TABLE of CONTENTS

1. Introduction to PDE ---- pag 1.
2. Parametric variational problem ---- pag 3.
3. Full order Numerical model ---- pag 4.
4. Low fidelity Model ---- pag 6.
5. Greedy strategy ---- pag 9.
6. Different approach to compute the coercivity function ---- pag 11.
7. Intro to Neural Networks and Deep Learning ---- pag 14.
8. How to compute the weights ---- pag 17.
9. Convolutional Neural Network ---- pag 19.
10. Physics informer Neural Network ---- pag 23.
11. POD - Neural Network ---- pag 25.
12. Stokes Equations in MOR ---- pag 27.
13. Non linear problems ---- pag 31.
14. Navier Stokes Equations ---- pag 33.
15. Not Affine problem ---- pag 35.
16. Geometric Parametrization ---- pag 37.
17. Optimal Control ---- pag 38.

1 INTRODUCTION TO PDE

Notation. Consider $x \in \mathbb{R}^d$, $\underline{x} := (x_1, \dots, x_d)$, $A \in \mathbb{R}^{m,n}$, V an Hilbert space (vectorial space with scalar product & complet) and $u, v \in V$. The scalar product is $\langle u, v \rangle_V$ and the norm $\|u\|_V^2 = \langle u, u \rangle_V$. The dual space of V is $V' := \{F : V \to \mathbb{R} \text{ linear and continuous}\}$ and $\langle F, u \rangle_V = F(u) \in \mathbb{R}$.

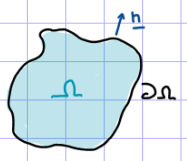
Example. Consider
i. $V = L^2(\Omega)$, with $\Omega \subset \mathbb{R}^d \Rightarrow \langle u, v \rangle_V = \int_\Omega u \cdot v$ and $\|u\|_V^2 = \int_\Omega u^2 = \langle u, u \rangle_V$
ii. $V = H^1(\Omega) \Rightarrow \langle u, v \rangle_V = \int_\Omega u \cdot v + \int_\Omega \nabla u \cdot \nabla v$ and $\|u\|_V^2 = \int_\Omega u^2 + \int_\Omega \nabla u \cdot \nabla u$

> The idea is to reduce the order of partial differential equations so that finding an approximate solution (whose distance from the real one is theoretically controlled) is much easier for the reduced problem

Example. Second order partial differential equations in $\mathbb{R}$ - PDE -

$x \in \mathbb{R}^d$ $\begin{cases} -\Delta u(\underline{x}) = f(\underline{x}) & \text{on } \Omega \subset \mathbb{R}^d \\ u(\underline{x}) = 0 & \text{on } \partial\Omega \end{cases}$ $\Rightarrow$ required $u \in C^2$, twice differentiable

where $\Delta u(x) = \frac{\partial^2 u}{\partial x^2}$



The solution is a function $u : \mathbb{R}^d \to \mathbb{R}$ and $f : \mathbb{R}^d \to \mathbb{R}$

we reformulate $-\Delta u = f$ as: search $u \in V$ such that

$-\int_\Omega \Delta u \cdot v = \int_\Omega f \cdot v \quad \forall v \in V$ **weak formulation**

$\Rightarrow$ we don't want to solve it directly (needed too many hypothesis). So we change its formulation so that the solutions requires less hypothesis

↓ by parts ↳ since less hp are need to find u

$\int_\Omega \nabla u \cdot \nabla v - \int_{\partial\Omega} v \nabla u \cdot \underline{n} = \int_\Omega f \cdot v$ $\Rightarrow$ $V = \{v \ \forall v,\ v \in L^2(\Omega),\ \nabla v \in L^2(\Omega)\} = H^1(\Omega)$ and $f \in L^2(\Omega)$ Hilbert space

Then we want to find $u \in H^1(\Omega)$ s.t. $\forall v \in H^1(\Omega)$, $f \in L^2(\Omega)$, so in general we introduce a ↳ hypothesis to integrate f

$\Rightarrow$ $a(u, v) = \int_\Omega \nabla u \cdot \nabla v - \int_{\partial\Omega} v \nabla u \cdot \underline{n}$ $\quad a : V \times V \to \mathbb{R}$ $\Rightarrow$ we need u only once differentiable

Hence, the problem becomes: finding $u \in V$ s.t. $a(u, v) = F(v) \ \forall v \in V$ that is the **variational problem VP**

↳ $F(v) : V \to \mathbb{R}$

Theorem - Lax Milgram If a is bilinear and continuous ($\exists \alpha \in \mathbb{R} < \infty$ s.t. $\forall u, v \in V$ $|a(u, v)| \le \alpha \|u\|_V \cdot \|v\|_V$ with $\alpha = \sup_{\substack{u, v \in V \\ \neq 0}} \frac{|a(u, v)|}{\|u\|_V \|v\|_V}$) and a is coercive ($\exists \alpha \in \mathbb{R} > 0$ s.t. $\forall v \in V$ $a(v, v) \ge \alpha \|v\|_V^2$, with $\alpha = \inf_{v \in V \neq 0} \frac{a(v, v)}{\|v\|_V^2} > 0$)

and F is linear and continuous ($\exists \rho \in \mathbb{R} < \infty$ s.t. $\forall v \in V$ $|F(v)| \le \rho \|v\|_V$, with $\rho = \sup_{v \in V \neq 0} \frac{|F(v)|}{\|v\|_V}$, $F \in V'$ ↳ $\rho = \|F\|_{V'}$ ↳ absolute value

$\Rightarrow$ (VP) has a unique solution and $\|u\|_V \le \frac{1}{\alpha} \|F\|_{V'}$

- VP is well-posed if $\forall F \in V'$ $\exists!\, u$ and u is controlled by F and α → controlled by data

Proof. Assume u exists and is unique, we want to prove $\|u\|_V \le \frac{1}{\alpha} \|F\|_{V'}$

- We know $a(u, v) = F(v) \ \forall v \in V$. Take $v = u$: $a(u, u) = F(u) \Rightarrow \alpha \|u\|_V^2 \le a(u, u) = F(u) \le \|F\|_{V'} \cdot \|u\|_V \Rightarrow \|u\|_V \le \frac{1}{\alpha} \|F\|_{V'}$ (property of dual norm)
- To prove uniqueness, assume $u_1, u_2 \in V$ solutions of VP: $a(u_1, v) = F(v)$ and $a(u_2, v) = F(v) \ \forall v \in V$

$\Rightarrow a(u_1 - u_2, v) = 0$ with $w = u_1 - u_2 \ \forall v \in V$ is VP, with solution w

$\Rightarrow \|w\|_V \le \frac{1}{\alpha} \|\tilde{F}\|_{V'} = 0$, $\|u_1 - u_2\|_V = 0 \Rightarrow u_1 = u_2$ $\quad \Rightarrow \tilde{F} = F(v) - F(v) = 0$: $\|u_1 - u_2\| = \|w\| \le \frac{1}{\alpha} \|F(v) - F(v)\| = \frac{1}{\alpha} \|\tilde{F}\| = 0 \Rightarrow \|w\| \ge 0 \Rightarrow \|w\| = 0$ ✓ □ ($\|w\| \le 0$)

both u_1 & u_2 : $a(u_1, v) = F(v) = a(u_2, v) \Rightarrow a(u_1, v) - a(u_2, v) = F(v) - F(v) = 0 \Rightarrow a$ is bilinear, then $a(u_1 - u_2, v) = 0$ ✓

Observation. Tacking $F_1 \neq F_2$ and u_i such that $a(u_1, v) = F_1(v)$ and u_2 s.t. $a(u_2, v) = F_2(v) \ \forall v \in V$

$\Rightarrow a(u_1 - u_2, v) = F_1(v) - F_2(v)$ is VP_2 and it has as solution $w = u_1 - u_2$

Moreover, we know $\|u_1 - u_2\|_V = \|w\|_V \le \frac{1}{\alpha} \|F_1 - F_2\|_{V'}$, in which $\frac{1}{\alpha}$ is the condition number of VP fixed $a(\cdot, \cdot)$ (proportional to $\frac{1}{\alpha}$)

↳ related to coercivity of $a(\cdot, \cdot)$, the smaller α, the harder is to solve (ill-conditioned)

a small perturbation on F is reflected on a small perturbation on u if α is NOT too small

Example. Consider the system

divergence $= \partial_{x_1}(\cdot) + \partial_{x_2}(\cdot) + \dots$

$\begin{cases} -\nabla \cdot (K(x) \nabla u(x)) = f(x) & \text{on } \Omega \subset \mathbb{R}^d \\ u(x) = 0 & \text{on } \Gamma_D \\ K(x) \nabla u(x) \cdot \underline{n} = \psi & \text{on } \Gamma_N = \partial\Omega \setminus \Gamma_D \end{cases}$ → before we have $-\nabla(\nabla u)$, now we add $K(x)$ ↳ boundary conditions

gradient $\nabla = \left(\frac{\partial}{\partial x_1}, \dots, \frac{\partial}{\partial x_n}\right)$

↳ diffusion parameter

↳ i. $u(x) = 0$ on Γ_D is the **Dirichlet boundary Condition**

ii. $K(x) \nabla u(x) \cdot \underline{n} = \psi$ on Γ_N is the **Neumann boundary Condition**

Let's compute the weak formulation: $\forall v \in V$ find $u \in V$ such that $-\int_\Omega \nabla(K(x)\nabla u(x))\, v = \int_\Omega f v$, $\forall v \in V$

$\Rightarrow$ we must define V such that it satisfies one boundary condition: we assume $V = \{ v \,|\, v|_{\Gamma_D} = 0 \}$

by parts $\int_\Omega K(x)\nabla u(x)\nabla v(x) - \int_{\Gamma_D} v\, K(x)\nabla u \cdot n - \int_{\Gamma_N} v\, K(x)\nabla u \cdot n = \int_\Omega f v \quad \Rightarrow \int_\Omega K(x)\nabla u \nabla v = \int_\Omega f v + \int_{\Gamma_N} v\, \psi$ VP

$= 0$ since $v|_{\Gamma_D} = 0$ (if $v = 0$ on Γ_D) $= \psi$ $\quad a(u,v)$ $\quad F(v)$

$\|v\|_V = \|v\|_{L^2(\Omega)} + \|\nabla v\|_{[L^2(\Omega)]^d}$

$|v|_V = \|\nabla v\|_{[L^2(\Omega)]^d}$

NB - Dirichlet boundary condition is imposed to the whole space (strong boundary condition)

Neumann boundary condition is imposed in the VP equations (weak boundary condition)

↳ $V = \{ v \in L^2(\Omega),\ \nabla v \in L^2(\Omega) \} \cap \{ v = 0 \text{ on } \Gamma_D \} = H^1(\Omega) \cap \{ v = 0 \text{ on } \Gamma_D \} = H^1_0(\Omega)$ and $f \in L^2(\Omega)$, $\psi \in L^2(\Omega)$

To have $\int_\Omega v v < \infty$, $\int_\Omega \nabla v \nabla u < \infty$, $\int_\Omega f v < \infty$ we need V defined ↑ sol is 0 on the Dirichlet boundary

↳ $K \in L^\infty(\Omega)$ and $\exists K_*, K^* : 0 < K_* \le K(x) \le K^* < +\infty \quad \forall x \in \Omega$

α of $a(u,v)$ $\quad$ β of $a(u,v)$

Poincaré Inequality. If $v \in H^1_0(\Omega) \Rightarrow \exists c_1 \in \mathbb{R} < +\infty$ s.t. $\|v\|_{L^2(\Omega)} \le c_1 \|\nabla v\|_{L^2(\Omega)^d}$, where $L^2(\Omega)^d = \underbrace{L^2(\Omega) \times \cdots \times L^2(\Omega)}_{d \text{ times}}$, then if

$v \in H^1_0(v) \Rightarrow \|v\|_{H^1_0(\Omega)} = \|v\|_{L^2(\Omega)} + \|\nabla v\|_{L^2(\Omega)^d} \le \tilde{c} \|\nabla v\|_{L^2(\Omega)^d}$ and $\|\nabla v\|_{L^2(\Omega)^d} \le \|v\|_{H^1_0(\Omega)} = \|v\|_{L^2(\Omega)} + \|\nabla v\|_{L^2(\Omega)^d}$

$\Rightarrow \|\cdot\|_{L^2(\Omega)^d}$ and $\|\cdot\|_{H^1_0(\Omega)}$ are equivalent and $\|v\|_{H^1(\Omega)} \simeq \|\nabla v\|_{L^2(\Omega)^d}$ if $v \in H^1_0(\Omega)$

If $v \in H^1_0(\Omega) \Rightarrow \exists c_2 \in \mathbb{R} < +\infty$ s.t. $\|v\|_{L^2(\Gamma_N)} \le c_2 \|v\|_{H^1_0(\Omega)}$. We can define the operator $\gamma : V \to W \equiv H^{1/2}(\Gamma_N)$, $v \mapsto \gamma(v) = v|_{\Gamma_N}$ the **continuous trace operator**, then:

↓ $V = H^1_0(\Omega)$, $f \in L^2(\Omega)$, $\psi \in L^2(\Omega)$ and $K \in L^\infty(\Omega)$ ($\exists K_*, K^*$ s.t. $0 < K_* \le K(x) \le K^* < \infty$)

The idea is to give a meaning to the value that the function has on the border of the domain

↓ map function from a space to the border of this space

② PARAMETRIC VARIATIONAL PROBLEMS

Consider $P \subset \mathbb{R}^P$, $\underline{\mu} \in P$ defined as vector of parameters $\underline{\mu} = [\mu_1, \dots, \mu_P]$. Now a VP($\underline{\mu}$) looks for $u(x;\underline{\mu}) \in V$ s.t. $a(u, v; \underline{\mu}) = F(v, \underline{\mu})$ $\forall v \in V$, $\forall \underline{\mu} \in P$ where $a : V \times V \times P \to \mathbb{R}$, $F : V \times P \to \mathbb{R}$

