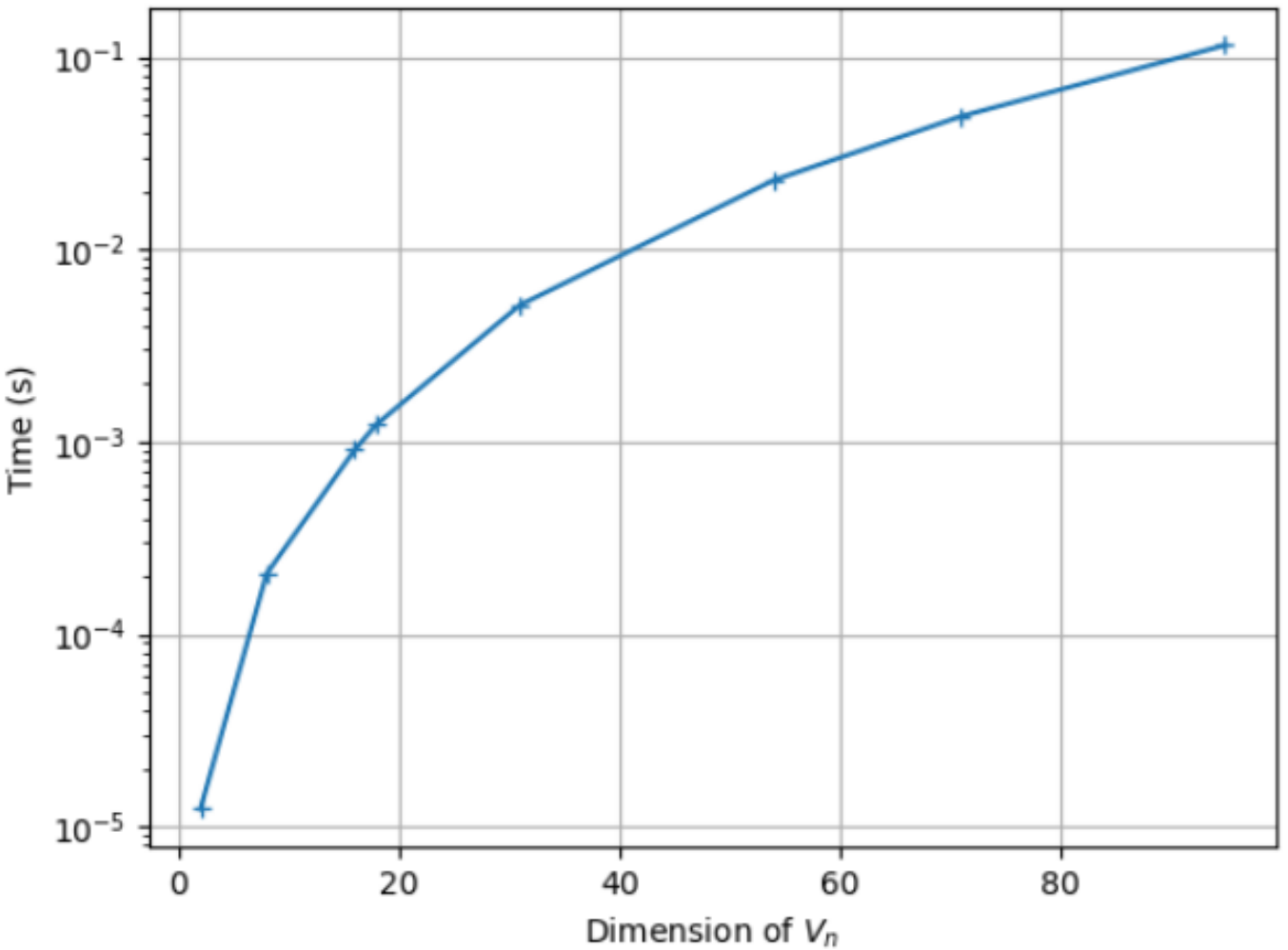
Figure: Cross-sectional views of the 3D core

- 9 spatial regions
- BC: $u_\mu(x) = 0, \quad x \in \partial\Omega$
- $N_h = 108800$ DoFs per group
- Training set of parameters $\mathcal{P}_{\text{train}}$ of cardinality 100 generated **randomly**
- Prefactor set $\mathcal{P}_{\text{pref}}$ of cardinality 5
- Test set $\mathcal{P}_{\text{test}}$ of cardinality 10

Convergence of the reduced basis approximation



Computational runtime of the reduced-order model



What else?

