ROM and Data Driven ROM

Christophe Prud'homme
prudhomme@unistra.fr

September 7, 2024

CeMosis - http://www.cemosis.fr
IRMA
Université de Strasbourg

Course Syllabus

Date & Time

- Lecture & Seminar
  - Thursday 13:30-15:30,
  - Start: 5/09/2024 (total 14 sessions)
  - Homework requires programming in Python
- Website :
- Assessment
- Final exam
  - Date to be determined

Instructor

- Christophe Prud'homme
  - Room P-210, UFR Math Info, third floor)
  - Email: `prudhomme@unistra.fr`
  - Office hours: by appointment via slack

Course Outline

Topics to be discussed: ROM and Data driven ROM

- Origin: structural mechanics; Parametrized PDEs
- Stationary and time-dependent problems: a priori theory, certified error bounds, coercivity/inf-sup lower bounds, non intrusive reduced basis (implementation in Python)
- Non-Affine, Non-Linear problems: introduction to EIM (implementation in Python)
- First steps in Data assimilation
  - Introduction to GEIM (implementation in Python)
  - Introduction to PBDW (implementation in Python)
  - Introduction to Kalman Filters and Ensemble Kalman Filters (implementation in Python)
  - Putting it all together, Data Assimilation

Keywords/Methods: Galerkin, POD, Greedy, RB, Intrusive RB, Non Intrusive RB, CRB, EIM, DEIM, GEIM, PBDW, 3DVAR, 4DVAR, EnKF

Digital Twins

What are Digital Twins: Definition

Figure 1: Digital Twins: Smart Cities (illustration)

A **Digital Twin** is a virtual representation of a physical system that mirrors the real system in real time by using

- **data** : observations of the real asset
- **models**: data driven and pde based, and
- **simulations** : real-time predictions from models as well as combination of data and models

Why Develop and Use Digital Twins?

Reasons to Develop Digital Twins:

- **Enhanced understanding and analysis:** Digital Twins help to better understand complex physical systems.
- **Experimentation and innovation:** Safely experiment with new regimes and treatments without impacting the real system.
- **Predictive capabilities:** Predict the performance of systems under various conditions.
- **Risk and uncertainty quantification:** Measure and manage uncertainty, and assist in decision-making by providing risk assessments.

Why Now?

- Affordable sensors and abundant data.
- Advances in AI, machine learning and ROM.
- Increased computing power (from edge to cloud)

How to Develop Digital Twins? (Key Ingredients)

Key Ingredients for Developing Digital Twins:

- **Real-world data:** Gather data from sensors and measurement devices.
- **Modeling the system:** Develop models that represent the physical system's behavior, including physics-based models or data-driven approaches.
- **High-dimensional data:** Handle complex systems that involve multiple scales, multiphysics, and uncertain parameters.
- **Simulations and predictions:** Generate predictions beyond current measurements using both simulation and machine learning.

Data Analytics and Simulation:

- Combine simulation with analytics for comprehensive modeling.
- Use models to generate data and vice versa (virtuous cycle).

Applications of Digital Twins

Key Applications:

- **System optimization:** Monitor and optimize system performance in real time (e.g., smart cities, manufacturing, healthcare).
- **Predictive maintenance:** Predict potential system failures and prevent downtime.
- **Innovation and testing:** Test new scenarios and optimize new processes in a virtual environment before applying them to the physical system.
- **Decision support:** Aid in making data-driven decisions by simulating outcomes and risks.

Examples:

- Smart cities integrating smart transportation, healthcare, and energy grids.
- Cyber-physical systems that manage complex systems with interconnected physical and computational components.

Cyber-Physical Systems (CPS)

Definition:

- **Cyber-Physical Systems (CPS)** are smart systems that integrate physical components with computational elements.
- CPS often operate with the Internet of Things (IoT) and the industrial internet, enabling seamless interaction between physical assets and digital infrastructure.

Applications:

- **Smart Cities:** Integration of transportation, energy, and healthcare into a connected infrastructure.
- **Manufacturing:** Advanced automation and real-time monitoring of industrial processes.
- **Healthcare:** Real-time monitoring of patients using interconnected medical devices.

Impact:

- Enhances productivity, efficiency, and decision-making across industries.
- Bridges the gap between the physical and digital worlds, fostering innovation.

The Inference Cycle - Overview

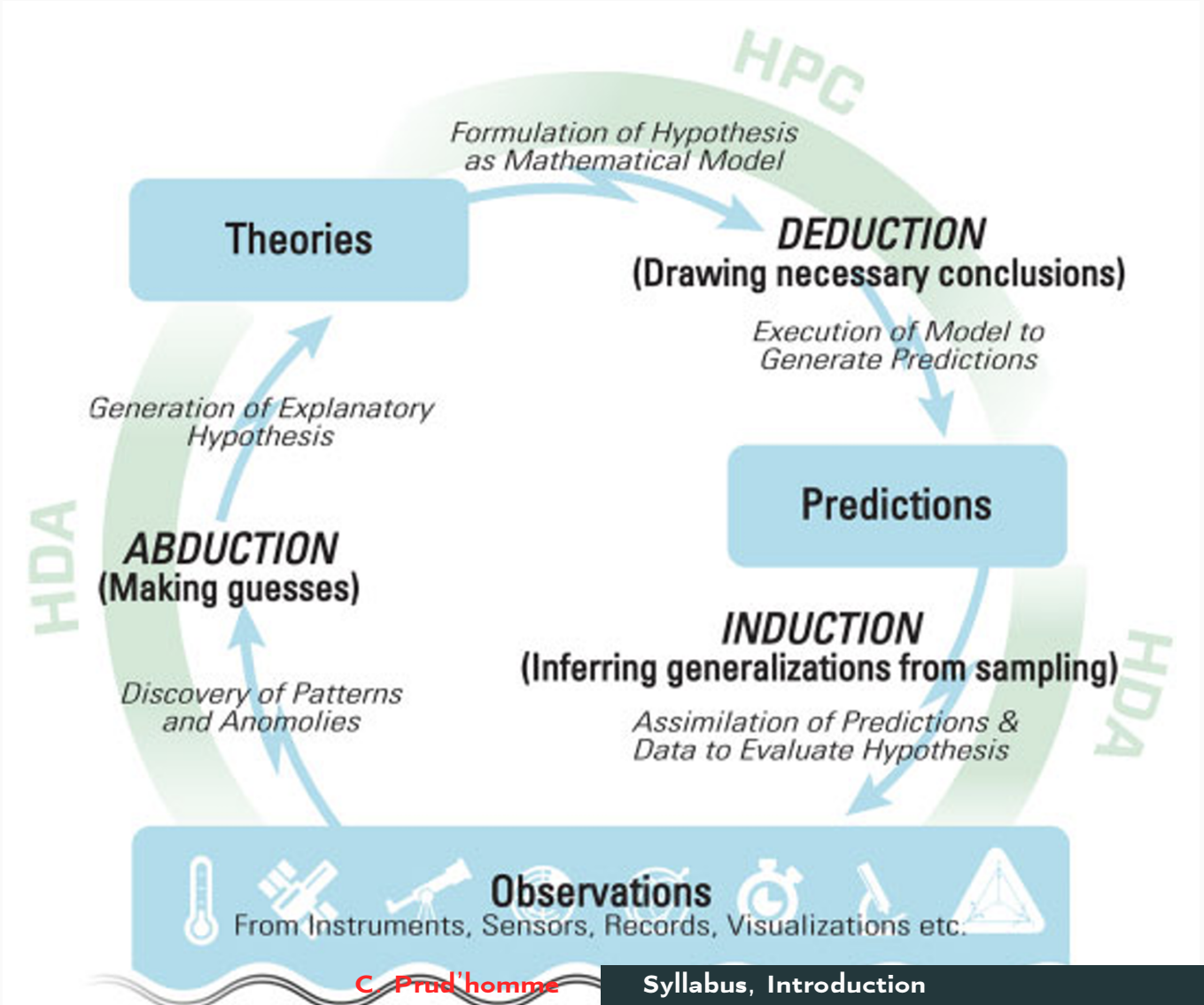
What is the Inference Cycle?

- A process to model and simulate complex systems by continuously improving understanding based on data and observations.
- The cycle uses three forms of inference: **Abduction**, **Deduction**, and **Induction**.

Key Steps:

- **Abduction:** Guessing causes based on observed effects and patterns.
- **Deduction:** Drawing conclusions based on models and predictions.
- **Induction:** Generalizing results and refining models based on new observations.

The Inference Cycle - Overview



Relevance of the Inference Cycle to Digital Twins

How it Applies to Digital Twins:

- Digital Twins combine real-time data, modeling, and simulation to predict outcomes.
- The inference cycle continuously refines the twin's understanding of the system through:
  - **Abduction:** Identifying new patterns or anomalies in real-time data.
  - **Deduction:** Simulating system behavior based on the twin's model.
  - **Induction:** Adjusting the model based on the system's actual response.

Impact:

- Improves accuracy of predictions and optimizes decision-making.
- Facilitates a virtuous cycle of data assimilation, model updating, and performance monitoring.