↳ $u(\underline{\mu})$

Theorem - Lax Milgram for VP($\underline{\mu}$) If

i. $a(\cdot,\cdot;\underline{\mu})$ is uniformly continuous: $\exists \gamma \in \mathbb{R} < +\infty$ s.t. $\forall \underline{\mu} \in P$ $|a(u,v;\underline{\mu})| \le \gamma(\underline{\mu}) \cdot \|u\|_V \cdot \|v\|_V \le \gamma \|u\|_V \cdot \|v\|_V < +\infty$
where $\gamma = \sup_{u,v \in V} \frac{|a(u,v,\underline{\mu})|}{\|u\|_V \cdot \|v\|_V} \le \gamma < \infty$

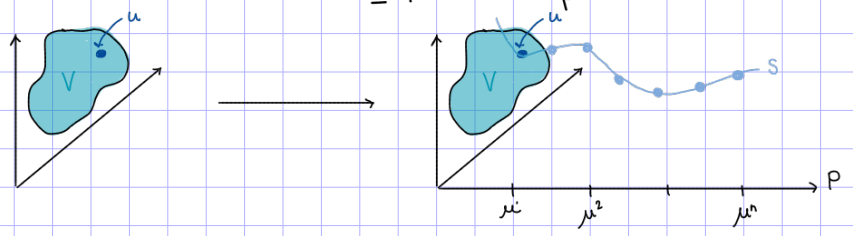
ii. $a(\cdot,\cdot;\underline{\mu})$ is uniformly coercive: $\exists \alpha \in \mathbb{R} < +\infty$ s.t. $\forall \underline{\mu} \in P$ $a(v,v;\underline{\mu}) \ge \alpha(\underline{\mu}) \|v\|_V^2 \ge \alpha \|v\|_V^2$, $\alpha(\underline{\mu}) = \inf_{v \in V} \frac{a(v,v;\underline{\mu})}{\|v\|_V} \ge \alpha > 0$

iii. F is uniformly continuous: $\exists \rho \in \mathbb{R} < +\infty$ s.t. $|F(v,\underline{\mu})| \le \rho(\underline{\mu}) \|v\|_V \le \rho \|v\|_V$, $\rho(\underline{\mu}) = \sup_{v \in V} \frac{|F(v,\underline{\mu})|}{\|v\|_V} = \|F(\cdot,\underline{\mu})\|_{V'} \le \rho < +\infty$

$\Rightarrow$ VP($\underline{\mu}$) has a unique solution $\forall \underline{\mu} \in P$

Example. We had one solution in V ⟶ But, if we add parameters the solution manifold is $\mathcal{M} = \{u(\underline{\mu}) \in V : \underline{\mu} \in P\} \subset V$

(we can find a solution for every value of $\underline{\mu}$)



So we define a curve of solutions

$S : P \to V$ solution mapping

$\underline{\mu} \mapsto u(\underline{\mu})$ of VP($\underline{\mu}$)

In general non linear

Proposition - If $a(\cdot,\cdot;\underline{\mu})$ and $F(\cdot,\underline{\mu})$ depend smoothly on $\underline{\mu} \in P$ $\Rightarrow$ $S : P \to V$ is smooth

↳ Proof on the notes if you're interested in

Example - Consider VP($\underline{\mu}$), $\int_{\Omega_P} K(\underline{\mu}) \nabla u \nabla v = \int_\Omega f v + \int_{\Gamma_N} \psi v$ $\forall v \in V$ and $K(x;\underline{\mu}) = K_0 + \sum_{i=1}^P K_i(x)\mu_i$,
$\underline{\mu} = (\mu_1, \dots, \mu_P) \in [0,1]^P = P$

with $K_0, K_i \in L^\infty(\Omega)$ s.t. • $K_0(x) + \sum_{i=1}^P |K_i(x)| \le K_* < +\infty$ $\forall x \in \Omega$
• $K_0(x) - \sum_{i=1}^P |K_i(x)| \ge K_* > 0$ $\forall x \in \Omega$

} ⇒ for the uniformity property of a

3 - FULL ORDER NUMERICAL MODEL (not reduced)

$VP(\mu)$ can not be solved automatically forall parameter (not computable) $\Rightarrow$ we solve it numerically (we select a finite subspace of V, V_δ, and we create a VP problem on V_δ). To do so, we must discretize the space: «δ» is the discretization parameter for the discrete problem (related to the finite dimension of V_δ) — GALERKIN METHOD

$VP_\delta(\mu)$: find $u_\delta \in V_\delta \subseteq V$ st $\forall \mu \in P$, $a(u_\delta, v_\delta; \mu) = F(v_\delta, \mu)$ $\forall v_\delta \in V_\delta$ (family of subspaces of V) → It is NOT garanted that there is a solution

Consider $\{V_\delta\}_{\delta>0}$ subspaces of V st the **consistency assumptions** holds: $\forall u \in V$ $\Rightarrow$ $\text{dist}_V(u, V_\delta) = \min_{v_\delta \in V_\delta} \|u - v_\delta\| \xrightarrow{\delta \to 0} 0$

$\delta \to 0 \Rightarrow V_\delta \to V$ **How fast $\text{dist}_V(u, V_\delta) \to 0$?** we choose V_δ so that $\exists$ an unique solution

We take $\{V_\delta\}_{\delta>0} \subseteq V$ such that $\exists Z \subset V$ normed space such that $\exists \psi_Z : \mathbb{R} \to \mathbb{R}$, $\psi_Z(\delta) \xrightarrow[\delta \to 0]{} 0$,

1. $\text{dist}_V(w, Z) \le \psi_Z(\delta) \|w\|_Z \quad \forall w \in Z$

controls the distance of the consistency assumptions so that we know how much fast goes $\text{dist}_V(u, V_\delta) \to 0$

NB Tells us that $u_\delta(\mu) \in V_\delta$, $\forall \mu \in P$ is the BEST APPROXIMATION we can obtain of $u \in V$ fixing $\{V_\delta\}_{\delta>0}$

Lemma - Cea If u_δ solution of $VP_\delta(\mu)$ and u is a solution of $VP(\mu)$

$\Rightarrow$ $\|u(\mu) - u_\delta(\mu)\|_V \le \frac{\gamma(\mu)}{\alpha(\mu)} \inf_{v_\delta \in V_\delta} \|u(\mu) - v_\delta\|_V \quad \forall \mu \in P$ 2.

(distance controlled by continuity and coercivity; for consistency assumption, if we chose the right δ, goes to zero)

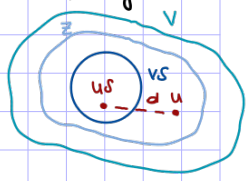
Theorem If $\{V_\delta\}_{\delta>0}$ satisfy 1. and 2. $\Rightarrow$ $\|u(\mu) - u_\delta(\mu)\|_V \le \frac{\gamma(\mu)}{\alpha(\mu)} \psi_Z(\delta) \|u(\mu)\|_Z$ → by hypothesis $u \in Z$

↳ Combined 1+2 $\Rightarrow$ A-priori error estimation

Example Consider $Z = H^r(\Omega)$, $r>1$ $\forall \mu \in P$, $V = H^1(\Omega)$ and $Z \subset V$.

1. $H^r \Rightarrow \psi_Z(\delta) = \delta^{r-1}$: $\text{dist}_V(w, V_\delta) \le \delta^{r-1} \|w\|_{H^r}$ $\forall \mu \in P$ $\Rightarrow$ $\|u(\mu) - u_\delta(\mu)\|_V \le \frac{\gamma(\mu)}{\alpha(\mu)} \delta^{r-1} \|u(\mu)\|_{H^r}$
2. $\nabla(K \cdot \nabla \mu) = f$ $u \in H^2 = Z$, $\psi_Z(\delta) = \delta^1$ $\Rightarrow$ $\|u(\mu) - u_\delta(\mu)\|_V \le \frac{\gamma(\mu)}{\alpha(\mu)} \delta \|u(\mu)\|_{H^2}$

In general



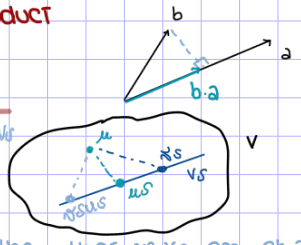
$VP_\delta(\mu)$: $a(u_\delta, v_\delta) = F(v_\delta)$ $\forall v_\delta \in V_\delta$ $\Rightarrow$ but $V_\delta \subset V$, so $\forall v_\delta \in V_\delta$: $a(u_\delta, v_\delta) - a(u, v) = F(v_\delta) - F(v) = 0$

$VP(\mu)$: $a(u, v) = F(v)$ $\forall v \in V$ $\qquad a(u_\delta - u, v_\delta) = 0 \quad \forall v_\delta \in V_\delta$

If $a(\cdot,\cdot)$ is symmetric, since is also bilinear and coercive, it's a **scalar product**

$\Rightarrow$ $a(u_\delta - u, v_\delta) = \langle u_\delta - u, v \rangle_a = 0$ $\Rightarrow$ $u_\delta - u \perp V_\delta$

$\Rightarrow$ **u_δ is the orthogonal projection of u on V_δ with respect to a-product**



↳ Subtracking $VP_\delta(\mu)$ from $VP(\mu)$: $a(u, v_\delta) - a(u_\delta, v_\delta) = F(v_\delta) - F(v_\delta) = 0$ $\forall \mu \in P \Rightarrow a(u - u_\delta, v_\delta; \mu) = 0$ $\forall v_\delta \in V_\delta$

Galerkin orthogonalization Consider $a(u - u_\delta, v_\delta) = 0$ $\forall v_\delta \in V_\delta$

$\Rightarrow$ $\|u - u_\delta\|_a = \sqrt{\langle u - u_\delta, u - u_\delta \rangle_a} = \sqrt{a(u - u_\delta, u - u_\delta)} \le \|u - v_\delta\|_a$ $\forall v_\delta \in V_\delta$, $\forall \mu \in P$

↳ being it the orthogonal projection, it's the element of V closest to u $\Rightarrow$ all the other $v_\delta \in V_\delta$ are at a greater distance

How to find $u_\delta(\mu)$ $\forall \mu \in P$? Consider $V_\delta \subset V$ finite, $N_\delta = \dim(V_\delta)$. Let's take $\{\varphi_k\}_{k=1}^{N_\delta}$ basis of V_δ (↳ basis functions)

$\varphi_k : \delta \to \mathbb{R}$ $\Rightarrow$ $\forall v_\delta \in V_\delta$ $v_\delta(\mu) = \sum_{n=1}^{N_\delta} \varphi_k \cdot v_{\delta n}(\mu)$ $\forall \mu \in P$ ($\in \mathbb{R}$)

Putting this expression in $VP_\delta(\mu)$, considering $v_\delta = \varphi_j$

we want to find $u_\delta \in V_\delta$ st $a(u_\delta, v_\delta; \mu) = F(v_\delta, \mu)$ $\forall v_\delta \in V_\delta$

$\Rightarrow$ $a\left(\sum_{k=1}^{N_\delta} u_{\delta,k} \varphi_k, v_\delta; \mu\right) = F(v_\delta; \mu)$ $\Rightarrow$ $\sum_{k=1}^{N_\delta} a(u_{\delta,k} \varphi_k, v_\delta; \mu) = F(v_\delta; \mu)$

$\Rightarrow$ $\sum_{k=1}^{N_\delta} u_{\delta,k} \cdot a(\varphi_k, v_\delta; \mu) = F(v_\delta; \mu)$ $\forall v_\delta \in V_\delta$ where N_δ and $u_{\delta,k}$ unknowns

What v_δ we take? We can take $v_\delta = \varphi_j$ for every $j \in \{1, \dots, N_\delta\}$ (one for every basis function φ_j)

$\Rightarrow$ it holds $\sum_{k=1}^{N_\delta} (u_{\delta,k})\, a(\varphi_k, \varphi_j; \mu) = F(\varphi_j; \mu)$ ⟶ N_δ equations and N_δ unknowns : I can solve it for $u_{\delta,k}$

In matrix form

$$u_\delta^\mu = \begin{bmatrix} u_{\delta,1} \\ \vdots \\ u_{\delta,N_\delta} \end{bmatrix} \begin{bmatrix} \quad \\ \quad \end{bmatrix} = \begin{bmatrix} f_1 \\ \vdots \\ f_{N_\delta} \end{bmatrix} = \begin{bmatrix} F(\varphi_1; \mu) \\ \vdots \\ F(\varphi_{N_\delta}; \mu) \end{bmatrix} \Rightarrow \begin{bmatrix} A^\mu \end{bmatrix} \begin{bmatrix} u_\delta^\mu \end{bmatrix} = \begin{bmatrix} f^\mu \end{bmatrix}$$

A^μ st $A_{jk} = a(\varphi_k, \varphi_j; \mu)$ $\quad \in \mathbb{R}^{N_\delta, N_\delta}$ $\quad \in \mathbb{R}^{N_\delta}$ $\quad \in \mathbb{R}^{N_\delta}$

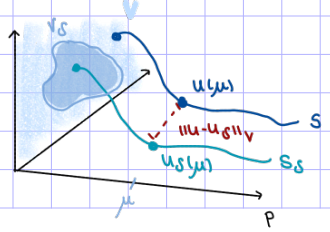
full problem
high fidelity problem
$A^\mu u_\delta^\mu = f^\mu$

So that we obtain a discrete map and a discrete manifold

$S_\delta : P \to V_\delta$

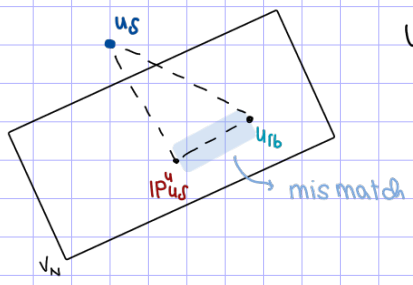
$\mu \mapsto u_\delta(\mu)$

$\mathcal{M}_\delta = \{u_\delta(\mu) \in V_\delta : \mu \in P\} \subset V_\delta$



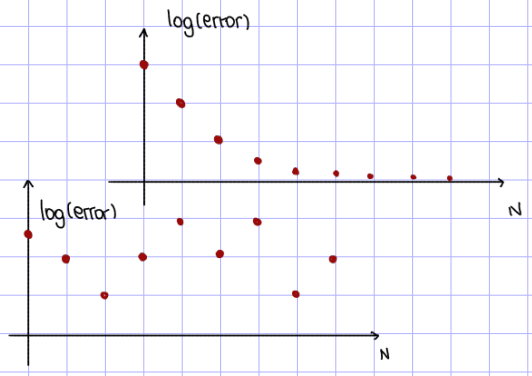
Step foreward...