Conclusions:

- Example of linear approximation method dedicated to one specific application in a high-dimensional context: efficient reduced-order model for criticity calculations in neutronics using the reduced basis method
- Very encouraging results obtained on two-group diffusion models with the APOLLO3 code

To go beyond:

- What if the linear approximation spaces built by the greedy algorithms had not yielded accurate enough approximations?
- Current trend: combine linear and **nonlinear** approximation approaches.
- What if we are not in a parametric setting?

Outline of the talk

Nonlinear approximation

Goal: find an approximation of a **high-dimensional function**

$$u(x_1, \ldots, x_d)$$

and assume that u belongs to some Hilbert space V.

For a certain subset of functions $Z \subset V$ described by a small number n parameters, find a **best approximation** z^* of u by elements of Z is defined by

$$z^* = \inf_{z \in Z} \|u - z\|_V$$

The set Z is **not** a linear space in general.

Nonlinear approximation

Goal: find an approximation of a **high-dimensional function**

$$u(x_1, \ldots, x_d)$$

and assume that u belongs to some Hilbert space V.

For a certain subset of functions $Z \subset V$ described by a small number n parameters, find a **best approximation** z^* of u by elements of Z is defined by

$$z^* = \inf_{z \in Z} \|u - z\|_V$$

The set Z is **not** a linear space in general.

Nonlinear approximation

Goal: find an approximation of a **high-dimensional function**

$$u(x_1, \ldots, x_d)$$

and assume that u belongs to some Hilbert space V.

For a certain subset of functions $Z \subset V$ described by a small number n parameters, find a **best approximation** z^* of u by elements of Z is defined by

$$z^* = \inf_{z \in Z} \|u - z\|_V$$

The set Z is **not** a linear space in general.

Typical examples of subsets Z

- **Low-rank tensors:**

$$u(x_1, \ldots, x_d) \approx \sum_{k=1}^{R} r_k^{(1)}(x_1) r_k^{(2)}(x_2) \ldots r_k^{(d)}(x_d)$$

- **Neural networks:**

$$u(x_1, \ldots, x_d) = u(x) \approx \sigma\left(A_1 \sigma(A_2(\ldots \sigma(A_L x + b_L)\ldots) + b_2) + b_1\right)$$

where for all $1 \leq i \leq L$, A_i are matrices, b_i vectors and $\sigma : \mathbb{R} \to \mathbb{R}$ is called the activation function

- ...

Tensor methods

For $r^1, \ldots, r^d$ univariate functions,

$$r^1 \otimes \cdots \otimes r^d(x_1, \cdots, x_d) = r^1(x_1) \cdots r^d(x_d)$$

and assume that $r^1 \otimes \cdots \otimes r^d$ belongs to V.

The function $r^1 \otimes \cdots \otimes r^d$ is then called a pure tensor product function.

Tensor methods are one family of approximation tools used for the resolution of high-dimensional PDEs. The solution $u \in V$ of a high-dimensional PDE is approximated as some linear combination of pure tensor product functions.

Classical tensor methods consist in approximating u in a certain tensor format, i.e. by a function which belongs to some subset Z of V, the elements of which can be characterized as particular linear combinations of pure tensor product functions with low complexity.

Tensor methods

For $r^1, \ldots, r^d$ univariate functions,

$$r^1 \otimes \cdots \otimes r^d(x_1, \cdots, x_d) = r^1(x_1) \cdots r^d(x_d)$$

and assume that $r^1 \otimes \cdots \otimes r^d$ belongs to V.

The function $r^1 \otimes \cdots \otimes r^d$ is then called a pure tensor product function.

Tensor methods are one family of approximation tools used for the resolution of high-dimensional PDEs. The solution $u \in V$ of a high-dimensional PDE is approximated as some linear combination of pure tensor product functions.

Classical tensor methods consist in approximating u in a certain tensor format, i.e. by a function which belongs to some subset Z of V, the elements of which can be characterized as particular linear combinations of pure tensor product functions with low complexity.