Digital Twin in the Digital Continuum

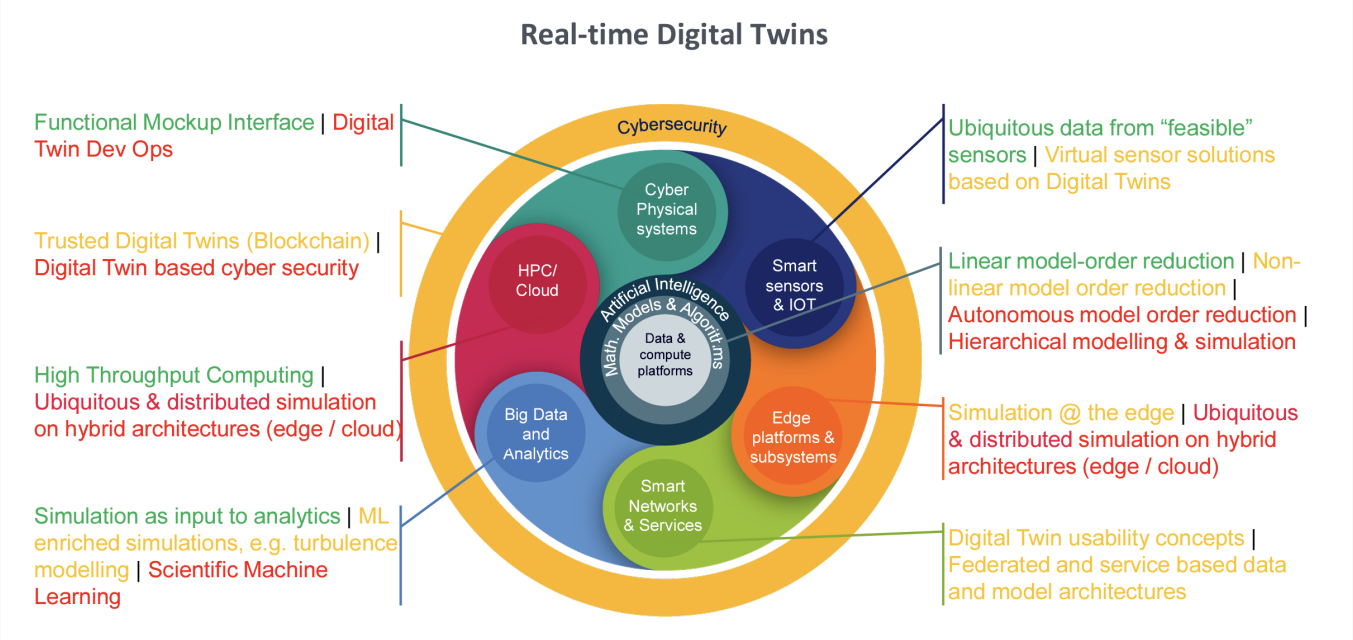


Figure 3: Digital Twin in the Digital Continuum, TCI Initiative

Introduction to Models, Data, and Coupling

Overview:

- Digital infrastructure serves as a continuum, involving AI, machine learning (ML), and data science.
- Three key questions:
  1. What is meant by a model?
  2. What are data to be used for?
  3. How can models and data be coupled?

Aim: To develop models that derive information from data and use that information to make predictions.

Types of Models

Two Main Types of Models:

- **Equation-based models:** Derived from conservation laws and fundamental principles (e.g., physics-based models).
- **Statistical, data-driven models:** Based on observed data and their analysis (e.g., machine learning models).

Guiding Principle: *Lex Parsimonia* (Occam's Razor):

- Simplify the model by eliminating unnecessary terms and data to avoid overfitting.

The Role of Data

Data Use:

- Data is not modeled itself, but its **information content** is essential for selecting optimal models.
- Avoid "data crimes": Be cautious when using data to prevent bias and overfitting.

Model Properties to Aim For:

- **Interpretability:** Models should provide clear and understandable results.
- **Generalization:** Models should work well across different datasets and situations.
- **Predictive Capacity:** Models should offer reliable predictions for unseen data.

Coupling Models and Data

The Importance of Coupling:

- Combining models and data requires careful consideration to avoid introducing bias or overfitting.
- The scientific approach involves:
  1. Starting from hypotheses.
  2. Developing models based on these hypotheses.
  3. Using data to validate and adjust the models.

Goal: To create models that balance interpretability, generalization, and predictive capacity while avoiding overfitting and underfitting.

Example: Overfitting and Underfitting

Overfitting: A model that works well for the training data but performs poorly on new, unseen data. **Underfitting:** A model that fails to capture the underlying trend in the data.

Goal: To develop robust models that avoid these issues by balancing complexity and generalization.

Introduction to Digital Twin Examples

Overview:

- Digital Twins can be applied in many different domains, ranging from toy cases to highly complex real-world systems.
- Students will choose one example and present it in detail.

Categories of Examples:

- Predictive maintenance
- Personalized medicine
- Autonomous transport
- Environmental surveillance
- Industrial use cases

Example 1 - Predictive Maintenance

Definition:

- Predictive maintenance involves monitoring the health of expensive, critical systems (e.g., aircraft, nuclear power plants).
- Digital Twins can help predict when and where maintenance is required, reducing costs and avoiding system breakdowns.

Applications:

- Aircraft engine monitoring
- Nuclear power plant system maintenance

Example 2 - Personalized Medicine

Definition:

- Digital Twins are used to simulate individual patients' medical conditions, creating a virtual model based on medical data.
- These models help in customizing treatments for conditions such as heart diseases and tumors.

Applications:

- Mapping individual genomes for therapy design.
- Virtual models of organs like the heart for personalized care.

Example 3 - Autonomous Transport

Definition:

- Autonomous vehicles (ships, trains, aircraft) rely on Digital Twins for navigation, real-time monitoring, and predictive decision-making.

Applications:

- Automatic piloting systems in aircraft and ships.
- Self-driving cars and trains, including trajectory optimization and collision avoidance.

Example 4 - Environmental Surveillance

Definition:

- Digital Twins are applied to environmental monitoring, including weather predictions, air pollution, and seismic activity.
- These models help improve prediction accuracy for natural events and monitor ecological conditions.

Applications:

- Weather forecasting (e.g., numerical weather prediction).
- Seismic monitoring and pollution tracking.
- Ecological studies, such as animal population monitoring.

Example 5 - Industrial Use Cases

Definition:

- In manufacturing and industry, Digital Twins are used to monitor production lines, optimize operations, and predict equipment failures.

Applications:

- Aerospace: Monitoring wing loads and landing gear.
- Oil and gas: Monitoring liquefied natural gas (LNG) systems.
- Manufacturing: Using ultrasonic welding and hydraulic press monitoring.

Instructions for Student Presentations

Assignment:

- Each group of students will choose one of the examples discussed and create a more detailed presentation of a concrete example.
- Presentations should cover:
  1. Detailed explanation of the chosen example.
  2. Current applications of Digital Twins in that field.
  3. if possible a description of the models, data and coupling used
  4. Potential future developments and challenges.

Format:

- 10-minute presentation
- Visual aids (slides, diagrams, etc)

Digital Twins, ROM and Data assimilation

Digital Twins, ROM and Data Assimilation: Why ?

A Digital Twin is a virtual representation of a physical system that mirrors the real system in real time by using data, models, and simulations. The key aspects of Digital Twins are:

- Real-time data integration: Continuous or periodic updates from sensors, devices, or other monitoring systems feed data into the model to reflect the real-world scenario.
- Simulation and prediction: The Digital Twin simulates the behavior of the physical system to predict future states, optimize operations, or test scenarios (e.g., manufacturing processes, energy grid management).
- Optimization and decision-making: Digital Twins provide a platform to make informed decisions using the virtual model to optimize outcomes based on both current and predicted conditions.

BUT

However, for complex systems (e.g., aeronautics, energy grids, manufacturing, health, sport), simulating the full system in real-time can be computationally expensive. This is where Reduced Order Methods (ROM) and Data Assimilation come into play.

Reduced Order Methods (ROM)

Reduced Order Methods aim to simplify high-fidelity models of complex systems, making simulations faster while retaining essential accuracy. ROMs are key to enabling Digital Twins to run simulations in real time or near real-time.

Types of ROM:

- A metamodel (also called a surrogate model) is a simplified model that approximates the behavior of a more complex, high-fidelity model.
- Reduced Basis (RB): This method selects a small set of representative system states (or "basis functions") to project the high-dimensional system into a lower-dimensional space. It reduces the computational load while maintaining accuracy.
- Empirical Interpolation Method (EIM): Used to handle nonlinear systems by approximating functions based on a small number of strategically selected points.
- Generalized Empirical Interpolation Method (GEIM): An extension of EIM that can handle multimodal data (e.g., different types of sensor inputs). GEIM is particularly useful in Digital Twins because it allows integrating real-time data from various sources into the reduced model.

By using ROM, Digital Twins can efficiently

What is a Metamodel?

- A metamodel, or surrogate model, is a simplified model that approximates the behavior of a more complex, high-fidelity system.
- It is created using data generated by simulations or real-world measurements.
- Metamodels reduce computational time while maintaining essential accuracy.