Let us make a plot of the projection of u_δ in V_N



Where $IP \in \mathbb{R}^{N_\delta \times N}$ and $\| IP^u_{u_\delta} - u_{rb} \|$ is the **mismach** from $IP^u_{u_\delta}$ & u_{rb}

Moreover, focusing on the error, we can plot



POD: via theorems we're sure that the error goes to zero

POD-NN: we do NOT have theorems ⟶ enlargying the space, hence using more N, the error does NOT decrease

↓

The issue is the accuracy: the error cannot go under 0.01

4- LOW FIDELITY MODEL

The idea is to reduce S_δ to solve the solution in a cheaper way => we approximate S_δ using a space $V_N \subseteq V_\delta$ for which $N = \dim(V_N)$, $N_\delta = \dim(V_\delta)$ and $N \ll N_\delta$. Hence, S_N is the approximation of S that is named as **low fidelity model** in contrapposition to S_δ that is high fidelity model → cheaper to solve it

Given the manifold $\mathcal{M}_\delta$ we wish to build an approximation in which is cheaper to evaluate $\forall \mu \in P$
For the approximation of S_δ we imagine to select a subset $\{\mu_1, \dots, \mu_N\} \subset P$ and to compute the **high fidelity solution** we use the so called **snapshots** $\{w_1 = u(\mu_1), \dots, w_N = u(\mu_N)\} \in V_\delta$ that produce a "surrogate" of S_δ that we call **low-fidelity model** S_N or reduced model

Different way to approximate S_δ:
1. Interpolation – not taken in account
2. Least Square – not taken in account
3. Reduces basis Method RBM => Galerkin approach
4. Neural Network: supervisio of unsupervision learning → later

4.1 Reduced basis method RBM

Consider $V_\delta \subset V$, $a(u_\delta, v_\delta; \mu) = F(v_\delta, \mu)$, the subspace of the solution is $\{w_1, \dots, w_N\} \in V_\delta$ that (N different solution) ↳ snapshots could be linear indipendent:

$$\sum_{i=1}^N a_i w_i = 0 \iff \{a_i\} = 0 \quad \forall i \in \{1, \dots, N\}$$

that solutions form a basis, so they build a subspace $V_N \subseteq V_\delta$, $N \in N_\delta$

we formulate a variational problem on V_N:

Find $u_N \in V_N$ st $\forall \mu \in P$, $a(u_N, v_N; \mu) = F(v_N; \mu)$ $\forall v_N \in V_N$ => $a(u_\delta - u_N, v_N) = 0$ $\forall v_N \in V_N$ Galerkin orthogonality in V_N

($\langle u_\delta - u_N, v_N \rangle_a = 0$; $V_N := \text{span}\{w_1, \dots, w_N\} \in V_\delta$)

↳ can be solved because is a bilinear form continuous, coercive, etc

So if the hp holds, it's true $\|u_\delta(\mu) - u_N(\mu)\|_V \leq \frac{\gamma}{\alpha} \inf_{v_N \in V_N} \|u_\delta(\mu) - v_N\|$ => it holds the Cea Lemma

Generation of the linear system. We have to select a basis $\{w_1, \dots, w_N\}$ of V_N and we rewrite the solution in that way to compute the degree of freedom (we express $u_N(\mu)$ in terms of $\{w_i\}_{i=1}^N$)

$$u_N(\mu) = \sum_{i=1}^N u_{N,i}(\mu) w_i \iff u_N(\mu) := [u_{N,1}, \dots, u_{N,N}] \in \mathbb{R}^N$$

array of coefficient

- $S_N : \mu \in P \mapsto u_N(\mu) \in V_N \subset V_\delta$
- $V_N := \{u_N(\mu) \in V_N : \mu \in P\}$

↓ Put that in the bilinear form: $a(u_N, v_N; \mu) = F(v_N; \mu)$ => $\sum_{i=1}^N a(u_{N,i} w_i, v_N; \mu) = F(v_N, \mu)$ $\forall v_N \in V_N$

(we have N unknows, we need N equations)

$$\Rightarrow \sum_{i=1}^N u_{N,i}(\mu)\, a(w_i, w_j; \mu) = F(w_j; \mu) \quad \forall \{w_j\}_{j=1}^N$$

set of linear indipendent vector from which we can derive the solution

In matrix form $u_N \in \mathbb{R}^N$, $A_N^\mu \in \mathbb{R}^{N,N}$, $f_N^\mu \in \mathbb{R}^N$ => $A_N^\mu u_N = f_N^\mu$ Linear system parametric (dimension very low, easy to solve)

NB $A_N^\mu := [a(w_i, w_j; \mu)]_{i,j=1}^N$, $f_N^\mu := [F(w_i, \mu)]_{i=1}^N$ ↳ what we want to compute

↓ In this way we can compute the solution in a cheaper way, with a not to huge error

$N \ll N_\delta$ ($\sim 10^3$ vs ~ 10.000)

priori formula, exponential decay

Exercise – How the matrix A_N is related to A_δ? Consider $A_N^\mu u_N = f_N^\mu$ and $A_\delta^\mu \cdot u_\delta = f_\delta^\mu$

=> From A_N^μ we can deduce $\{w_i\}_{i=1}^N \in V_N \subset V_\delta$ (basis of V_N) and $\{\varphi_j\}_{j=1}^{N_\delta} \in V_\delta$ (basis of V_δ) so $w_i = \sum_{j=1}^{N_\delta} b_{ji} \varphi_j$

To answer, we can use the definition of the matrix A:

$$(A_N^\mu)_{ji} = a(w_i, w_j; \mu) = a\left(\sum_{k=1}^{N_\delta} b_{ki} \varphi_k, \sum_{e=1}^{N_\delta} b_{ej} \varphi_e; \mu\right) = \sum_{k,e=1}^{N_\delta} b_{ki}\, a(\varphi_k, \varphi_e; \mu)\, b_{ej}$$

$(A_\delta^\mu)_{ek}$: depends on the parameters, has to compute each time

NB The computation of A_N^μ requires element of size N_δ, that is NOT good if $N_\delta \gg N$!

In matrix form $A_N^\mu = B^T A_\delta^\mu B$, where $(B)_{ki} = b_{ki}$, so the two matrices are similar

($A_N^\mu \in \mathbb{R}^{N,N}$, $A_\delta^\mu \in \mathbb{R}^{N_\delta, N_\delta}$, $B \in \mathbb{R}^{N_\delta, N}$: do not depends on the parameters)

How to select $\{w_i\}_{i=1}^N$ basis of V_N? Proper Orthogonal Decomposition POD

4.2 How to select the snapshots?

We saw that if we have a set of snapshots $\{w_i\}_{i=1}^N$ linearly indipendent we are able to reduce and solve the $VP_N(\mu)$

BUT, how to detect them? $P_N = \{\mu_1, \dots, \mu_N\} \subseteq P$

Suppose we know $H \le N_S$ snapshots $w_m = u_S(\mu_m) \in V_S$, $m = 1, \dots, H$. They may be linearly dependent $\Rightarrow X_H := \text{span}\{w_m\}_{m=1}^H$ may have dimension $< H$.

The idea is to reduce the unecessary snapshots

↓

4.2 Proper orthogonal decomposition POD

↗ Also know as PRINCIPAL COMPONENT ANALYSIS

→ can be seen from the Singular-Value-Decomposition's point of view

Consider $P \in \mathbb{R}^p$ the parameter space, so the subspace $P_H \subseteq P$ with $|P_H| = H$ is defined as $P_H = \{\mu_1, \dots, \mu_H\} \in P$. Here, we can compare a high fidelity solution $\forall \mu_i \in P_H : w_i = u_S(\mu_i)$ of VPS (solution → for that problem)

So we can build $X_H := \text{span}\{w_i\}_{i=1}^H$ that are **linear dependent**

The main idea is to reduce X_H so that to have an indipendent space. To do that we introduce an operator:

$$\mathcal{C}: X_H \to X_H, \quad v \mapsto \sum_{m=1}^H \langle v, w_m \rangle_V w_m$$

in which w_m is called snapshot and represent a solution

Properties of $\mathcal{C}$.

1. $\mathcal{C}$ is **self-adjoint** $\forall v, z \in X_H \quad \langle \mathcal{C}v, z \rangle_V = \langle \mathcal{C}z, v \rangle_V \Rightarrow \langle \mathcal{C}v, z \rangle_V = \sum_{m=1}^H \langle v, w_m \rangle_V \langle w_m, z \rangle_V = \langle \mathcal{C}z, v \rangle_V$ (eigenfunctions)
2. $\mathcal{C}$ is **semi-definite positive** $\forall v \in X_H : \langle \mathcal{C}v, v \rangle_V = 0$, so $\mathcal{C}$ has H eigenpairs $\{\lambda_m, \psi_m\}_{m=1}^H$ (eigenvalue)
   ↳ so $\{\lambda_m \ge 0\}$, $\lambda_1 \ge \lambda_2 \ge \dots \ge \lambda_H \ge 0$ and $\|\psi_m\| = 1$ $\Rightarrow \langle \mathcal{C}v, v \rangle_V = \sum_{m=1}^H \langle v, w_m \rangle_V^2 \ge 0$ $\quad \mathcal{C}\psi_m = \lambda_m \psi_m \ \forall m = 1, \dots, H$
3. $\mathcal{C}$ is **injective** iff $\{w_m\}_{m=1}^H$ are linearly indipendent

Aim. Find linear indipendent snapshot: we need eigenpair decomposition

↓

Theorem. We define $(C)_{m,n} := \langle w_m, w_n \rangle_V$ and $C \in \mathbb{R}^{H,H}$ that is the covariance matrix

$\Rightarrow$ The eigenvalue $\{\lambda_m\}_{m=1}^M$ of C are the eigenvalue of the operator $\mathcal{C}$

$\Rightarrow$ The eigenfunctions $\psi_m := \frac{1}{\sqrt{\lambda_m}} \sum_{k=1}^H w_k [a_m]_k \ \forall m \in \{1, \dots, m\}$, where a_m are the eigenvectors of C, $[a_m]_k = a_k$

↓

Proof on the lecture notes

More Details on POD on Lecture 4

4.3 Affine case

If the bilinear form $a(\cdot, \cdot; \mu)$ admits an **affine decomposition** s.t. $a(u, v; \mu) = \sum_{q=1}^{Q_a} \theta_a^q(\mu) a_q(u, v)$ where $\forall q \in \{1, \dots, Q_a\}$

$a_q : V \times V \to \mathbb{R}$ and $\theta_a^q : P \to \mathbb{R}$, then

$$A_N^\mu = B^t A^\mu B = B^t \left(\sum_{q=1}^{Q_a} \theta_a^q(\mu) A_q \right) B = \sum_{q=1}^{Q_a} \theta_a^q(\mu) B^t A_q B \in \mathbb{R}^{N \times N}$$

can be computed **once** for all $\mu \in P$ $\Rightarrow$ **off-line phases**

and A_N is assemble **fastly** and **cheaply** each value of μ $\Rightarrow$ **on-line phases**

Similarly, the computation of f_N^μ can be deduced from f^μ with $(f_N^\mu) = F(w_i; \mu) = \sum_{j=1}^{N_S} b_{ji} F(\varphi_j; \mu) = (B^T f^\mu)_i$ and, if $F(\cdot; \mu)$ admits **affine decomposition** i.e

$$F(\cdot; \mu) = \sum_{q=1}^{Q_f} \theta_f^q(\mu) f_q(v)$$

with $F_q : V \to \mathbb{R}$ and $\theta_f^q : P \to \mathbb{R}$ then $f_N^\mu = B^T f^\mu = B^T \sum_{q=1}^{Q_f} \theta_f^q(\mu) f_q = \sum_{q=1}^{Q_f} \theta_f^1(\mu) (B^T f_q) \in \mathbb{R}^N$ and again $B^T f_q$ is indipendent of μ (off-line)

4.4 Error estimation in RBM

Consider the problem $a(u,v) = f(v)$ in the space V, projecting on lower dimension fixing $\mu \in \mathcal{P}$

$a(u_\delta, v_\delta) = f(v_\delta)$ in the space $V_\delta \subset V$ $\forall v_\delta \in V_\delta$ that represents the high fidelity problem

$a(u_N, v_N) = f(v_N)$ in the space $V_N \subset V_\delta$ $\forall v_N \in V_N$ that is the low fidelity problem

Before going into details how to find V_N, we need to say something about the decay of the error $\|u_\delta(\mu) - u_N(\mu)\|_V$

We want to compute the error $\longrightarrow$ $\|u_\delta(\mu) - u_N(\mu)\|_V$

Definition. Consider the finite subset $Y \subset V$ and $w \in V$, the **distance** of an element from a space is

$$\text{dist}_V(w, Y) = \min_{y \in Y} \|w - y\|_V$$

Definition. Given $M \subset V$ finite, the **distance** of M from Y is

$$\text{dist}_V(M, Y) = \sup_{w \in M} \left(\text{dist}_V(w, Y) \right) = \sup_{w \in M} \left(\min_{y \in Y} \|y - w\|_V \right)$$

Definition. Given an integer $N \geq 1$, the **Kolmogorov N-th width** of $M \subset V$ is

$$d_N(M) = \inf_{\substack{Y_N \subset V \\ \dim(Y_N) = N}} \left\{ \text{dist}_V(M, Y_N) \right\} = \inf_{\substack{Y_N \subset V \\ \dim(Y_N) = N}} \left\{ \sup_{w \in M} \left(\min_{y \in Y_N} \|y - w\|_V \right) \right\}$$

Why we need all this definition? With reduced Space method we want to compute $V_N \subset V_\delta$ in the way that the error that I perform is lower as possible, to do that we exploit these definitions

NB
- From the definition, we can say that $d_N(M)$ is the maximum error that we can commit approximating an element of M with element contained in a n-dimensional subspace
- $d_{N+1}(M) \leq d_N(M)$ $\forall N \geq 1$
- If M is compact, $d_N(M) \to 0$, $N \to +\infty$ BUT HOW IS IT FAST?