Tensor methods

For $r^1, \ldots, r^d$ univariate functions,

$$r^1 \otimes \cdots \otimes r^d(x_1, \cdots, x_d) = r^1(x_1) \cdots r^d(x_d)$$

and assume that $r^1 \otimes \cdots \otimes r^d$ belongs to V.

The function $r^1 \otimes \cdots \otimes r^d$ is then called a pure tensor product function.

Tensor methods are one family of approximation tools used for the resolution of high-dimensional PDEs. The solution $u \in V$ of a high-dimensional PDE is approximated as some linear combination of pure tensor product functions.

Classical tensor methods consist in approximating u in a certain tensor format, i.e. by a function which belongs to some subset Z of V, the elements of which can be characterized as particular linear combinations of pure tensor product functions with low complexity.

Tensor formats

(Grasedyck, Khoromskij, Kolda, Hackbusch, Lubich, Oseledets, ...)

- **Canonical polyadic format of rank lower than $R \in \mathbb{N}^*$:**

$$Z_R^{\text{can}} := \left\{ z = \sum_{k=1}^{R} r_k^1 \otimes \cdots \otimes r_k^d \right\}. \tag{6}$$

$$\boxed{COMP = \mathcal{O}(RNd)}$$

- **Tucker format with rank $\boldsymbol{R} := (R, \cdots, R)$ with $R \in \mathbb{N}^*$:**

$$Z_{\boldsymbol{R}}^{\text{Tucker}} := \left\{ \begin{array}{c} z = \sum_{k_1=1}^{R} \cdots \sum_{k_d=1}^{R} c_{k_1,\ldots,k_d} r_{k_1}^1 \otimes \cdots \otimes r_{k_d}^d, \\ \left(c_{k_1,\ldots,k_d}\right)_{1 \le k_1 \le R, \ldots, 1 \le k_d \le R} \in \mathbb{R}^{R \times \cdots \times R} \end{array} \right\}. \tag{7}$$

$$\boxed{COMP = \mathcal{O}(R^d + NRd)}$$

- **Tensor Train format with rank $\boldsymbol{R} := (R, R, \cdots, R)$ with $R \in \mathbb{N}^*$:**

$$Z_{\boldsymbol{R}}^{\text{TT}} := \left\{ \begin{array}{c} z(x_1, \ldots, x_d) = S_1(x_1)^T M_2(x_2) \cdots M_{d-1}(x_{d-1}) S_d(x_d), \\ S_1(x_1) \in \mathbb{R}^R, \ S_d(x_d) \in \mathbb{R}^R, M_i(x_i) \in \mathbb{R}^{R \times R}, \ \forall 2 \le i \le d-1 \end{array} \right\}. \tag{8}$$

$$\boxed{COMP = \mathcal{O}(R^2 Nd)}$$

Tensor formats

(Grasedyck, Khoromskij, Kolda, Hackbusch, Lubich, Oseledets, ...)

- **Canonical polyadic format of rank lower than $R \in \mathbb{N}^*$:**

$$Z_R^{\text{can}} := \left\{ z = \sum_{k=1}^{R} r_k^1 \otimes \cdots \otimes r_k^d \right\}. \tag{6}$$

$$\boxed{COMP = \mathcal{O}(RNd)}$$

- **Tucker format with rank $\boldsymbol{R} := (R, \cdots, R)$ with $R \in \mathbb{N}^*$:**

$$Z_{\boldsymbol{R}}^{\text{Tucker}} := \left\{ \begin{array}{c} z = \sum_{k_1=1}^{R} \cdots \sum_{k_d=1}^{R} c_{k_1,\ldots,k_d} r_{k_1}^1 \otimes \cdots \otimes r_{k_d}^d, \\ \left(c_{k_1,\ldots,k_d}\right)_{1 \le k_1 \le R, \ldots, 1 \le k_d \le R} \in \mathbb{R}^{R \times \cdots \times R} \end{array} \right\}. \tag{7}$$

$$\boxed{COMP = \mathcal{O}(R^d + NRd)}$$

- **Tensor Train format with rank $\boldsymbol{R} := (R, R, \cdots, R)$ with $R \in \mathbb{N}^*$:**

$$Z_{\boldsymbol{R}}^{\text{TT}} := \left\{ \begin{array}{c} z(x_1, \ldots, x_d) = S_1(x_1)^T M_2(x_2) \cdots M_{d-1}(x_{d-1}) S_d(x_d), \\ S_1(x_1) \in \mathbb{R}^R, \ S_d(x_d) \in \mathbb{R}^R, M_i(x_i) \in \mathbb{R}^{R \times R}, \ \forall 2 \le i \le d-1 \end{array} \right\}. \tag{8}$$

$$\boxed{COMP = \mathcal{O}(R^2 Nd)}$$

Tensor formats

(Grasedyck, Khoromskij, Kolda, Hackbusch, Lubich, Oseledets, ...)