Types of Metamodels

- **Polynomial Chaos Expansions (PCE)**: Approximates complex functions using orthogonal polynomials.
- **Gaussian Process Regression (Kriging)**: Provides predictions and measures uncertainty.
- **Neural Networks**: Learns nonlinear relationships from data, commonly used in machine learning.
- **Support Vector Machines (SVM)**: A machine learning approach for classification and regression tasks.

Metamodels as ROM (Reduced Order Methods)

- Metamodels simplify complex systems, acting as ROMs by reducing dimensionality.
- Computational efficiency is achieved by approximating the behavior of the full system.
- Metamodels allow for real-time or near real-time simulations.

Comparing Metamodels to Other ROM Techniques

- **Metamodels vs. Reduced Basis (RB)**: RB reduces the system by projecting it into a lower-dimensional space; metamodels approximate outputs directly from data.
- **Metamodels vs. EIM/GEIM**: EIM/GEIM handle nonlinear systems; metamodels are often more flexible for data-driven approaches.

Data Assimilation

Data Assimilation integrates real-world data into models to update and correct predictions, ensuring that the virtual model remains aligned with the physical system in Digital Twins.

Key Techniques

- **Kalman Filter:** Updates model state based on new observations (stochastic, for linear systems), extension to nonlinear systems (Extended Kalman Filte or Ensemble Kalman Filter).
- **Variational Methods:** Minimizes difference between predictions and data (deterministic).
- **GEIM:** ROM-based, efficiently approximates high-dimensional, nonlinear systems from multiple data sources.
- **PBDW:** ROM-based, integrates offline/online data, adjusts reduced models with real-time data for error correction.

Reduced Order Methods with Data Assimilation efficiently handle complex, nonlinear systems.

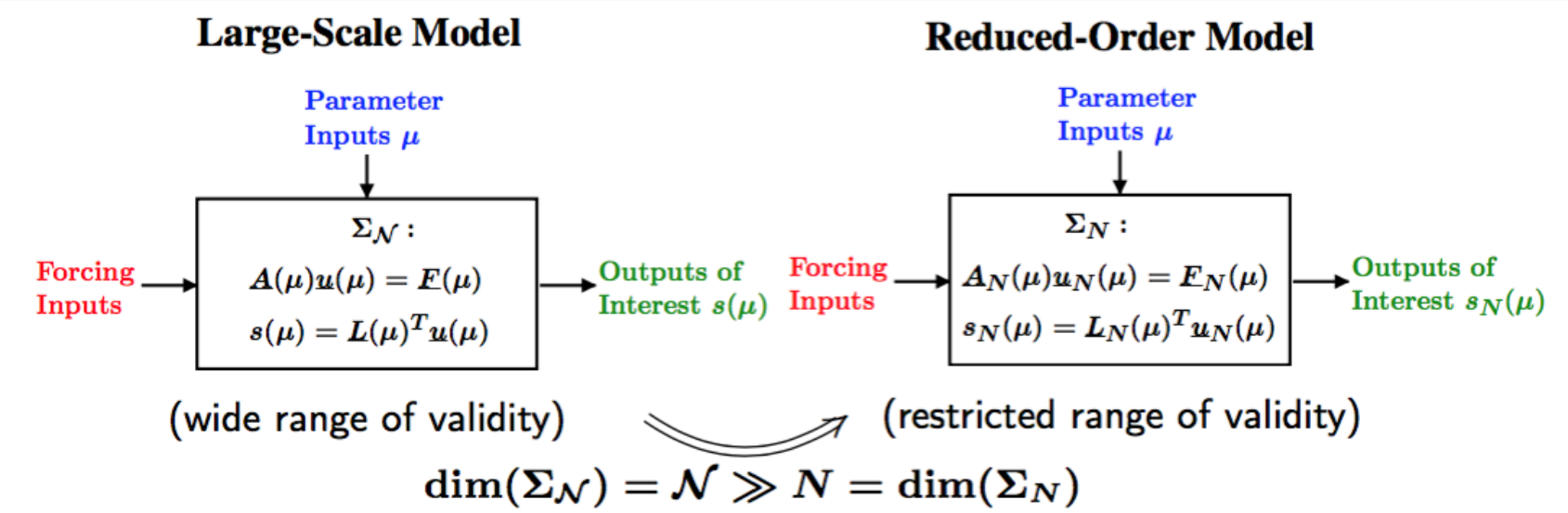
Model Order Reduction

Problem statement

Goal

Replicate input-output behavior of large-scale system Σ over a certain (restricted) range of

- forcing inputs and
- parameter inputs



Problem statement

Given large-scale system $\Sigma_{\mathcal{N}}$ of dimension $\mathcal{N}$, find a reduced order model Σ_N of dimension $N << \mathcal{N}$ such that: The approximation error is small, i.e., there exists a global error bound such that

- $\|u(\mu) - u_N(\mu)\| \leq \varepsilon_{\text{des}}$, and $|s(\mu) - s_N(\mu)| \leq \varepsilon^s_{\text{des}}, \forall \mu \in D^\mu$.
- Stability (and passivity) is preserved.
- The procedure is computationally stable and efficient.

Generalized Inverse Problem

- Given PDE(μ) constraints, find value(s) of parameter μ which:
  - (OPT) minimizes (or maximizes) some functional;
  - (EST) agrees with measurements;
  - (CON) makes the system behave in a desired manner;
  - or some combination of the above
- Full solution computationally very expensive due to a repeated evaluation for many different values of μ
- Goal: Low average cost or real-time online response

Methodologies

- Reduced Basis Methods
- Proper Orthogonal Decomposition
- Balanced Truncation
- Krylov Subspace Methods
- Proper Generalized Decomposition
- Modal Decomposition
- Physical model reduction
- ...

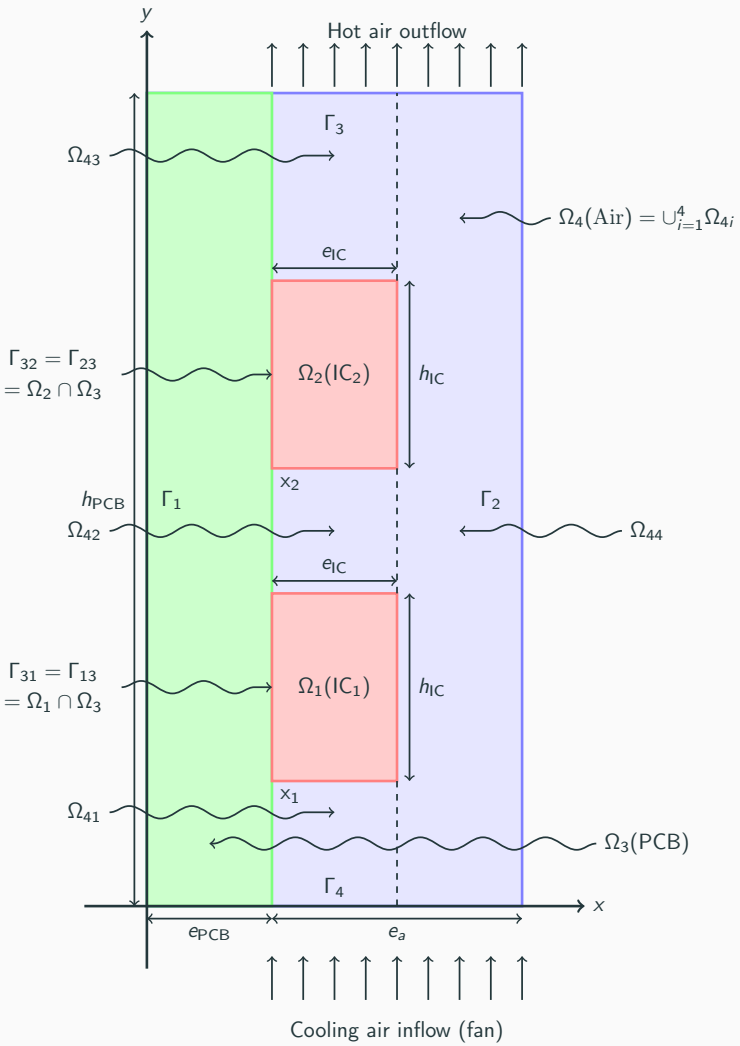
Disclaimer

Model Order Reduction Techniques

- DO NOT replace your favorite discretization scheme (e.g. FE, FV, FD), but instead are build upon and supplement these schemes.
- ARE NOT useful if you are interested in a single high-fidelity solution of your high-dimensional problem, but instead if you are interested in the many-query or real-time context.