More detail on Lecture 3

4.5 More info NOT required during the exam

In our case V is the variational set of the weak formulation, and $\mathcal{M} \overset{V_h}{\subseteq} V$ FEM dimension and $Y_N \overset{V_N}{\subset} \mathcal{M}_\delta \subset V$ RB

Suppose that $\mathcal{M}$ is compact and $d_{N+1}(\mathcal{M}) \leq d_N(\mathcal{M})$ and $d_N(\mathcal{M}) \underset{N \to +\infty}{\to} 0$: if we enlarge N, the error dicrease, but we know that we cannot ser $N \to \infty$ because we are in finite dimension space

↓

How fast this error goes to zero?

Property. Consider $\mathcal{M}_\delta := \{u_\delta(\mu) \in V_\delta : \mu \in \mathcal{P}\} \subseteq V_\delta$, where $u_\delta(\mu)$ is a solution of $VP_\delta(\mu)$ in which we have $a(\cdot,\cdot;\mu) = F(\cdot;\mu)$

1. If $a(\cdot,\cdot;\mu)$ and $F(\cdot;\mu)$ are V-differentiable wrt μ $\Rightarrow$ $d_N(\mathcal{M}_\delta) \leq C N^{-r}$
2. If $a(\cdot,\cdot;\mu)$ and $F(\cdot;\mu)$ are analytic wrt $\mu \in \mathcal{P}$ $\Rightarrow$ $d_N(\mathcal{M}_\delta) \leq C e^{-\gamma N}, \gamma > 0$

↓

analytic for $f: D \to \mathbb{R}$ wrt $x_0 \in D$: $\exists a_n \sum_{n=0}^{\infty} a_n (x - x_0)^n = f(x)$ $\forall x \in B_\varepsilon(x_0)$ with $a_n := \frac{f^{(n)}(x_0)}{n!}$

↓ the Taylor expansion

The reduced space converges with exponential decay

This is a property of the weak formulation, hence of the proplem itself, does NOT depend on the discretization

5. GREEDY STRATEGY => with the SVD we're able to chose a reduces space V_N s.t. the average error $\Phi(y_N)$ is minimum

The POD produce a space V_N in which we can minimize the function

$$\Phi(y_N) := \frac{1}{M}\sum_{m=1}^{M} \inf_{v \in V_N} \|w_m - v\|_V^2 \quad \Rightarrow \quad \min \Phi(y_N) := \frac{1}{M}\sum_{m=1}^{M} \inf_{v \in V_N} \|w_m - v\|_V^2$$

Hence, we chose the V_N space st the error on average is the smallest one → This is a way to produce a space, we can find other way

AIM_ We want to find this relation : $\|u_\delta(\mu) - u_N(\mu)\|_V < \varepsilon_{toe}$ → we have to produce a space in this way GREEDY

↳ To do this we need the generation of $N_h > N$ snapshots and the computation of SVD decomposition (expensive), even if it is off-line

↓ for this reason the greedy strategy is implemented => This strategy requires an **a-posteriori** error estimator $\eta = \eta(\mu, V_N)$

5a. Idea of greedy strategy_ We start from $P_h \subseteq P$, we pick one value of the parameter, compute the HF solution and if it is good, this enters in the set

↓ Starting from $V_N = \{\}$

We select $\mu_1 \in P_h$ and we compute $u_\delta(\mu_1)$ ⟶ we add $u_\delta(\mu_1)$ to V_N but to do that we have to modify it a little bit

Indeed, we compute $w_1 := \frac{u_\delta(\mu_1)}{\|u_\delta(\mu_1)\|_V}$, so we have just to normalize it before to put in the basis

↳ computed one each iteration => cheap!

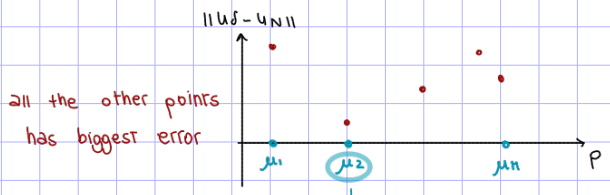
Now we have $V_N := \text{span}\{w_1\}$ and $N = 1$

- Is the space enought ? $\max_{\mu \in P} \|u_\delta(\mu) - u_N(\mu)\|_V < \varepsilon_{toe}$ so all the error has to be under a tolerance
  - yes we have finished
  - No we need an iterative algorithm to enlarge the space : $\mu_{iH} = \underset{\mu \in P_h}{\text{argmax}} \|u_\delta(\mu) - u_N(\mu)\|_V$

So I take the parameter that gives me the largest error

$i = 0$ $V_N = \{w_1 = u(\mu_1)/\|u(\mu_1)\|_V\}$

$i = 1$ $V_N = \{w_1, w_2\}$



Starting from this parameter : here there is the smallest error

I compute the high fidelity solution $u_\delta(\mu_{i+1})$

Hence, I normalize $w_{i+1} := \frac{u_\delta(\mu_{i+1})}{\|u_\delta(\mu_{i+1})\|_V} \perp V_N$ NB Has to be orthogonal to all the other function inside the space V_N to do that we exploit GRAM SCHMIDT

So that $V_{N+1} := \text{span}\{V_N, w_{i+1}\}$

We check again the tolerance

BUT, we have to know all the value of the HF solution? No, we want to compute that in a cheaply way

So we will use a quantity that controls the error η, so that when this quantity $\eta < \varepsilon$ we know also that all our value are under the tolerance => we will use an **a posteriori estimator** that controls the error

$\|u_\delta(\mu) - u_N(\mu)\|_V \leq \eta(\mu, V_N) < \varepsilon_{toe}$ => $\eta(\mu)$ shall be computed for many values of $\mu \in P$, then we can find $\eta(\mu)$ cheaply

↳ is the **reduced basis approximation** of $u_\delta(\mu)$ in the space V_N

Why greedy is better than POD?

1. Controls the maximum error and NOT the average error (as POD)
2. Compute at MOST N solution
3. It is cheapier

5b. A posteriori error computation

We recall the VPN(μ) and we want to find $u_N(\mu) \in V_N$ => $a(u_N(\mu), v_N; \mu) = F(v_N; \mu)$ $\forall v_N \in V_N \subseteq V_\delta$, $\forall \mu \in P$

We introduce the **weak residual** : $R_\delta(u_N(\mu); \mu) \in V_\delta'$ → Stay in the dual of V_δ

$\langle R_\delta(u_N; \mu), v_\delta \rangle_{V_\delta'} := F(v_\delta; \mu) - a(u_N(\mu), v_\delta; \mu)$ $\forall v_\delta \in V_\delta$ from the definition of $\|\cdot\|_{V_\delta'}$, we have

$= v_\delta(u_N(\mu), v_\delta; \mu)$ ↳ This is a sort of bilinear function

$$\|R_\delta(u_N(\mu); \mu)\|_{V_\delta'} := \sup_{v_\delta \in V_\delta} \frac{v_\delta(u_N(\mu), v_\delta; \mu)}{\|v_\delta\|_{V_\delta}} \quad \forall v_\delta \in V_\delta$$

This is the duality pair

Property - It is true that : $\frac{1}{\gamma_\delta(\mu)} \|R_\delta(u_N(\mu);\mu)\|_{V_\delta'} \le \|u_\delta(\mu) - u_N(\mu)\|_V \le \frac{1}{\alpha_\delta(\mu)} \|R_\delta(u_N(\mu);\mu)\|_{V_\delta'}$

(high-fidelity solution; reduce solution; coercivity constant; we are in the discrete formulation of the problem)

Proof - From coercivity we have $\alpha_\delta(\mu)\|u_\delta(\mu) - u_N(\mu)\|_V^2 \le a(u_\delta(\mu) - u_N(\mu); u_\delta(\mu) - u_N(\mu); \mu)$

$= F(u_\delta(\mu) - u_N(\mu); \mu) - a(u_N(\mu), u_\delta(\mu) - u_N(\mu); \mu)$

$= \le R_\delta(u_N(\mu);\mu), u_\delta(\mu) - u_N(\mu)\rangle_{V_\delta} \le \|R_\delta(u_N(\mu);\mu)\|_{V_\delta'} \|u_\delta(\mu) - u_N(\mu)\|_V$

The other direction follow from the continuity hypothesis $\forall v_\delta \in V_\delta$

$r_\delta(u_N(\mu), v_\delta; \mu) := a(u_\delta(\mu), v_\delta; \mu) - a(u_N(\mu), v_\delta; \mu) = a(u_\delta(\mu) - u_N(\mu), v_\delta; \mu) \le \gamma_\delta(\mu) \|u_\delta(\mu) - u_N(\mu)\|_V \|v_\delta\|_V$

Using the definition of $\|R_\delta(u_N(\mu), \mu)\|_{V'}$

Now, using the upper bound of the property, we can derive an **a-posteriori error estimator**. Let's make it computable using the **Riesz' representation theorem** (It is good if it can be computer faster (cheaper!))

Riesz' Representation Theorem - $\exists\, \hat{r}_\delta(u_N(\mu);\mu) \in V_\delta$ (in the original space V_δ) s.t. $\langle \hat{r}_\delta(u_N(\mu);\mu), v_\delta\rangle_V = \le R_\delta(u_N(\mu), \mu, v_\delta\rangle_{V_\delta'}$ (V_δ' the dual space V_δ; Duality pair) $\forall v_\delta \in V_\delta$

and $\|\hat{r}_\delta(u_N(\mu);\mu)\|_V = \|R_\delta(u_N(\mu), \mu)\|_{V_\delta'}$

NB

1. The computation of $\hat{r}_\delta(\mu) := \hat{r}_\delta(u_N(\mu);\mu)$ requires to solve an high-fidelity problem (expensive) However, in the affine case the computation can be efficient (but we can perform offline-online phase)
2. The error a-posteriori estimation $\|u_\delta(\mu) - u_N(\mu)\|_V \le \frac{1}{\alpha(\mu)} \|\hat{r}_\delta(\mu)\|_V := \eta_V(\mu)$ requires also the computation of $\alpha(\mu)$ → we need to know for all the parameter

This can be accomplished with the lower bound estimation $\alpha_{LB}(\mu) \le \alpha(\mu)$ cheap to be computed

There is the possibility to compute also an **a-priori relative error estimator** (To use in algorithm with relative tolerance), observing that

$\|u_N(\mu)\|_V = \|u_\delta(\mu) + u_N(\mu) - u_\delta(\mu)\|_V \le \|u_\delta(\mu)\|_V + \|u_N(\mu) - u_\delta(\mu)\|_V \le \|u_\delta(\mu)\|_V + \eta_V(\mu)$

If we define $\eta_{rel,V}(\mu) := \frac{2\eta_V(\mu)}{\|u_N(\mu)\|_V}$ (Norm of the reduce solution), we obtain

$\|u_N(\mu)\|_V \le \|u_\delta(\mu)\|_V + \frac{1}{2}\|u_N(\mu)\| \eta_{rel,V}(\mu) \le \|u_\delta(\mu)\|_V + \frac{1}{2}\|u_N(\mu)\|_V \Rightarrow \|u_N(\mu)\|_V \le 2\|u_\delta(\mu)\|_V$

$\frac{\|u_\delta(\mu) - u_N(\mu)\|_V}{\|u_\delta(\mu)\|_V} \le \frac{\eta_V(\mu)}{\frac{1}{2}\|u_N(\mu)\|_V}$

Property

1. $\|u_\delta(\mu) - u_N(\mu)\|_V \le \eta_V(\mu) := \frac{\|\hat{r}_\delta(\mu)\|_V}{\alpha(\mu)}$
2. $\frac{\|u_\delta(\mu) - u_N(\mu)\|_V}{\|u_\delta(\mu)\|_V} \le \eta_{rel,V}(\mu) := \frac{1}{2}\frac{\eta_V(\mu)}{\|u_N(\mu)\|_V}$
3. $\text{eff}_V(\mu) := \frac{\eta_V(\mu)}{\|u_\delta(\mu) - u_N(\mu)\|_V}$

$\text{eff}_{rel,V}(\mu) := \frac{\eta_{rel,V}(\mu)\|u_\delta(\mu)\|_V}{\|u_\delta(\mu) - u_N(\mu)\|_V}$

Algorithm 1 Greedy Algorithm for Reduced Basis Construction in POD

Initialization:
Choose a set of training parameters (or a continuous parameter domain) $\mathcal{P}_H$.
Select an initial parameter $\mu_1 \in \mathcal{P}_H$.
Compute the high-fidelity solution $u(\mu_1)$.
Define the initial reduced basis $V_1 = \text{span}\{u(\mu_1)\}$ (or its normalization w_1).
Set $N = 1$.
Define an error tolerance ϵ_{tol}.
while $\max_{\mu\in\mathcal{P}_H} ||u(\mu) - u_N(\mu)||_{E_{tol}} > \epsilon_{tol}$ **do**
  Find the parameter $\mu_{max} \in \mathcal{P}_H$ that maximizes the estimated error: $\mu_{max} = \arg\max_{\mu\in\mathcal{P}_H} ||u(\mu) - u_N(\mu)||_{E_{tol}}$ where $u_N(\mu)$ is the approximated solution in the space V_N.
  Compute the high-fidelity solution for this parameter: $u(\mu_{max})$.
  Orthogonalize $u(\mu_{max})$ with respect to the functions already in V_N using Gram-Schmidt to obtain a new basis function w_{N+1}.
  Extend the reduced basis: $V_{N+1} = \text{span}\{V_N, w_{N+1}\}$.
  Increment the counter: $N = N + 1$.
end while
Result:
The final reduced basis $V_N = \text{span}\{w_1, w_2, ..., w_N\}$.