- **Canonical polyadic format of rank lower than $R \in \mathbb{N}^*$:**

$$Z_R^{\text{can}} := \left\{ z = \sum_{k=1}^{R} r_k^1 \otimes \cdots \otimes r_k^d \right\}. \tag{6}$$

$$\boxed{COMP = \mathcal{O}(RNd)}$$

- **Tucker format with rank $\boldsymbol{R} := (R, \cdots, R)$ with $R \in \mathbb{N}^*$:**

$$Z_{\boldsymbol{R}}^{\text{Tucker}} := \left\{ \begin{array}{c} z = \sum_{k_1=1}^{R} \cdots \sum_{k_d=1}^{R} c_{k_1,\ldots,k_d} r_{k_1}^1 \otimes \cdots \otimes r_{k_d}^d, \\ \left(c_{k_1,\ldots,k_d}\right)_{1 \le k_1 \le R, \ldots, 1 \le k_d \le R} \in \mathbb{R}^{R \times \cdots \times R} \end{array} \right\}. \tag{7}$$

$$\boxed{COMP = \mathcal{O}(R^d + NRd)}$$

- **Tensor Train format with rank $\boldsymbol{R} := (R, R, \cdots, R)$ with $R \in \mathbb{N}^*$:**

$$Z_{\boldsymbol{R}}^{\text{TT}} := \left\{ \begin{array}{c} z(x_1, \ldots, x_d) = S_1(x_1)^T M_2(x_2) \cdots M_{d-1}(x_{d-1}) S_d(x_d), \\ S_1(x_1) \in \mathbb{R}^R, \ S_d(x_d) \in \mathbb{R}^R, M_i(x_i) \in \mathbb{R}^{R \times R}, \ \forall 2 \le i \le d-1 \end{array} \right\}. \tag{8}$$

$$\boxed{COMP = \mathcal{O}(R^2 Nd)}$$

Greedy algorithms

Greedy algorithms are **iterative algorithms** used in nonlinear approximation theory. ([Temlyakov, 2008], Cohen, Dahmen, DeVore, Le Bris, Lelièvre, Maday...)

After n iterations of a greedy algorithm, an element $u \in V$ is approximated as the sum of n elements belonging to a subset $Z \subset V$, called a dictionary of V.

More precisely, at the n^{th} iteration of the greedy algorithm,

$$u \approx u_n = u_{n-1} + z_n$$

for some *best* element $z_n \in Z$, the definition of which depends on the problem u is solution to.

In computational mechanics, the **Progressive Generalized Decomposition** (PGD) method is a particular type of greedy algorithm used for the resolution of high-dimensional PDEs, which has been used in a wide variety of contexts (Ladevèze, Chinesta, Nouy, Néron, Chamoin...)

Greedy algorithms

Greedy algorithms are **iterative algorithms** used in nonlinear approximation theory. ([Temlyakov, 2008], Cohen, Dahmen, DeVore, Le Bris, Lelièvre, Maday...)

After n iterations of a greedy algorithm, an element $u \in V$ is approximated as the sum of n elements belonging to a subset $Z \subset V$, called a dictionary of V.

More precisely, at the n^{th} iteration of the greedy algorithm,

$$u \approx u_n = u_{n-1} + z_n$$

for some *best* element $z_n \in Z$, the definition of which depends on the problem u is solution to.

In computational mechanics, the **Progressive Generalized Decomposition** (PGD) method is a particular type of greedy algorithm used for the resolution of high-dimensional PDEs, which has been used in a wide variety of contexts (Ladevèze, Chinesta, Nouy, Néron, Chamoin...)