Some Examples

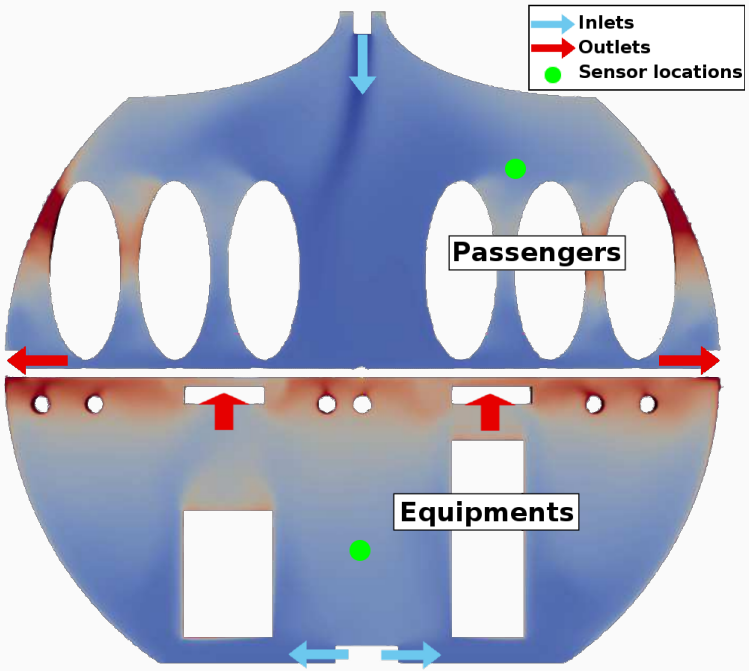
Thermal Testcase Description



Overview

- Heat-Transfer with conduction and convection possibly coupled with Navier-Stokes
- Simple but complex enough to contain all difficulties to test the certified reduced basis
  - non symmetric, non compliant
  - steady/unsteady
  - physical and geometrical parameters
  - coupled models
- Testcase can be easily complexified

Airbus Use-Case



Objective

Apply reduced basis methods on an aerothermal simulation in an avionic bay

Model

- Steady Navier-Stokes/Heat transfert
- Incompressible Newtonian Fluid
- Boussinesq Approximation
- Turbulent Flow

Some Scientific Issues

- Turbulence
- Mixed forced and natural convection
- Boundary conditions coupled to an ECS (Environment Control System)

HiFiMagnet project

High Field Magnet Modeling

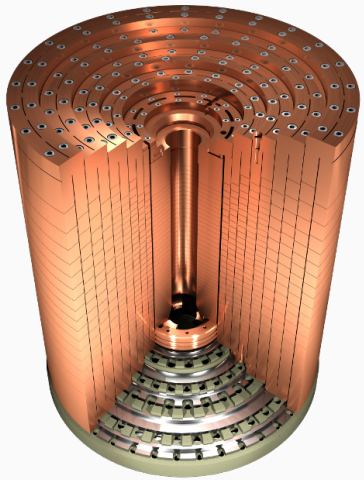
Laboratoire National des Champs Magnétiques Intenses

Large scale user facility in France

- High magnetic field : from 24 T
- Grenoble : continuous magnetic field (36 T)
- Toulouse : pulsed magnetic field (90 T)

Application domains

- Magnetoscience
- Solide state physic
- Chemistry
- Biochemistry



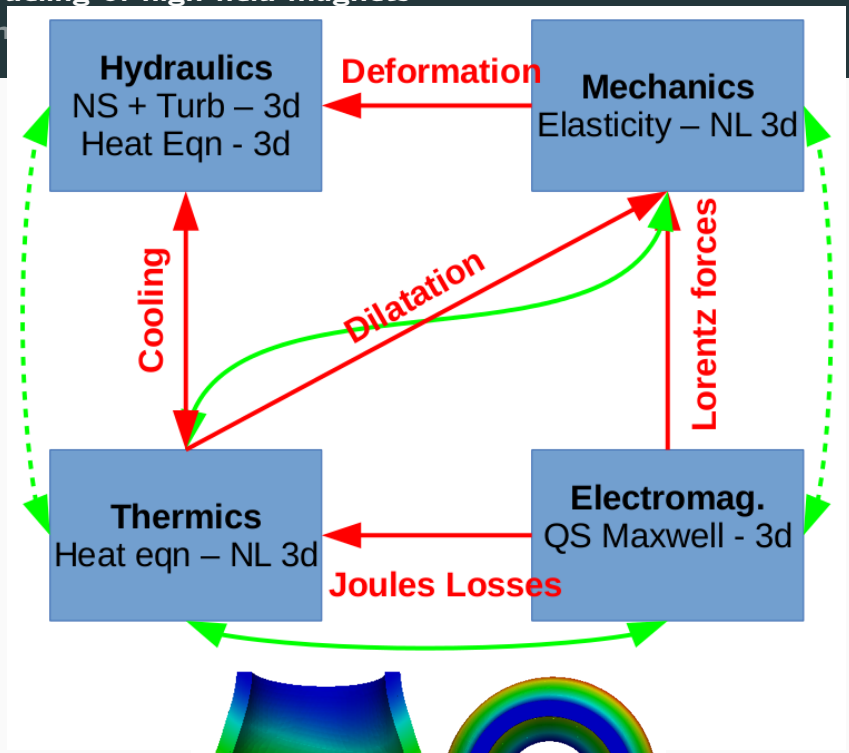
Magnetic Field

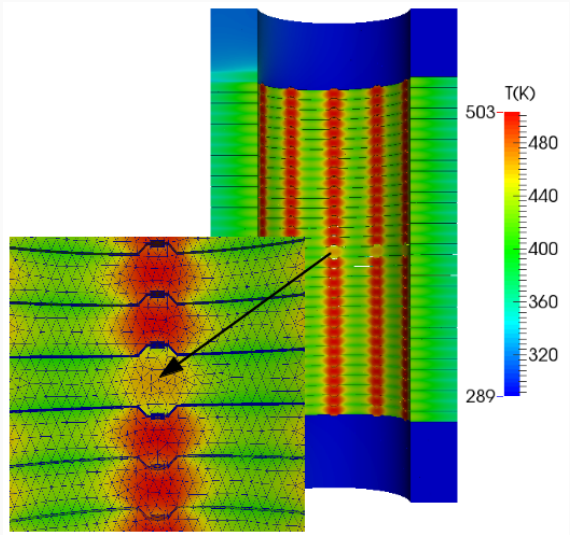
- Earth : $5.8 \cdot 10^{-4}\,T$
- Supraconductors : $24\,T$
- Continuous field : $36\,T$
- Pulsed field : $90\,T$

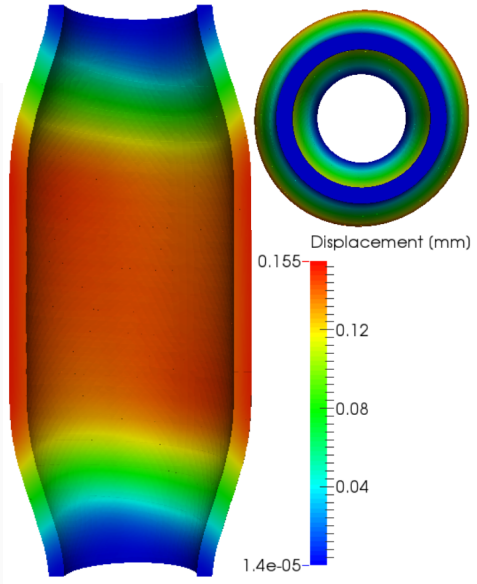
Access

- Call for Magnet Time : 2 $\times$ per year
- $\approx$ 140 projects per year

High Field Magnet Modeling







Why use Reduced Basis Methods ?

Challenges

- Modeling : multi-physics non-linear models, complex geometries, genericity
- Account for uncertainties : uncertainty quantification, sensitivity analysis
- Optimization : shape of magnets, robustness of design

Objective 1 : Fast

- Complex geometries
  - Large number of dofs
- Uncertainty quantification
  - Large number of runs

Objective 2 : Reliable

- Field quality
- Design optimization
  - Certified bounds
  - Reach material limits

Summary

Many problems in computational engineering require

many or real-time evaluations of PDE(μ)-induced input-output relationships.

Model order reduction techniques enable

certified, real-time calculation of outputs of PDE(μ) for parameter estimation, optimization, and control.

Reduced Basis Method

The Reduced Basis Method

Reduced basis method

A model order reduction technique that allows efficient and reliable reduced order approximations for a large class of parametrized partial differential equations (PDEs) in real-time or in the limit of many queries.

The Reduced Basis Method

Reduced basis method
A model order reduction technique that allows efficient and reliable reduced order approximations for a large class of parametrized partial differential equations (PDEs) in real-time or in the limit of many queries.

- Comparison to other model reduction techniques:
  - Parametrized problems(material, constants, geometry,...)
  - A posteriori error estimation
  - Offline-online decomposition
  - Greedy algorithm (to construct reduced basis space)

The Reduced Basis Method

Reduced basis method
A model order reduction technique that allows efficient and reliable reduced order approximations for a large class of parametrized partial differential equations (PDEs) in real-time or in the limit of many queries.

- Comparison to other model reduction techniques:
  - Parametrized problems(material, constants, geometry,...)
  - A posteriori error estimation
  - Offline-online decomposition
  - Greedy algorithm (to construct reduced basis space)
- Motivation:
  - Efficient solution of optimization and optimal control problems governed by parametrized PDEs.