5. DIFFERENT APPROACH TO COMPUTE THE COERCIVITY FUNCTION $\alpha(\mu)$

Recall the error estimator $\eta(\mu) := \frac{\|\hat{r}_s(\mu)\|_V}{\alpha_s(\mu)}$ → To compute η we need $\alpha_s(\mu)$ for each value of the parameters

$\|\hat{r}_s(\mu)\|_V$ computed cheaply with online / offline phase

$\alpha_s(\mu)$ This part, coercivity function, DIFFICULT TO COMPUTE

Associated to high fidelity problem $\forall \mu \in \mathcal{P}$ $\alpha(\mu) = \inf_{v_s \in V_s} \frac{a(v_s, v_s; \mu)}{\|v_s\|_V^2}$ ← The smallest eigenvalue of the spectral problem

So this is an eigenvalue problem

$\Rightarrow \lambda_{min} = \alpha(\mu)$ coercivity $\quad a(w_s, v_s; \mu) = \lambda (w_s, v_s)_V \quad \forall v_s \in V_s$ smallest eigenvalue

$\lambda_{max} = \gamma(\mu)$ continuity

How we can compute the eigenvalue? We fix the basis $\{\varphi_i\}$ in V_s and we have this linear combination $w_s = \sum_{j=1}^{N_s} w_{s,j} \varphi_j$, so that we obtain the matrix form

$A_s^\mu = \lambda X_s w_s$ ↳ Generalized eigenvalue problem, where $(X_s)_{ij} = \langle \varphi_i, \varphi_j \rangle_V$ is the mass inner product matrix in V is the scalar product matrix, and $w_s = \begin{pmatrix} w_{s,1} \\ w_{s,N_s} \end{pmatrix}$

BUT this is very costly, so this has to be modified

Different approach to compute $\alpha(\mu)$ - Most of this approach approximate $\alpha(\mu)$ from the bottom computing lower-bound $(\mu) \le \alpha(\mu)$

↓ We want now to analize these different approaches.

5.1 Min-θ approach → This method works for the affine case and for the so called parametrically coercive problems

We need * The affinity hypothesis (separate the parameter part to the other)

Hypothesis $a(u, v; \mu) = \sum_{q=1}^{Q_a} \theta_a^q(\mu) a_q(u, v)$

* The positivity hypothesis $\theta_a^q(\mu) \ge 0 \quad \forall \mu \in \mathcal{P}$

* A sort of coercivity $a(u, v) \ge 0 \quad \forall v \in V_s$

So that here we can compute, selecting one parameter $\tilde{\mu} \in \mathcal{P}$, $\alpha(\tilde{\mu})$ ⇒ Let's assume we know $\alpha(\tilde{\mu})$ for one value of $\tilde{\mu} \in \mathcal{P}$

By definition, it holds $\alpha(\mu) := \inf_{v_s \in V_s} \frac{a(v_s, v_s; \mu)}{\|v_s\|_V^2}$

$$= \inf_{v_s \in V_s} \sum_{q=1}^{Q_a} \theta_a^q(\mu) \frac{a_q(v_s, v_s)}{\|v_s\|_V^2} \cdot \frac{\theta_a^q(\tilde{\mu})}{\theta_a^q(\tilde{\mu})} \ge \inf_{v_s \in V_s} \min_{1 \le q \le Q_a} \frac{\theta_a^q(\mu)}{\theta_a^q(\tilde{\mu})} \sum_{q=1}^{Q_a} \theta_a^q(\tilde{\mu}) \frac{a_q(v_s, v_s)}{\|v_s\|_V^2}$$

$$= \min_{1 \le q \le Q_a} \frac{\theta_a^q(\mu)}{\theta_a^q(\tilde{\mu})} \inf_{v_s \in V_s} \frac{a(v_s, v_s; \tilde{\mu})}{\|v_s\|_V^2}$$

$= \frac{a(v_s, v_s, \tilde{\mu})}{\|v_s\|_V^2}$

$= \alpha(\tilde{\mu})$ computed in offline phase

Hence, we obtain $\alpha(\mu) \ge \min_{1 \le q \le Q_a} \frac{\theta_a^q(\mu)}{\theta_a^q(\tilde{\mu})} \alpha(\tilde{\mu}) := \alpha_{LB}(\mu)$ ⇒ lower bound of α

↓ Extending this idea we can build

5.2 Multi-parameter min-θ approach → This case requires to know $I > 1$ values of $\alpha(\tilde{\mu}^i)$, $i \in \{1, .., I\}$, and following the previous analize we can obtain

Select $I \ge 1 \Rightarrow \{\tilde{\mu}^i\}_{i=1,..,I}$ so that we can compute $\{\alpha(\tilde{\mu}^i)\}_{i=1,..,I}$

Hence, $\alpha(\mu) \ge \max_{1 \le i \le I} \left(\min_{1 \le q \le Q_a} \frac{\theta_a^q(\mu)}{\theta_a^q(\tilde{\mu}^i)} \alpha(\tilde{\mu}^i) \right) := \alpha_{LB}(\mu)$

5.3 Successive constraint method SCM : complex, but works very properly → based on the definition of a functional

Idea: build a functional $S : \mathcal{P} \times \mathcal{Y} \to \mathbb{R}$ where $\mathcal{Y} \subseteq \mathbb{R}^{Q_a}$, $y \in \mathcal{Y} := [y_1, .., y_{Q_a}]$ → affine element

$$S(\mu, y) := \sum_{q=1}^{Q_a} \theta_a^q(\mu) y_q$$

To better see the relation between the bilinear form and the functional, we can define the space:

$$Y := \left\{ y \in \mathbb{R}^{Q_a} : \exists \vec{v}_s \in V_s \text{ s.t. } y_q = \frac{a_q(\vec{v}_s, \vec{v}_s)}{\|\vec{v}_s\|^2_{\mathbb{V}}}, q = \{1, .., Q_a\} \right\}$$

Hence, the coercivity can be seen as

$$\alpha(\mu) := \inf_{\vec{v}_s \in V_s} \frac{a(\vec{v}_s, \vec{v}_s; \mu)}{\|\vec{v}_s\|^2_{\mathbb{V}}} = \min_{y \in Y} s(\mu; y) \quad \forall \mu \in \mathcal{P}$$

equivalent problem to avoid to compute the eigenvalue problem

But, instead of solving the minimitation problem in Y, we solve it in Y_{LB} so that the problem is easier to solve

How can we find Y_{LB}? The method suggest to find $Y_{UB} \le Y \le Y_{LB}$, so that we can compute

$$0 < \alpha_{LB}(\mu) \le \alpha_{UB}(\mu) \Rightarrow 0 \le 1 - \frac{\alpha_{LB}(\mu)}{\alpha_{UB}(\mu)} < 1$$

$$\alpha_{LB} \le \alpha \le \alpha_{UB}$$

↳ in a bigger set Y_{LB}, I can achieve a better solution

↳ NB in a smaller set Y_{UB}, I have a bigger constant α_{UB} because α is the solution of the minimization problem, so that in $Y_{UB} \subset Y$ could NOT exist a smaller value of α_{UB}

↳ Measure of the quality of approximation of $\alpha(\mu)$

Moreover, we define $\eta_\alpha(\mu) := 1 - \frac{\alpha_{LB}(\mu)}{\alpha_{UB}(\mu)}$

Define a sequence of sets $\cdots \subseteq Y_{UB}^{n} \subseteq Y \subseteq Y_{LB}^{n} \subseteq Y_{LB}^{n-1} \subseteq \cdots \quad \forall n \ge 1$

Idea: iterative algorithm to improve the value with goal $Y_{UB} \le Y \le Y_{LB}$

Algorithm -

1. Selecting $P_a \subseteq \mathcal{P}$, take $\mu_1 \in P_a$ and exactly compute $\alpha(\mu_1)$ with eigenvalue problem
2. Compute w_s eigenfunction associated to $\alpha(\mu_1)$

   ↳ $a(w_s, \vec{v}_s) = \lambda(w_s, \vec{v}_s)_{\mathbb{V}} \quad \forall \vec{v}_s \in V_s$

$$Y_{UB}^1 = \left\{ y^1 \in \mathbb{R}^{Q_a} : y_q^1 = \frac{a_q(w_s, \vec{v}_s)}{\|\vec{v}_s\|}, \ 1 \le q \le Q_a \right\}$$

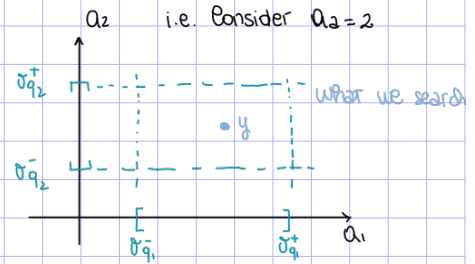
this is an approximation of Y, an upper bound approxim.

3. $\forall q \in \{1, .., Q_a\}$ we have to solve $a_q(w_s, \vec{v}_s) = \lambda(w_s, \vec{v}_s)_{\mathbb{V}} \quad \forall \vec{v}_s \in V_s$ obtaining the eigenvalue
   - $\sigma_q^- :=$ smallest eigenvalue
   - $\sigma_q^+ :=$ largest eigenvalue

$$Y_{LB}^1 := \prod_{q=1}^{Q_a} [\sigma_q^-, \sigma_q^+]$$

because we select $\mu_1 \in \mathcal{P}$

↳ Hence, here we build $Y_B^1 \subseteq Y \subseteq Y_B^1$



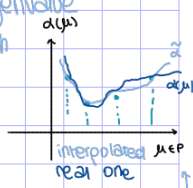
4. We have to improve with the recursion step
   - we have $\{\mu_1, .., \mu_n\} \in P_a \subset \mathcal{P}$
   - we have $Y_{UB}^n \subseteq Y \subseteq Y_{LB}^n$ (with $\alpha_{UB}(\mu)$ and $\alpha_{LB}(\mu)$)
   - we compute $\eta_\alpha(\mu) := 1 - \frac{\alpha_{LB}(\mu)}{\alpha_{UB}(\mu)}$ → Is this lower than a tolerance? $\eta < \varepsilon$
   - If so, we have to enlarge the spaces: select $\mu_{n+1} \in \mathcal{P}$ and compute $\{\alpha(\mu_{n+1}), w_s^{n+1}\}$

$$Y_{UB}^{n+1} := \left\{ y^{n+1} \in \mathbb{R}^{Q_a} : y_q^{n+1} = \frac{a_q(w_s^{n+1}, w_s^{n+1})}{\|w_s\|^2_{\mathbb{V}}} \quad \forall q \in \{1, .., Q_a\} \right\}$$

   - So that $Y_{UB}^{n+1} := Y_{UB}^n \cup Y^{n+1}$
   - $\forall q \in \{1, .., Q_a\}$ solve the eigenvalue problem $a_q(w_s, \vec{v}_s) = \lambda(w_s, \vec{v}_s) \quad \forall \vec{v}_s \in V_s$
   - Compute $B = \prod_{q=1}^{Q_a} [\sigma_q^-, \sigma_q^+]$ → solving the eigenvalue problem computing min e max eigenvalue

$$Y_{LB}^{n+1} := \left\{ y \in B : \sum_{q=1}^{Q_a} \sigma_q^a(\mu) \, y \ge \alpha(\mu) \quad \forall \mu \in \{\mu_1, .., \mu_{n+1}\} \right\}$$

The minimitation is a linear programming minimitation problem with at most $2Q_a + (n+1)$ constraints for Q_a variable



5. Interpolation approach ⟶ The idea is to interpolate the function $\alpha(\mu)$ with $\underline{\alpha}$ over a subset $P_I \subseteq \mathcal{P}$

Consider $\underline{\alpha} : \mathcal{P} \to \mathbb{R}$, $\mu \mapsto \alpha(\mu)$, we can select $P_I \subset \mathcal{P}$ and $\{\mu_i\}_{i=1}^{I} \subset P_I$ we can compute the real value of $\alpha(\mu_i)$ (I as small as possible)

After computing all the exact value, we interpolate the point to compute our $\alpha(\mu)$

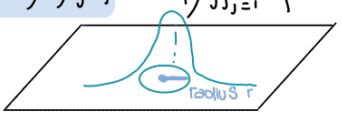
But interpolation in huge space is not so good

=> CURSE of DIMENSIONALITY: to avoid we use (RBF) (exponential function) not polynomial one

↳ We cannot use the "classic" polynomial interpolation, as it suffers of the curse of dimensionality, i.e. the number of interpolant points is too large when $\mathcal{P}$ increase

=> To AVOID this we use radial basis function

In particular, we seek for the logarithm of $\alpha_1(\mu)$, s.t. $\log(\alpha_1(\mu)) = w_0 + w^T\mu + \sum_{i=0}^{N_1} \gamma_i \phi_i(\|\mu - \mu_j\|)$ in $\{\mu_j\}_{j=1}^{N_1}$ points belong to P s.t. $\log(\alpha_1(\mu_j)) = \log(\alpha(\mu_j))$ and s.t.
$\sum_{i=0}^{N_1} \gamma_i = 0$, $\sum_{i=0}^{N_1} \gamma_i \mu_i^p = 0 \ \forall p \in [1,\dots,P]$ where $\varphi: P \to \mathbb{R}$, $r \mapsto e^{-r^2}$ is the RBF, i.e. the gaussian



Algorithm 1 Successive Constraint Method (SCM)

1: ***Idea.*** *Build a functional* $S(\mu, y)$ *where* $y \in Y : S(\mu, y) = \sum_{q=1}^{Q_a} \Theta_a^q(\mu) y_p$

2: *Define the* ***space***

$$Y := \{y \in \mathbb{R}^{Q_a} : \exists v_\delta \in V_\delta \ s.t. \ y_q = \frac{a_q(v_\delta, v_\delta)}{\|v_\delta\|_V^2}, \ q = \{1, \dots, Q_a\}\}$$

3: *The* ***coercivity*** *is*

$$\alpha(\mu) \geq \inf_{v_\delta \in V_\delta} \frac{a(v_\delta, v_\delta; \mu)}{\|v_\delta\|_V^2}, \forall \mu \in P$$

4: ***Equivalent problem.*** *Instead of solving in* Y*, we solve it in a cheaper way in* Y_{UB} *and* Y_{LB}

5: ***Goal.*** *Find* $\alpha_{LB} \in Y_{LB}$ *and* $\alpha_{UB} \in Y_{UB}$ *to bound the solution* $\alpha \in Y$*. using an* ***iterative algorithm***

6: ***Quality measure.*** $\eta_\alpha = 1 - \frac{\alpha_{LB}(\mu)}{\alpha_{UB}(\mu)}$

7: **Input**.