Greedy algorithms

Greedy algorithms are **iterative algorithms** used in nonlinear approximation theory. ([Temlyakov, 2008], Cohen, Dahmen, DeVore, Le Bris, Lelièvre, Maday...)

After n iterations of a greedy algorithm, an element $u \in V$ is approximated as the sum of n elements belonging to a subset $Z \subset V$, called a dictionary of V.

More precisely, at the n^{th} iteration of the greedy algorithm,

$$u \approx u_n = u_{n-1} + z_n$$

for some *best* element $z_n \in Z$, the definition of which depends on the problem u is solution to.

In computational mechanics, the **Progressive Generalized Decomposition** (PGD) method is a particular type of greedy algorithm used for the resolution of high-dimensional PDEs, which has been used in a wide variety of contexts (Ladevèze, Chinesta, Nouy, Néron, Chamoin...)

Dictionary

Assume V is a Hilbert space.

Definition

A set $Z \subset V$ is called a dictionary of V if and only if it satisfies the three following conditions:

(D1) The set $\mathrm{Span}\ Z$ is dense in V.

(D2) For all $\lambda \in \mathbb{R}$ and $z \in Z$, $\lambda z \in Z$.

(D3) Z is weakly closed in V.

Example: convex minimization

Let $\mathcal{E} : V \to \mathbb{R}$ be a strongly convex differentiable functional so that $\nabla\mathcal{E}$ is Lipschitz on bounded sets.

$$\boxed{u = \operatorname*{argmin}_{v \in V} \mathcal{E}(v)}$$

Pure Greedy algorithm:

1. set $u_0 = 0$ and $n = 1$;
2. find $z_n \in Z$ such that
$$z_n \in \operatorname*{argmin}_{z \in \Sigma} \mathcal{E}\left(u_{n-1} + z\right). \tag{9}$$
3. set $u_n = u_{n-1} + z_n$ and $n = n + 1$. Return to step 2.

Theorem

The iterations of the Pure Greedy algorithm are well-defined (i.e. there exists at least one minimizer $z_n \in Z$ to (11) for all $n \in \mathbb{N}^*$ and z_n is non-zero if and only if $u_{n-1} \neq u$). Moreover, the sequence $(u_n)_{n \in \mathbb{N}^*}$ strongly converges in V towards u.

Example: convex minimization

Let $\mathcal{E} : V \to \mathbb{R}$ be a strongly convex differentiable functional so that $\nabla\mathcal{E}$ is Lipschitz on bounded sets.

$$\boxed{u = \operatorname*{argmin}_{v \in V} \mathcal{E}(v)}$$

Orthogonal Greedy algorithm:

1. set $u_0 = 0$ and $n = 1$;
2. find $z_n \in Z$ such that
$$z_n \in \operatorname*{argmin}_{z \in \Sigma} \mathcal{E}\left(u_{n-1} + z\right). \tag{10}$$
3. set
$$u_n = \operatorname*{argmin}_{v \in \mathrm{Span}\{z_1, \ldots, z_n\}} \mathcal{E}\left(v\right). \tag{11}$$
and $n = n + 1$. Return to step 2.

Galerkin method in the linear space spanned by the elements $z_1, \ldots, z_n$

A posteriori error estimators!

Theoretical convergence results on greedy algorithms

Temlyakov, Lelièvre, Le Bris, Maday, Cancès, Falco, Nouy, Ehrlacher...