Problem Statement

Given $\underbrace{\mu}_{\text{parameter}} \in \underbrace{D^{\mu}}_{\text{parameter domain}}$

Problem Statement

Given $\underbrace{\mu}_{\text{parameter}} \in \underbrace{D^{\mu}}_{\text{parameter domain}}$, evaluate

$$\underbrace{s(\mu)}_{\text{output}} = L(\mu)^T \underbrace{u(\mu)}_{\text{field variable}}$$

Problem Statement

Given $\underbrace{\mu}_{\text{parameter}} \in \underbrace{D^{\mu}}_{\text{parameter domain}}$, evaluate

$$\underbrace{s(\mu)}_{\text{output}} = L(\mu)^T \underbrace{u(\mu)}_{\text{field variable}}$$

where $u(\mu) \in \underbrace{X}_{\text{FE space}}$ satisfies a $PDE_{FE}(\mu)$

Problem Statement

Given $\underbrace{\mu}_{\text{parameter}} \in \underbrace{D^{\mu}}_{\text{parameter domain}}$, evaluate

$$\underbrace{s(\mu)}_{\text{output}} = L(\mu)^T \underbrace{u(\mu)}_{\text{field variable}}$$

where $u(\mu) \in \underbrace{X}_{\text{FE space}}$ satisfies a $PDE_{FE}(\mu)$

$$\underbrace{A(\mu)}_{\text{linear operator}} u(\mu) = \underbrace{f(\mu)}_{\text{loading,control...}}$$

Problem Statement

Given $\underbrace{\mu}_{\text{parameter}} \in \underbrace{D^{\mu}}_{\text{parameter domain}}$, evaluate

$$\underbrace{s(\mu)}_{\text{output}} = L(\mu)^T \underbrace{u(\mu)}_{\text{field variable}}$$

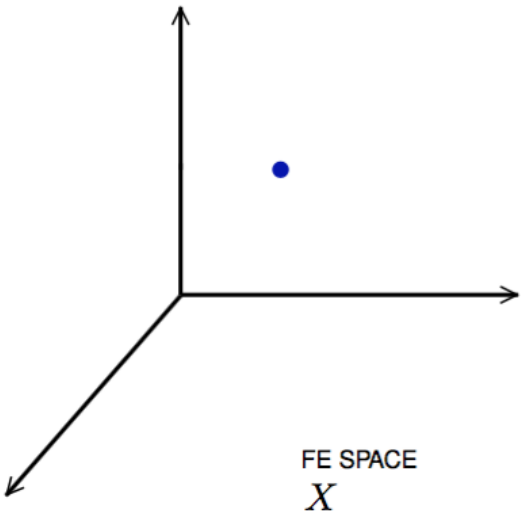
where $u(\mu) \in \underbrace{X}_{\text{FE space}}$ satisfies a $PDE_{FE}(\mu)$

$$\underbrace{A(\mu)}_{\text{linear operator}} u(\mu) = \underbrace{f(\mu)}_{\text{loading,control...}}$$

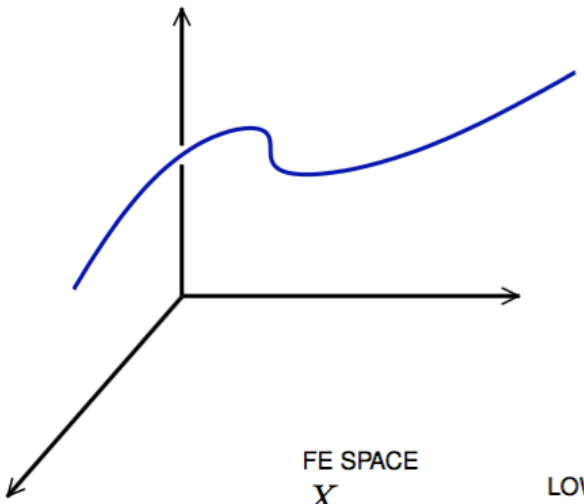
Difficulties:

- Need to solve $PDE_{FE}(\mu)$ numerous times at different values of μ
- Finite element space X has large dimension $\mathcal{N}$

The Main Idea - Key Observation



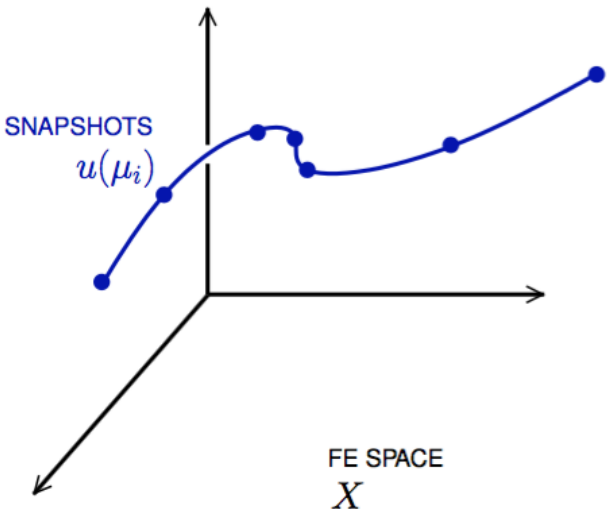
The Main Idea - Key Observation



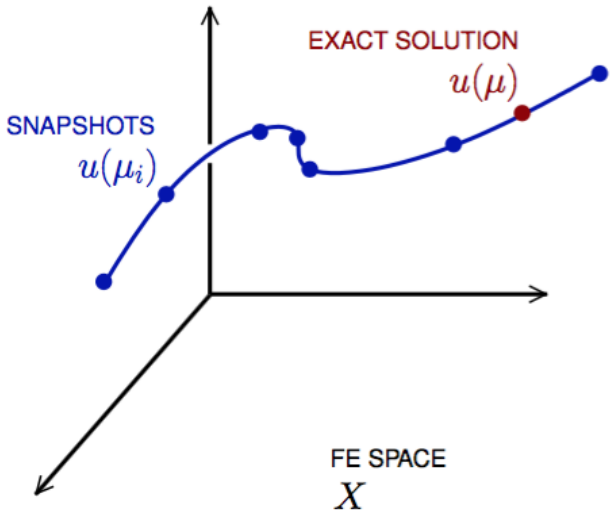
FE SPACE
X

LOW-DIMENSIONAL SOLUTION MANIFOLD
$\mathcal{M} = \{u(\mu) | \mu \in \mathcal{D}\}$