- Set of candidate functions P_a
- Tolerance ϵ

8: **Initialize**. Take $\mu_1 \in P_a$ and exactly compute $\alpha(\mu_1)$ by solving the eigenvalue problem

9: Compute w_δ^1 eigenfunction associated to $\alpha(\mu_1)$

10: Define

$$Y_{UB}^1 = \{y^1 \in \mathbb{R}^{Q_a} : y_q^1 = \frac{a_q(w_\delta^1, v_\delta^1)}{\|v_\delta\|}, 1 \leq q \leq Q_a\}$$

11: $\forall q \in \{1, \dots, Q_a\}$ solve the eigenvalue problem $\forall v_\delta \in V_\delta$

$$a_q(w_\delta, v_\delta) = \lambda(w_\delta, v_\delta)_V$$

obtaining $\theta_q^- :=$ the smallest eigenvalue and $\theta_q^+ :=$ the biggest eigenvalue

12: Build $Y_{LB}^1 = \prod_{q=1}^{Q_a} [\theta_q^-, \theta_q^+]$

13: **while** $\eta_\alpha(\mu) > \epsilon$ **do**

14: Compute $\eta_\alpha(\mu) = 1 - \frac{\alpha_{LB}(\mu)}{\alpha_{UB}(\mu)}$

15: ***Enlarge*** *the space.* Select $\mu_{n+1} \in P_a$ and compute $\alpha(\mu_{n+1}; w_\delta^{n+1})$

16: Define

$$Y_{UB}^{n+1} = \{y^{n+1} \in \mathbb{R}^{Q_a} : y_q^{n+1} = \frac{a_Q(w_\delta^{n+1}, w_\delta^{n+1})}{\|w_\delta\|_V^2} \ \forall q \in 1, \dots, Q_a\}$$

$\to Y_{UB}^{n+1} = Y_{UB}^n \cup Y^{n+1}$

17: $\forall q \in \{1, \dots, Q_a\}$ solve the eigenvalue problem $\forall v_\delta \in V_\delta$

$$a_q(w_\delta, v_\delta) = \lambda(w_\delta, v_\delta)_V$$

obtaining $\theta_q^- :=$ the smallest eigenvalue and $\theta_q^+ :=$ the biggest eigenvalue

18: Build $B = \prod_{q=1}^{Q_a} [\theta_q^-, \theta_q^+]$

19:

$$Y_{LB}^{n+1} = \{y^{n+1} \in B : \sum_{q=1}^{Q_a} \Theta_q^a(\mu) y \geq \alpha(\mu) \ \forall \mu \in \{\mu_1, \dots, \mu_{n+1}\}$$

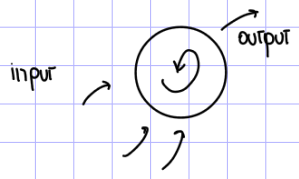
20: **end while**

21: **Output**. Approximate solution α

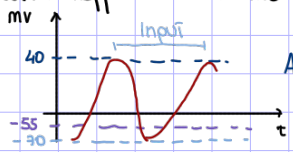
7. NEURAL NETWORKS : DEEP LEARNING → We're going to talk about deep learning based on the artificial neural network, a model inspired by the biological network of the human brain

Terminology -
1. AI : artificial intelligence → simulation of the human intelligence in a machine
2. ML : machine learning → the machine learn something by NOT give it directly the instruction
3. DL : deep learning → the machine learn by using neural networks

Biological model of NEURON



The IDEA is to simulate the neural inside our head
There are many inputs that are called **dendrides** an there is only one ourputs that is the **axon** → High parallel processed
What happen inside the neuron? The voltage (eletrical) increases a lot ↘



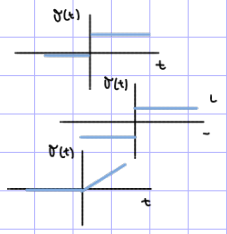
Eletricity in the neuron — at each input the potential increase
Action potential: that is the activation 3.
This is the threshold potential: the neuron activates 2.
Membrane potential at rest: the neuron is "Sleeping" 1.

7.1 What is the first model of NN? PERCEPTON → This is the first mathematical surrogate

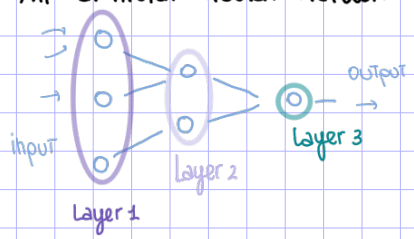
↳ We have different input x_i that represent the neuron (binari : 0-1)
In the middle there is the γ : activation function that takes a linear combination of x_i
$\gamma(w_1x_1 + \dots + w_nx_n + w_0)$ ← weight of the rest potential
The ourput is a binary one : active (1) or not active (0)
All the weights are collected in an array $w = (w_1, \dots, w_n)$, BUT the w_0 is called bias

Example of activation function — usually NON-LINEAR

1. Step function $\gamma(t) = \begin{cases} 0 & t \le 0 \\ 1 & t > 0 \end{cases}$ — discontinue
2. Sign function $\gamma(t) = \begin{cases} -1 & t \le 0 \\ 1 & t > 0 \end{cases}$
3. ReLU function $\gamma(t) = \max(0,t)$ — continuous



An artificial neural network is a connection of different perceptons collected in different layers



7.2 A little bit of history

- 1943: Description of the neural model → Pitts & McCulloch propose the first and simplest mathematical model of a neuron
- 1949: Study the «activaction» memory → Hebb proposed a mathematical formula to describe the learning process (memory of activation)
- 1957: Presentation of the perceptron model → Rosenbatt — Canpr simulate xor function
- 1969: Study the limitation of the perceptron → STOP using NN because there are limitation in the mathematical model ↘ Hweky & Papert
- 1986: Proposal - back propagation theory → STUDY START AGAIN. Rumel Hart
- 1990: Unsupervised learning → Kohonen
- 1990 - 2000: Study about the universal approximation theorems → mathematical results

=> Our purpose is to use NN to solve PDE in a faster way

Notation -
- y := output, generated by the activation function
- γ := activation function
- x := input
- w := weight and w_0 := bias

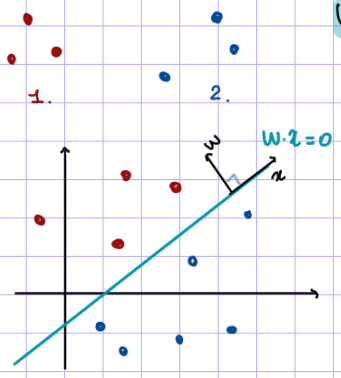
$y = \gamma(\tilde{w}\cdot\tilde{x} + w_0) = \gamma(w \cdot x)$ where $w = (w_0, \tilde{w}) \in \mathbb{R}^{n+1}$, $x = (1, \tilde{x}) \in \mathbb{R}^{n+1}$
↓
$y = \gamma(w_1x_1 + \dots + w_nx_n + w_0)$

How we can use perceptron to CLASSIFY data?

Imagine to have two class and we want to associated to the ≠ element of the class the corresponding layer
(randomly distributed in the space)

PURPOSE: I want that a machine, given an input, tells me if it is blue or red, so the belonging class of the element

We want to performe the line that separate the two classes



$w \cdot x = 0$ hyperplane in higher dimension

When I found w, given the input x I can have as answer the label
$\Rightarrow$ I create the classifier $\longrightarrow$ Let's use $\sigma(t) = \begin{cases} 1 & t>0 \\ 0 & t\leq 0 \end{cases}$

How to build the CLASSIFIER? Hence, I we can find the weights? **TRAINING PHASE**

Consider to have a known-sample $(x^n, y^n)_{n=1,\dots,N}$

The **goal** is to find w

We can build the functional $E(w) = \frac{1}{N}\sum_{n=1}^{N} |y_n - \sigma(w \cdot x^n)|^2 \geq 0$

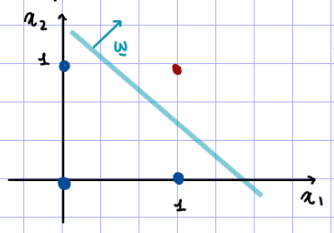
LOSS FUNCTION: measures the mismatch between the data that I know and the one product by my classifier

Hence, we want $w := \underset{w \in \mathbb{R}^{n+1}}{\arg\min}\ E(w)$ so to find the minimization of the LOSS function

this is why NN is difficult: we have to minimize in higher space

ISSUE: no globale minimum in the majority of the situation

Example. We want to approximate with the NN the **AND function** (logical operator)

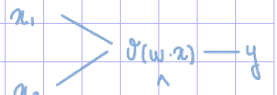


We're looking for a line
But it's evident that we have different solutions

x_1	x_2	$x_1 \wedge x_2$	
0	0	0	y_1
0	1	0	y_2
1	0	0	y_3
1	1	1	y_4

We can build the LOSS function: $E(w) = \frac{1}{N}\sum_{n=1}^{N} |y_n - \sigma(w \cdot x^n)|^2 \longrightarrow N=4,\ w=(w_0,w_1,w_2)\ \ x=(1,x_1,x_2)$

$$E(w) = \frac{1}{4}\left(|0-\sigma(w_0+0+0)|^2 + |0-\sigma(w_0+0+w_2)|^2 + |0-\sigma(w_0+w_1+0)|^2 + |1-\sigma(w_0+w_1+w_2)|^2\right)$$



We decide as activation function the step function: $\sigma(t) = \begin{cases} 0 & t\leq 0 \\ 1 & t>0 \end{cases}$

Hence, to solve the problem $w := \underset{w \in \mathbb{R}^N}{\arg\min}\ E(w)$ we have to solve the linear system

$$\begin{cases} |0-\sigma(w_0)|^2 = 0 \\ |0-\sigma(w_0+w_2)|^2 = 0 \\ |0-\sigma(w_0+w_1)|^2 = 0 \\ |0-\sigma(w_0+w_1+w_2)|^2 = 0 \end{cases} \Rightarrow \begin{cases} w_0 \leq 0 \\ w_0+w_2 \leq 0 \\ w_0+w_1 \leq 0 \\ w_0+w_1+w_2 > 0 \end{cases}$$

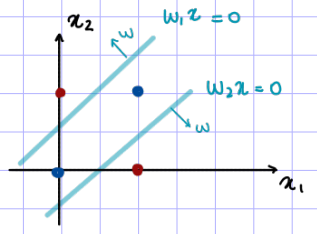
Taking in account $\sigma(t)$

This is the linear system that we have to solve to find the green line
Moreover, there are ≠ solutions, i.e. $w=\left(-\frac{2}{3},\frac{1}{2},\frac{1}{2}\right)$ & $w=(-1,1,1)$

7.4 Feed forward neural network ⟶ We need at least two perceptrons

Limitation - XOR operator.