1. **Convex minimization problems:**

$$u = \underset{v \in V}{\operatorname{argmin}} \mathcal{E}(v).$$

2. **Linear bounded from below symmetric eigenvalue problems:**

$$Au = \lambda u.$$

3. **Non-symmetric linear problems:**

$$\forall v \in V, \quad a(u, v) = b(v).$$

4. **Parabolic evolution problems:**

$$\partial_t u + Au = f.$$

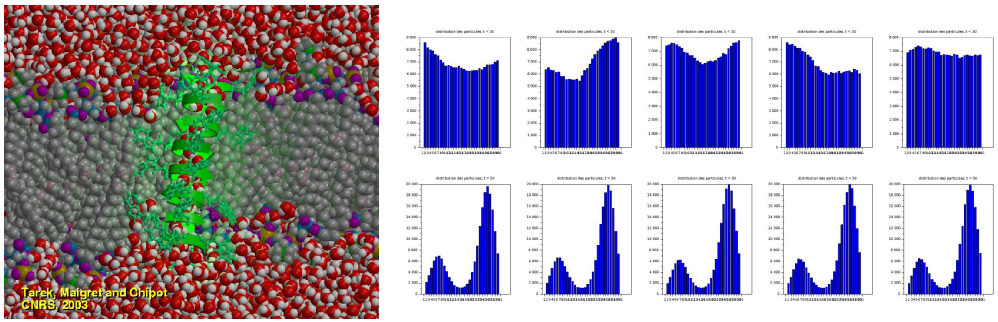
5. **Schrödinger evolution problems:**

$$i\partial_t u + Hu = f.$$

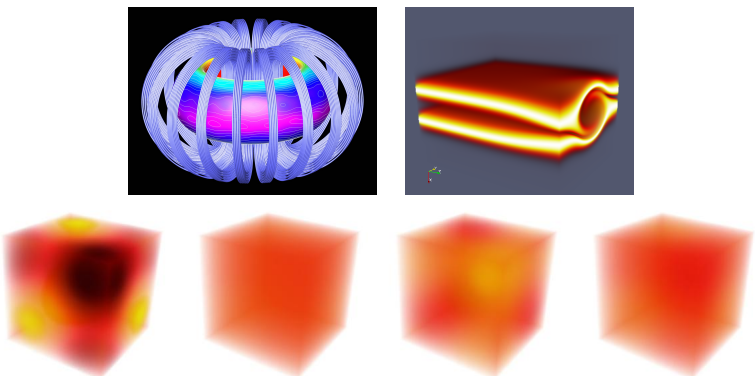
Applications of greedy algorithms in materials science

Lelièvre, Monmarché, Dabaghi, Strössner, Lombardi, Grigori, Song, Ruiz, Dupuy, Guillot..

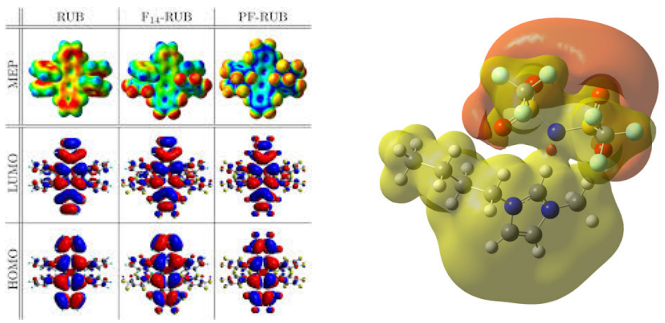
- Molecular dynamics



- Kinetic equations



- Electronic structure calculations



Perspectives

- Very recently, new numerical schemes for the resolution of high-dimensional PDEs, called **Galerkin neural networks**, have been introduced in

  [Ainsworth, Dong, 2022], [Siegel, Hong, Jin, Hao, Xu,2023]

  These are **greedy algorithms** associated to a dictionary Z defined by means of neural networks.

- **Current trend in numerical methods for high-dimensional PDEs:**

  Maday, Farhat, Somacal, Cohen...

combine methods and approaches from both the linear and nonlinear approximation world

Thank you for your attention!

Perspectives

- Very recently, new numerical schemes for the resolution of high-dimensional PDEs, called **Galerkin neural networks**, have been introduced in

  [Ainsworth, Dong, 2022], [Siegel, Hong, Jin, Hao, Xu,2023]

  These are **greedy algorithms** associated to a dictionary Z defined by means of neural networks.

- **Current trend in numerical methods for high-dimensional PDEs:**

  Maday, Farhat, Somacal, Cohen...

combine methods and approaches from both the linear and nonlinear approximation world

Thank you for your attention!

Perspectives

- Very recently, new numerical schemes for the resolution of high-dimensional PDEs, called **Galerkin neural networks**, have been introduced in

  [Ainsworth, Dong, 2022], [Siegel, Hong, Jin, Hao, Xu,2023]

  These are **greedy algorithms** associated to a dictionary Z defined by means of neural networks.

- **Current trend in numerical methods for high-dimensional PDEs:**

  Maday, Farhat, Somacal, Cohen...

combine methods and approaches from both the linear and nonlinear approximation world

Thank you for your attention!

Many thanks to all my collaborators!