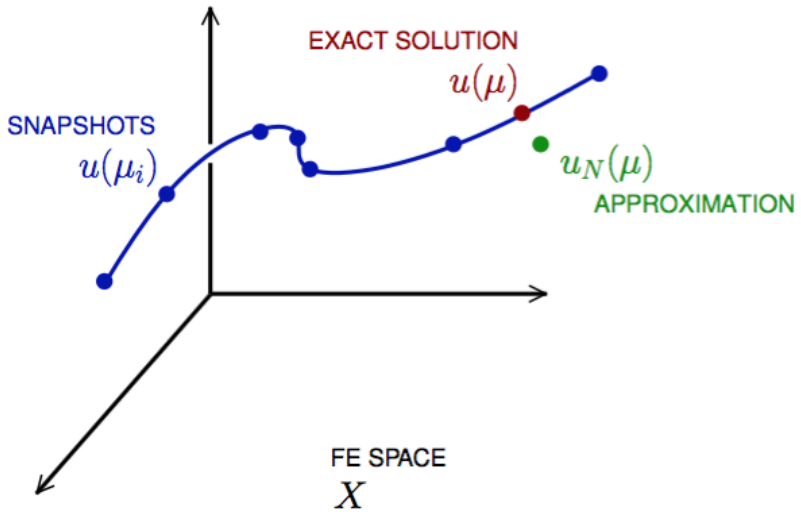
The Main Idea - Key Observation



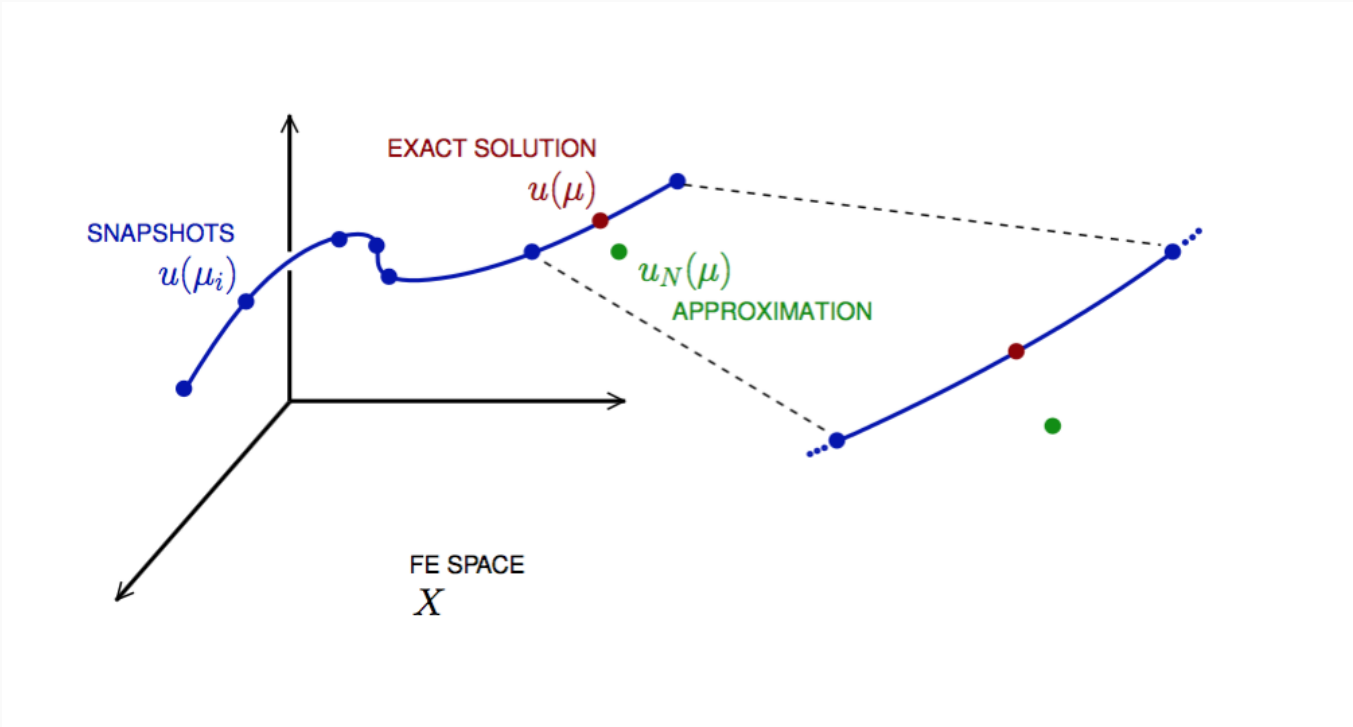
The Main Idea - Key Observation



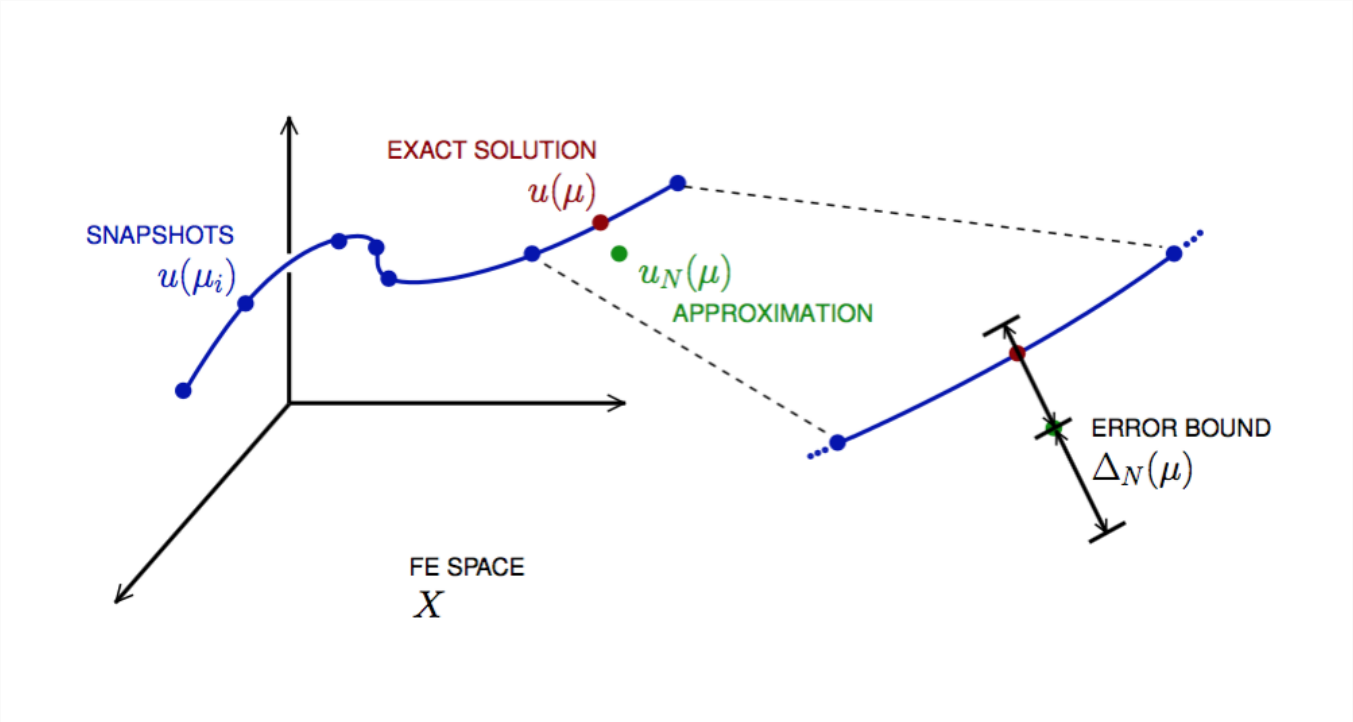
The Main Idea - Key Observation



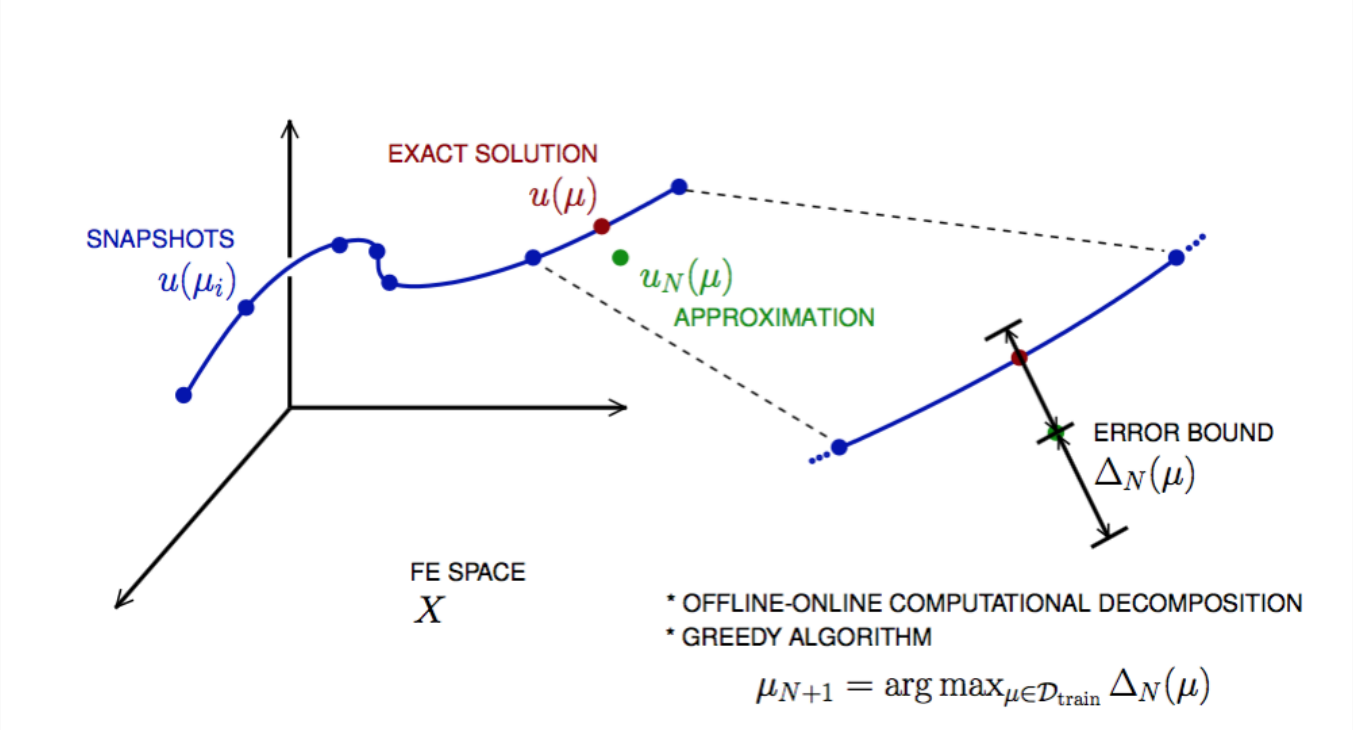
The Main Idea - Key Observation



The Main Idea - Key Observation

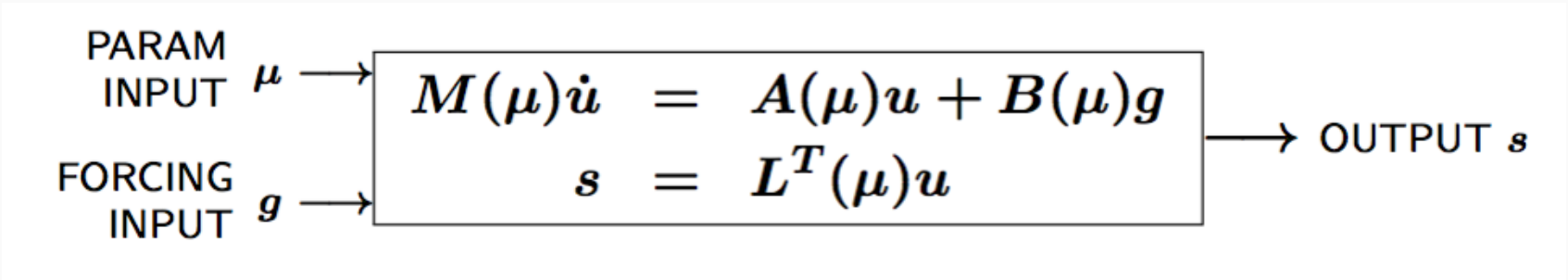


The Main Idea - Key Observation



General Problem Statement

Given a system $\Sigma_{\mathcal{N}}$ of large dimension N,



where

- $u(\mu, t) \in \mathbb{R}^{\mathcal{N}}$, the state
- $s(\mu, t)$, the outputs of interest
- $g(t)$, the forcing or control inputs