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

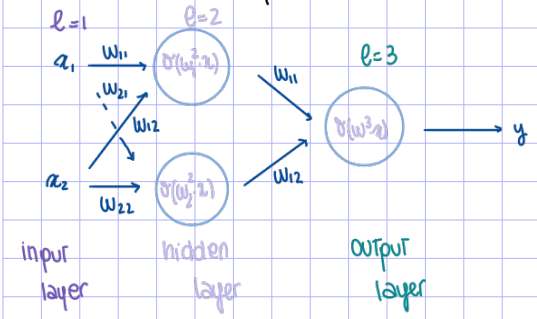


BUT using a line we cannot separate the point that XOR generate ⟶ one perceptron fails

↓

SOLUTION: If we use two line (two perceptron) we can face this limitation

Now we can represent the NN of the XOR

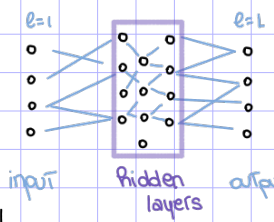


So we have different layer and 3 perceptron to have the solution and we build feed forward NN in which we can use ≠ σ in different layer ⟶ This is difficult to solve!
BUT in each layer, to solve fastly and parallelite the computation, we usually use the same activation function

For more details on the generalization of FF-NN see the lecture note's lesson 7

8. HOW TO COMPUTE THE WEIGHT?

8.1 FFNN recall



All the layer are fully connected with σ_ℓ the activation function of the layer ℓ

$$y_i^\ell = \sum_{j=1}^{N_{\ell-1}} \sigma_\ell\left(w_{ij}^\ell \, x_j^{\ell-1}\right) \quad \forall i \in (1,..,N_\ell) \quad \forall \ell \in (2,..,L) \quad (*)$$

(y_i^ℓ: neuron i of layer ℓ; y := output; w_{ij}^ℓ: weight; $x_j^{\ell-1}$: input before layer; σ_ℓ: non linear)

$\Rightarrow$ The non-linearity is contained in σ_ℓ

↓ Rewrite in compact form $y^\ell = f_\ell(w^\ell, y^{\ell-1})$

$y^{\ell-1} \equiv x^{\ell-1}$ in which we consider the input of layer j is the output of layer $j-1$

Imagine to know the weight w (you already solve the optimization problem), the algorithm to the evaluation process is

ALGORITHM to compute y

$y^1 = x$ ⟶ input
for $\ell = 2,..,L$
  $z^\ell = w^\ell y^{\ell-1}$
  $y^\ell = \sigma^\ell(z^\ell)$
end
$y = y^L$ ⟶ output

↳ The optimization problem is the offline phase that is NOT fast

↳ This is similar to the online phase that is very fast

How to compute the weight? TRAINING PHASE: we have different strategies

1. **Supervised Learning**: we have known snapshot ⟶ $\{x^n, y^n\}_{n=1,..,N}$ that minimize the loss (x: input, n: layer)
   ↳ the network is trained using example of input & output given by a teacher $\Rightarrow$ **goal**: learn the rule to map $x \mapsto y$
2. **Unsupervised learning**: we do NOT have snapshot ⟶ PINN: Physics Informed Neural Network → See next lessons
   ↳ the network is trained using only inputs: no outputs are given $\Rightarrow$ **goal**: discover hidden patterns in the data
3. **Reinforcement Learning**: You give information and reward and the net learn
   ↳ the network is trained in a dynamic environment to perform a fixed goal: while training, the environment gives feedback (positive & negative reward) and the network try to maximizes

8.2 Supervised learning: we want to approximate the target function

Consider the function $\underline{f}: K \subset \mathbb{R}^n \longrightarrow \mathbb{R}^m$ that is the function in the hidden layer that produce the output

($\underline{f}$: unknown; K: compact; $x \longmapsto y$)

We know N "samples": $(x^n, y^n = \underline{f}(x^n))$

Problem - Given FFNN fixed $(L, N_\ell \; \forall \ell = 2,..,L)$
Find w^ℓ s.t.
$y = f(w^\ell, x)$ → where f is an approximation of $\underline{f}$

IDEA: we want to find w so the f approximate $\underline{f}$ (the hidden function) that we DO NOT have, but we have the model (FFNN) so we know (*)

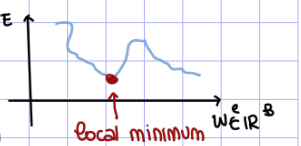
We suppose to know some snapshots $(x^n, \bar{y}^n)_{n=1,..,N}$ and we select the **loss function** n-th loss error

$$E_n = \frac{1}{2}\| y^n - \bar{y}^n \|_2^2 = \frac{1}{2}\| f(w^\ell, x^n) - \bar{y}^n \|_2^2$$

(predicted value; real value)

↓ The **total loss error** is $E = \frac{1}{N}\sum_{n=1}^{N} E_n(w)$ and we want to find a **local minimum**

we try to get a minimization problem

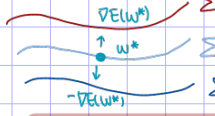


could be NOT unique

We need a **minimization method** ⟶ to solve the minimization problem

a. **GRADIENT DESCENT METHOD**: based on the gradient of the loss function

Find $\min_{w \in \mathbb{R}^B} E(w)$ where $\Sigma'_a := \{w \in \mathbb{R}^B : E(w) = a\}$ $\forall a \in \mathbb{R}$ are the level curve



To go in a curve with lower value we use the gradient $\Rightarrow$ **iterative step**: $w_i = w^{i-1} - \alpha \nabla E(w^{i-1})$ $\alpha > 0 \in \mathbb{R}^+$ (learning rate)

BUT this works well iif the dimension is NOT so high, because we have $\nabla E(w) = \sum_{n=1}^{N} \nabla E_n(w)$ so it depends on N := number of snapshots that has to be high to learn properly

↓

b. Solution? **STOCHASTIC GRADIENT DESCENT ALGORITHM**: compute the gradient for only some snapshots (↳ Garantees only to reach a local minimum, not a global one)

$$w_i = w_{i-1} - \alpha \frac{1}{M}\sum_{n \in S_M} \nabla E_m(w_{i-1}) \qquad M \ll N$$

↓ take $M \neq$ snapshots, chosen in a random way, S_M is called **mini-batch**

Example -

•) Real gradient $\nabla E = \sum_{n=1}^{N} \nabla E_n$

•) Stocastic gradient set $S_M = \{n_1, \dots, n_M\}$ $n_i \in \{1, \dots, N\}$ $\forall i \in \{1, \dots, M\}$ $\quad \tilde{\nabla E} = \frac{1}{M} \sum_{n \in S_M} \nabla E_n$

mini-batch: selected randomly at each epoc, is a subset of all the snapshots

Each iteration of the stochastic gradient descent is called **epoch**

↓

ADAM ALGORITHM (Adaptive Moment Estimation): improvement of the STOCHASTIC in which exploit the previous computation

all the previous gradient are called momentum of E

This is the standard used

L-BGFS: Second order algorithm → exploit the Hessian (estimate), so it is more precise ⟶ following quasi-Newton method

↳ Limited Memory Broyden - Fletcher - Goldfarb - Shanno

c. How to compute $\nabla E(w)$? **BACKPROPAGATION** ⟶ Yann le Cun 1988

Consider $E(w) = \frac{1}{M} \sum_{m=1}^{M} \frac{1}{2} \| y^m(w) - \bar{y}^m \|_2^2$, where i.e. $w = \begin{pmatrix} w_{11}^1 \\ w_{12}^1 \\ \vdots \\ w_{24}^1 \end{pmatrix}$ if $\in \mathbb{R}^{3,2}$ W^1 ⟶ W^2 $\in \mathbb{R}^{2,4}$ perceptron

From definition, we have $\nabla E_n(w) = \frac{1}{2} \cdot 2 \left(y^n(w) - \bar{y}^n \right) \frac{\partial y^n(w)}{\partial w}$, where $y^n(w) = f(w, x^n)$

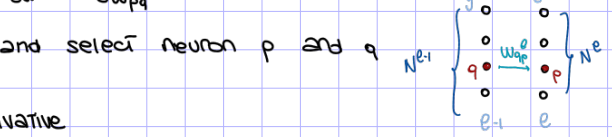
($\bar{y}^n$: we DO NOT know; $\frac{\partial y^n(w)}{\partial w}$: we know)

HOW TO COMPUTE THAT?

we know everything, but not the partial derivative $f: \mathbb{R}^S \to \mathbb{R}^m$ $\quad \frac{\partial f}{\partial w} = \left\{ \frac{\partial f_j}{\partial w_{pq}^e} \right\}$ ⟶ we need to compute the Jacobian

Imagine to fix the layers $e-1, e$ that are fully connected, and select neuron p and q

$y^e = \sigma_e(w^e \cdot y^{e-1})$



So we can use the chain rule to compute the partial derivative

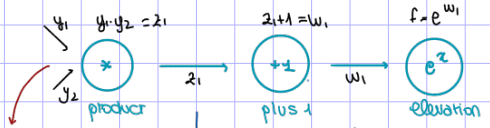
$$\frac{\partial f_j}{\partial w_{pq}^e} = \frac{\partial f_j}{\partial y_p^e} \cdot \frac{\partial y_p^e}{\partial w_{pq}^e}$$

↳ because I derive wrt w_{pq}^e so all the other element becomes ϕ

↳ that is $\sigma'_e(w^e y^{e-1}) \cdot y_q^{e-1}$ and we can compute because we know everything

We DO NOT KNOW i.e. if $f_j = e^{(y_1 y_2 + 1)} \Rightarrow \frac{\partial f_j}{\partial y_1} = e^{y_1 y_2 + 1} \cdot y_2 \quad \frac{\partial f_j}{\partial y_2} = e^{y_2 y_1 + 1} \cdot y_1$

BUT How to compute that in NN logic?



$f = e^{y_1 y_2 + 1} \longrightarrow \frac{\partial f}{\partial f} = 1$ where $\frac{\partial f}{\partial w_1} = e^{w_1}$, $\frac{\partial w_1}{\partial z_1} = 1$, $\frac{\partial z_1}{\partial y_1} = y_2$

chain rule $\frac{\partial f}{\partial y_1} = \frac{\partial f}{\partial z_1} \frac{\partial z_1}{\partial y_1}$

chain rule $\frac{\partial f}{\partial z_1} = \frac{\partial f}{\partial w_1} \frac{\partial w_1}{\partial z_1}$

chain rule $\frac{\partial f}{\partial w_1} = \frac{\partial f}{\partial f} \frac{\partial f}{\partial w_1}$

Hence, go back we have all the element and we can compute everything

$$\frac{\partial f}{\partial y_1} = \frac{\partial z_1}{\partial y_1} \cdot \frac{\partial w_1}{\partial z_1} \cdot \frac{\partial f}{\partial w_1} \cdot \frac{\partial f}{\partial f} = y_2 e^{y_1 y_2 + 1}$$

Starting to the end, we know everything so we can compute the derivative!

Now, we can apply this IDEA to compute $\frac{\partial f_j}{\partial y_p^e} = \frac{\partial f_j}{\partial y_h^{e+1}} \cdot \frac{\partial y_h^{e+1}}{\partial y_p^e}$, where by definition $y_p^{e+1} = \sigma_{e+1} \left(\sum_{q=1}^{N_e} w_{pq}^{e+1} y_q^e \right)$

$= f(w, x^n) = y^n$

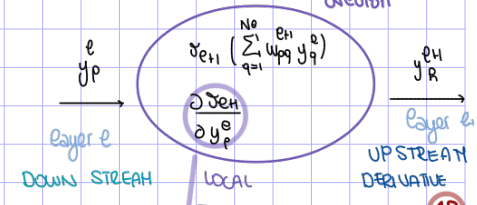
$\frac{\partial y_h^{e+1}}{\partial y_p^e} = \sigma'_{e+1}(w^{e+1} y^e) w_{hp}^{e+1}$

BUT! Know we do NOT know this element

We can put in the case $e+1 = L \Rightarrow \frac{\partial f}{\partial y^L} = \frac{\partial y^L}{\partial y^L} = I$ identity

So we can do all the steps: $\frac{\partial f}{\partial y^{L-1}} \to \frac{\partial f}{\partial y^{L-2}} \to \dots \to \frac{\partial f}{\partial y'} \to \frac{\partial y'}{\partial x}$ $\Rightarrow$ zoom

And $\frac{\partial y'}{\partial x} = \frac{\partial}{\partial x_j} \left(\sigma_i(w^i \cdot x) \right) = \sigma_i'(w^i \cdot x) w_{ij}^1$



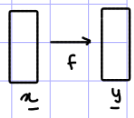
$\frac{\partial f_j}{\partial y_p^e} = \frac{\partial f_j}{\partial y_R^{e+1}} \frac{\partial y_R^{e+1}}{\partial y_p^e}$ because $y^{e+1} = \sigma^{e+1}(\dots)$

To see more detail on UNIVERSAL APPROXIMATION THEORMS see lecture notes Lesson 8

9. CONVOLUTIONAL NEURAL NETWORK

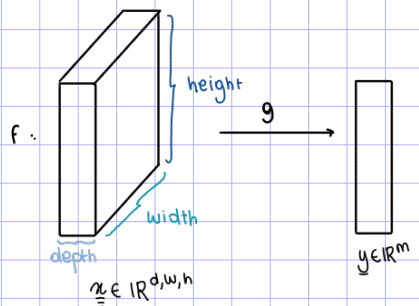
9.1 Increase dimensional input of FFNN

In feed forewarded neural network we can map $\mathbb{R}^n \longrightarrow \mathbb{R}^m$ with the non-linear function $f: \mathbb{R}^n \to \mathbb{R}^m$, $\underline{x} \mapsto \underline{y} = f(\underline{x})$



How we can generalite this kind of NN so that we have an input of bigger dimension, i.e. 3 dimensional input?

Convolutional neural network CNN. This NN is tipically used to process image



We want to add a new function so that the 3d input can transform, via g, in a long array so that can be the input of the FFNN to find the output

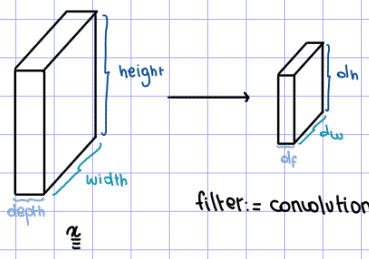
↑ the result

Problem. How to pass from a 3dimensional array to 1dimensional?