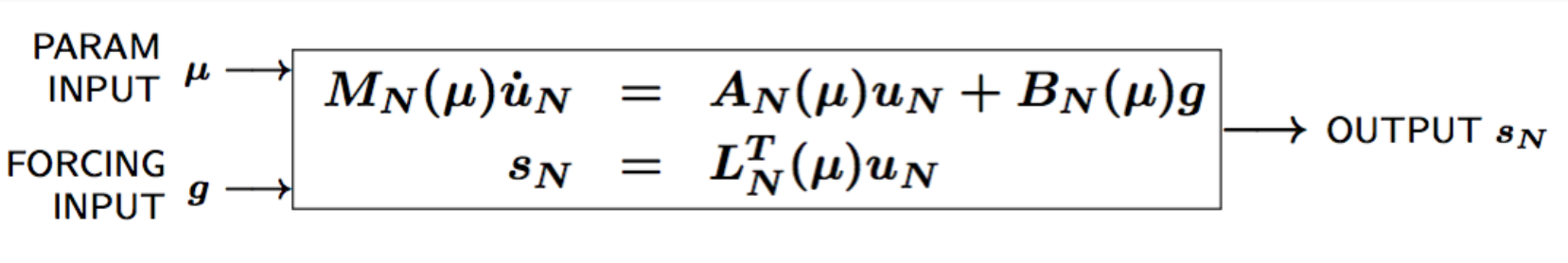
are functions of

- $\mu \in D$, the parameter inputs
- t, time

and the matrices M, A, B, and L also depend on μ ...

General Problem Statement

... construct a reduced order system Σ_N of dimension $N << \mathcal{N}$,



where $u_N(\mu) \in \mathbb{R}^N$ is the reduced state.

Special case

We start by considering $\dot{u} = 0$

Full Model

$$A(\mu)u(\mu) = F(\mu)$$
$$s(\mu) = L^T(\mu)u(\mu)$$

Reduced Model

$$A_N(\mu)u_N(\mu) = F_N(\mu)$$
$$s_N(\mu) = L_N^T(\mu)u_N(\mu)$$

Approximation

- Take "snapshots" at different μ-values: $u(\mu_i), i = 1 \dots N$, and let

$$Z_N = [\xi_1, \dots, \xi_N] \in \mathbb{R}^{\mathcal{N} \times N}$$

where the basis/test functions, ξ_i "=" $u(\mu_i)$, are orthonormalized

Approximation

- Take "snapshots" at different μ-values: $u(\mu_i), i = 1 \dots N$, and let

$$Z_N = [\xi_1, \dots, \xi_N] \in \mathbb{R}^{\mathcal{N} \times N}$$

  where the basis/test functions, ξ_i "=" $u(\mu_i)$, are orthonormalized
- For any new μ, approximate u by a linear combination of the ξ_i

$$u(\mu) \approx \sum_{i=1}^{N} u_{N,i}(\mu)\xi_i = Z_N u_N(\mu)$$

  determined by Galerkin projection, i.e.,

Approximation

- Take "snapshots" at different μ-values: $u(\mu_i), i = 1 \dots N$, and let

$$Z_N = [\xi_1, \dots, \xi_N] \in \mathbb{R}^{\mathcal{N} \times N}$$

  where the basis/test functions, ξ_i "=" $u(\mu_i)$, are orthonormalized
- For any new μ, approximate u by a linear combination of the ξ_i

$$u(\mu) \approx \sum_{i=1}^{N} u_{N,i}(\mu)\xi_i = Z_N u_N(\mu)$$

  determined by Galerkin projection, i.e.,

$$\underbrace{Z_N^T A(\mu) Z_N}_{\equiv A_N(\mu)} u_N(\mu) = \underbrace{Z_N^T F(\mu)}_{\equiv F_N(\mu)}$$

$$s_N(\mu) = \underbrace{L^T(\mu) Z_N}_{\equiv L_N^T(\mu)} u_N(\mu)$$

A posteriori error estimation

- Assume well-posedness; $A(\mu)$ pos.def. with min eigenvalue $\alpha_a := \lambda_1 > 0$, where $A\xi = \lambda X\xi$ and X corresponds to the X-inner product, $(v, v)_X = \|v\|_X^2$

A posteriori error estimation

- Assume well-posedness; $A(\mu)$ pos.def. with min eigenvalue $\alpha_a := \lambda_1 > 0$, where $A\xi = \lambda X\xi$ and X corresponds to the X-inner product, $(v, v)_X = \|v\|_X^2$
- Let $\underbrace{e_N = u - Z_N\ u_N}_{\text{error}}$, and $\underbrace{r = F - A\ Z_N\ u_N}_{\text{residual}}, \forall \mu \in D$, so that

$$A(\mu)e_N(\mu) = r(\mu)$$

A posteriori error estimation

- Assume well-posedness; $A(\mu)$ pos.def. with min eigenvalue $\alpha_a := \lambda_1 > 0$, where $A\xi = \lambda X\xi$ and X corresponds to the X-inner product, $(v, v)_X = \|v\|_X^2$
- Let $\underbrace{e_N = u - Z_N\ u_N}_{\text{error}}$, and $\underbrace{r = F - A\ Z_N\ u_N}_{\text{residual}}, \forall \mu \in D$, so that

$$A(\mu)e_N(\mu) = r(\mu)$$

- Then for any $\mu \in D$, LAX-MILGRAM

$$\|u(\mu) - Z_N u_N(\mu)\|_X \leq \frac{\|r(\mu)\|_{X'}}{\alpha_{LB}(\mu)} =: \Delta_N(\mu)$$

$$|s(\mu) - s_N(\mu)| \leq \|L\|_{X'} \Delta_N(\mu) =: \Delta_N^s(\mu)$$

where $\alpha_{LB}(\mu)$ is a lower bound to $\alpha_a(\mu)$, and $\|r\|_{X'} = r^T X^{-1} r$.

Offline-Online decomposition

How do we compute u_N, s_N, Δ_N^s for any μ efficiently online?

Offline-Online decomposition

How do we compute u_N, s_N, Δ_N^s for any μ efficiently online?

We assume

$$A(\mu) = \sum_{q=1}^{Q} \underbrace{\theta^q(\mu)}_{\text{parameter dependent coefficients}} \underbrace{A^q}_{\text{parameter independent matrices}}$$

so that

$$A_N(\mu) = Z_N^T A(\mu) Z_N = \sum_{q=1}^{Q} \theta^q(\mu) \underbrace{Z_N^T A^q Z_N}_{\text{paramerer independent}}$$

Offline-Online decomposition

How do we compute u_N, s_N, Δ_N^s for any μ efficiently online?

We assume

$$A(\mu) = \sum_{q=1}^{Q} \underbrace{\theta^q(\mu)}_{\text{parameter dependent coefficients}} \underbrace{A^q}_{\text{parameter independent matrices}}$$

so that

$$A_N(\mu) = Z_N^T A(\mu) Z_N = \sum_{q=1}^{Q} \theta^q(\mu) \underbrace{Z_N^T A^q Z_N}_{\text{paramerer independent}}$$

Since all required quantities can be decomposed in this manner, we can

- **OFFLINE**: Form and store μ-independent quantities at cost $O(\mathcal{N}^*)$
- ONLINE: For any $\mu \in D$, compute approx and error bounds at cost $O(QN^2 + N^3)$ and $O(Q^2N^2)$.

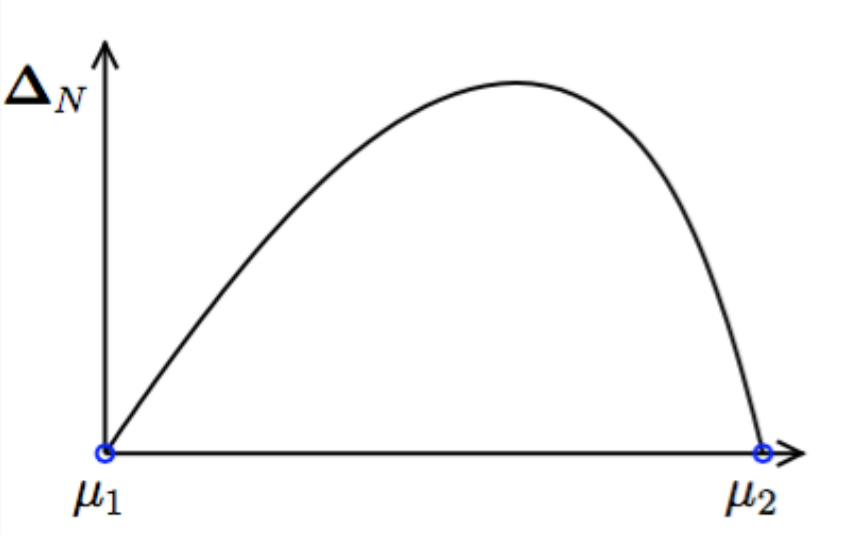
Greedy Algorithm

How do we choose the sample points μ_i (snapshots) optimally?

Greedy Algorithm

How do we choose the sample points μ_i (snapshots) optimally?

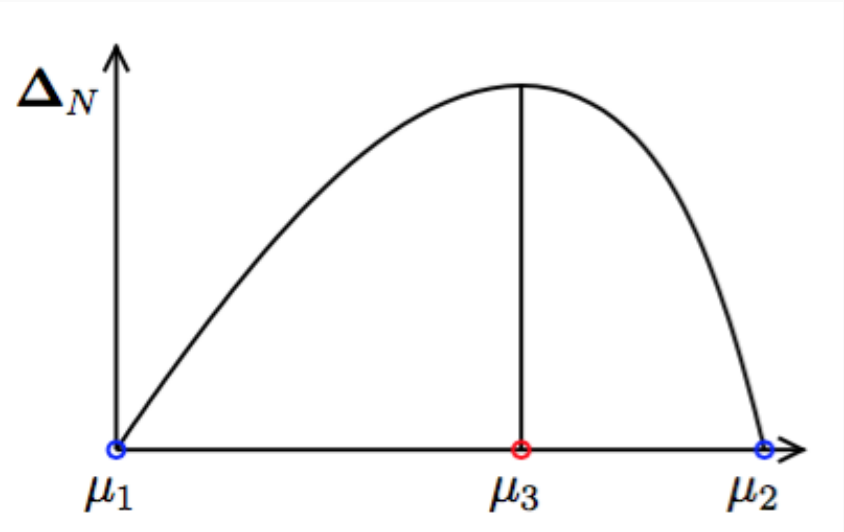
Given $Z_{N=2}$ "=" $[u(\mu_1), u(\mu_2)]$, we choose μ_{N+1} as follows:



Greedy Algorithm

How do we choose the sample points μ_i (snapshots) optimally?

Given $Z_{N=2}$ "=" $[u(\mu_1), u(\mu_2)]$, we choose μ_{N+1} as follows:

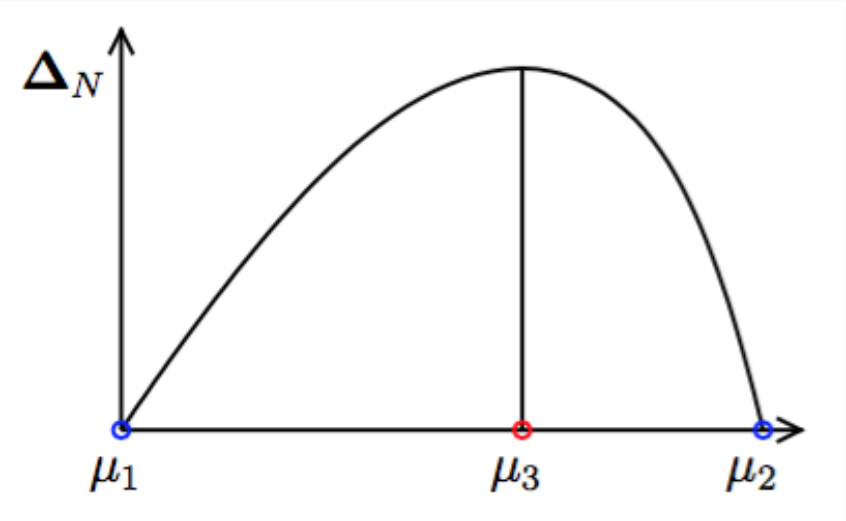


$$\mu_{N+1} = \text{argmax}_{\mu \in D^{\text{train}}} \frac{\Delta_N(\mu)}{\|u_N(\mu)\|_X}$$

Greedy Algorithm

How do we choose the sample points μ_i (snapshots) optimally?

Given $Z_{N=2}$ "=" $[u(\mu_1), u(\mu_2)]$, we choose μ_{N+1} as follows:



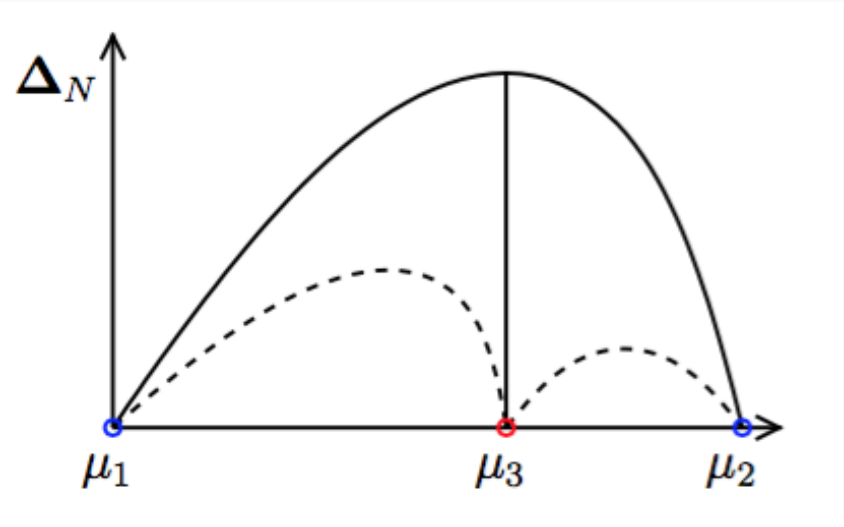
$$\mu_{N+1} = \text{argmax}_{\mu \in D^{\text{train}}} \frac{\Delta_N(\mu)}{\|u_N(\mu)\|_X}$$

$$Z_{N+1} = [u(\mu_1), \ldots, u(\mu_{N+1})]$$

Greedy Algorithm

How do we choose the sample points μ_i (snapshots) optimally?

Given $Z_{N=2}$ "=" $[u(\mu_1), u(\mu_2)]$, we choose μ_{N+1} as follows:



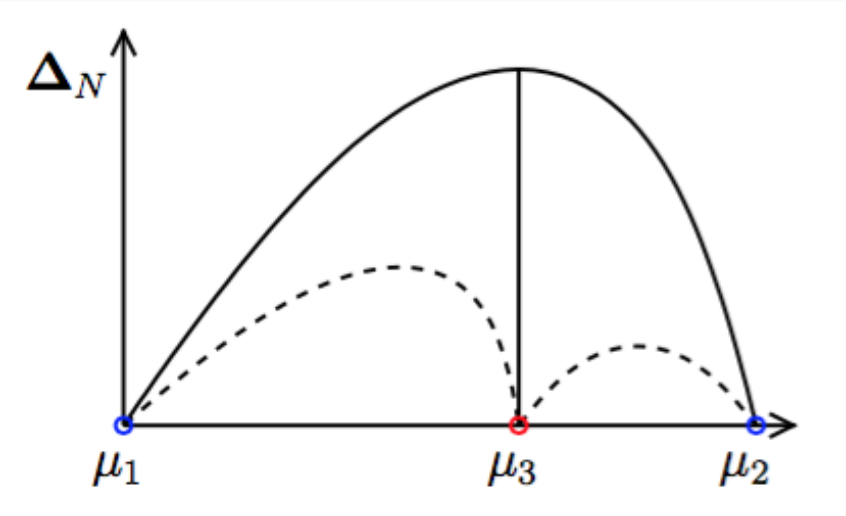
$$\mu_{N+1} = \text{argmax}_{\mu \in D^{\text{train}}} \frac{\Delta_N(\mu)}{\|u_N(\mu)\|_X}$$

$$Z_{N+1} = [u(\mu_1), \ldots, u(\mu_{N+1})]$$

Greedy Algorithm

How do we choose the sample points μ_i (snapshots) optimally?

Given $Z_{N=2}$ "=" $[u(\mu_1), u(\mu_2)]$, we choose μ_{N+1} as follows:



$$\mu_{N+1} = \text{argmax}_{\mu \in D^{\text{train}}} \frac{\Delta_N(\mu)}{\|u_N(\mu)\|_X}$$

$$Z_{N+1} = [u(\mu_1), \ldots, u(\mu_{N+1})]$$

- Key: $\Delta_N(\mu)$ is *sharp* and *inexpensive* to compute (online)
- Error bound "optimal" samples $\Rightarrow$ good approximation $u_N(\mu)$.