Solutions. Ingredients of g ⟶
a. Convolutional layer
b. Pooling layer
c. Normalitation layer

NB Convolutional NN extend the idea of FFNN but mantaining the structure of the original data

9.2 Convolutional layer ⟶ is a filter



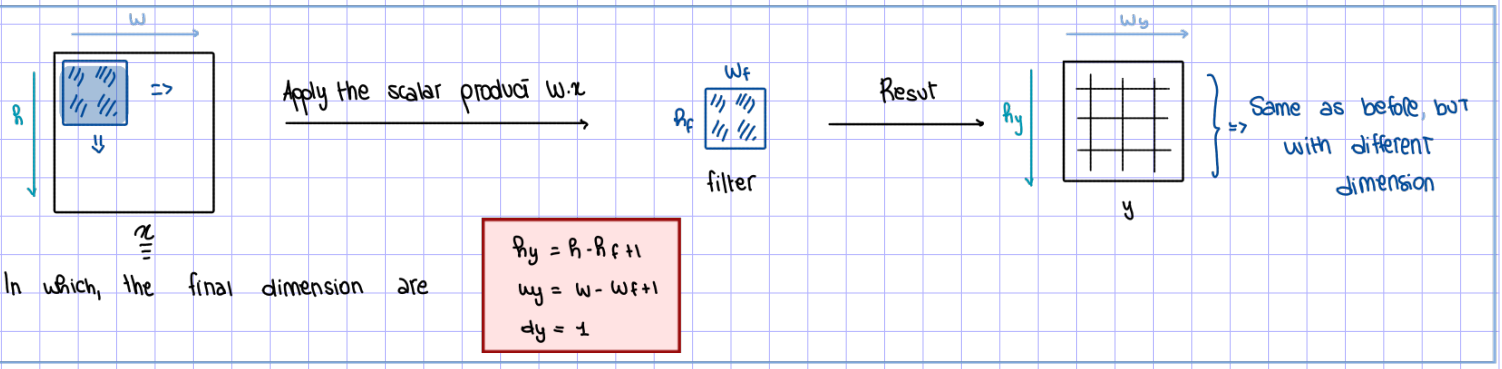
filter:= convolutional layer

Consider $d_f = d$, $\underline{W} :=$ the weight $\in \mathbb{R}^{d_f \times w_f \times h_f}$ and we want to build a filter so that

$$\underline{y} = \underline{W} \cdot \underline{x} + w_0$$

, where w_0 is the bias and W the associated matrix

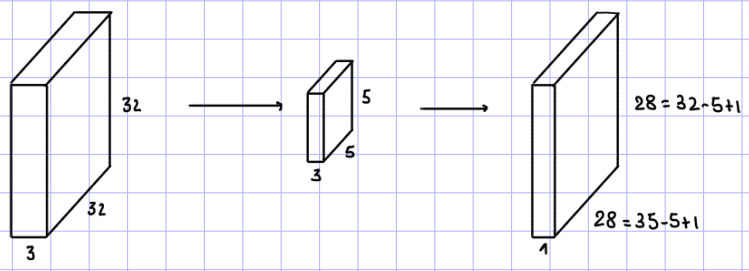
The idea is that the filter can move left-right & up-down on the image
↳ We shift the convolutional filter around $\underline{x}$ doing convolution $y = Wx + w_0$



In which, the final dimension are

$$h_y = h - h_f + 1$$
$$w_y = w - w_f + 1$$
$$d_y = 1$$

Example.

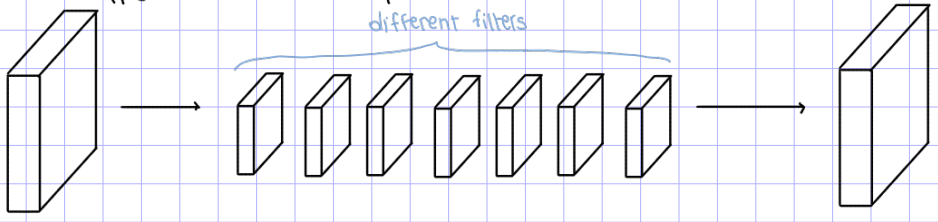


NB In general, we can use more than one convolutional filter, obtaining different filters inputs

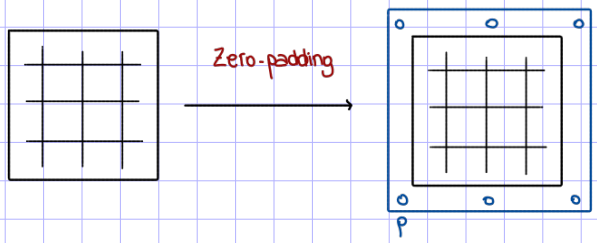
Observation. After the convolutional layer you have reduced the dimension of the image, this caused some issue.

↓ Let's now try to solve them

a. **Channel dimension.** The depht is $d_y = 1$. How we can deal with this?
We can apply different filter in parallel in which I can choose the dimension d called channel to use for all the layer

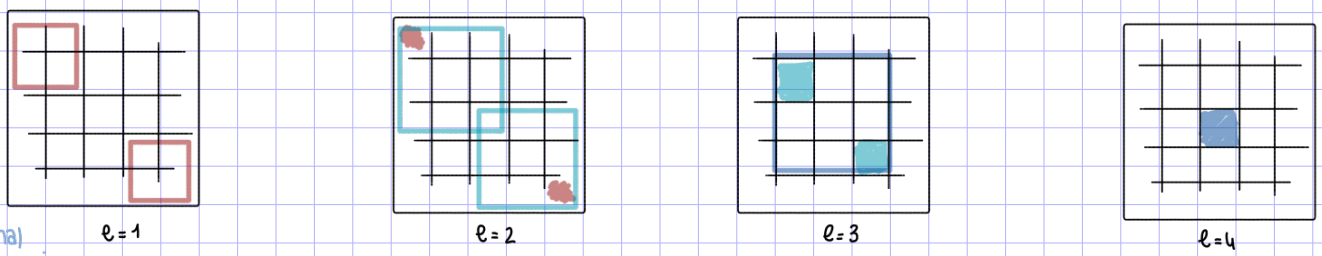


b. **Padding.** If we apply K convolution one after the other it happens that the dimension of $\underline{x}$ goes to zero rapidly (Shrink)
How we can deal with this? We generalite the convolutional layer with the operation of padding, to avoid constraints on the number of layer in the NN



$$W_y = W - W_f + 1 + 2p$$
$$R_y = R - R_f + 1 + 2p$$

c. **Receptive fields.** With convolutional layer the next data is influenced by a particular region of the previous data and by a particular region of the input
Let's now imagine to apply the filter to my input.
How many layer we need to see all the information of the input?

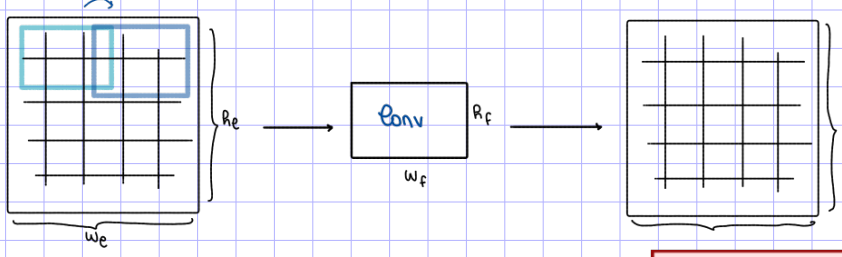


Convolutional layer:

In this simple example, we need 4 layer, think about bigger one!

↓

We add the last parameter to move it "faster" in the image: **Stride**



$$R_{e+1} = \frac{(R_e - R_f + 2p)}{s} + 1$$

$$W_{e+1} = \frac{(W_e - W_f + 2p)}{s} + 1$$

Example. Consider

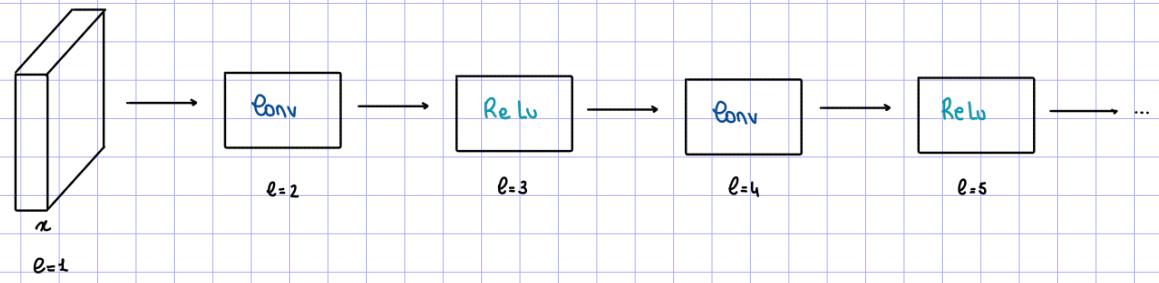
Conv d W_f h_f

- $\underline{x}^e$: $3 \times 32 \times 32$ (d w R) is a convolutional filter where W: $10 \times 3 \times 5 \times 5$ with $s=1$, $p=2$ ⟶ $W = 760$ unknows to minimize
- $\underline{x}^{e+1}$: $10 \times 32 \times 32$

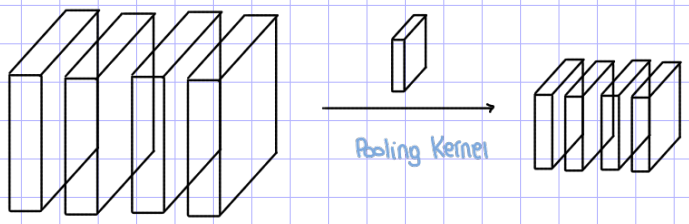
The number of operations to do are $\simeq$ 1 milion to perform on each convolutional layer ⟶ done in parallel via GPU

Definitions - • **Learnable parameters**: weights inside the filter for each channel w_{ij} ⟶ learn by the minimization problem
• **Hyperparameters**: not learnable parameters ⟶ fixed by the user before the learning phase, i.e. padding, stride

Till now, we have NOT used a non-linearity ...
... Hence, we add the activation function, i.e. ReLu to add non-linearity



9.3 Pooling layers ⟶ the idea is to strinck the input

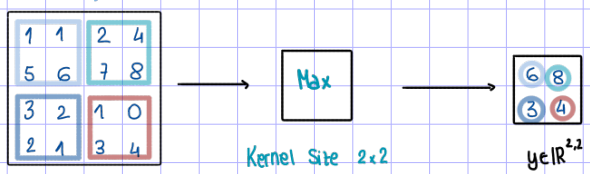


The **pooling Kernels** are just function i.e.
- Avg pool $K=2, S=2$
- Max pool $K=2, S=2$
- AlexNet pool $K=3, S=2$

↓
are used to downsize the sample without any learnable parameters

Let's make an example to better understand

Max pooling



$x \in \mathbb{R}^{4,4}$
Stride = 2
$y \in \mathbb{R}^{2,2}$

The pooling, here, just select one element, i.e. the maximum
↓
we apply non linear function use just to reduce the dimension
NB It's NOT mandatory that the pooling layer is NON-linear

↓

With 9.2 I enlarge the numer of channel, mantaining w_f and b_f applying convolutional layer, activation σ
After with 9.3 I modify the w_f and b_f in which I enlarge the number of channel in output
Till we can transform the dimension in an array so that we can apply FFNN

9.4 Normalitation ⟶ Important to find the right solution, used especially to emprove the speed of the computation
Consider to minimite $J(z,y) = \|z-\hat{z}\|^2 + \|y-\hat{y}\|^2$ ⟶ if y is bigger, you focus on it to emprove
But we do not want to focus only in one variable ⇒ we normalite so that z & y has the same order of magnitude

$$\hat{z} = \frac{z - E(z)}{\sqrt{Var(z)}}$$

so that we have $E(z)=0$ & $Var(z)=1$

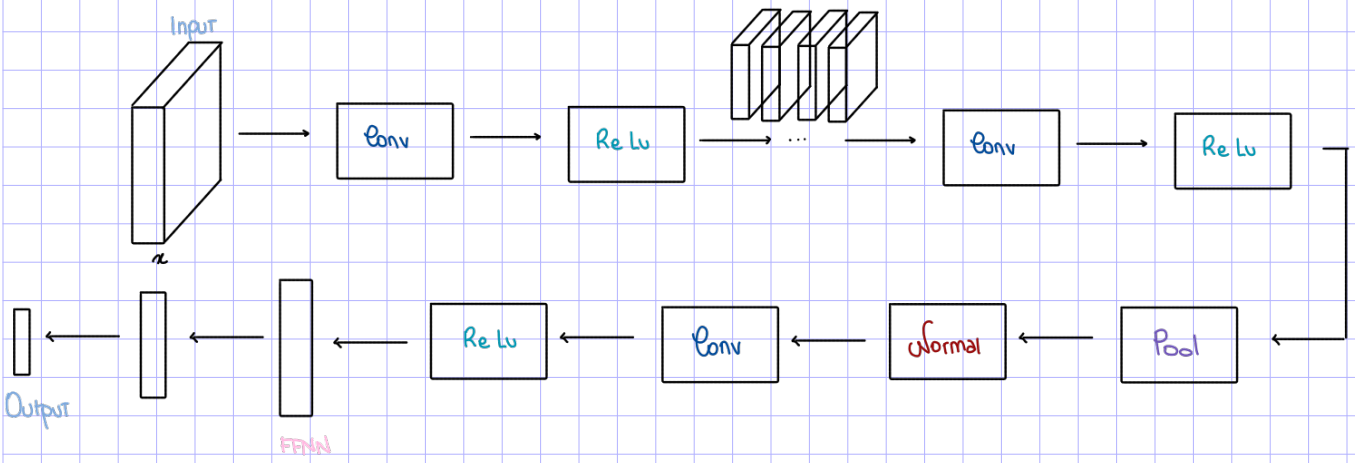
$E(z) = \frac{1}{N}\sum_{i=1}^{N} z$ $\quad$ $Var(z) = \frac{1}{N-1}\sum_{i=1}^{N}(z-E(z))^2$

Important: the mean & variance are computed only offline^{-1} during the training phase, in the validation phase they're fixed
The final oupur, has to belong to the originale scale, hence

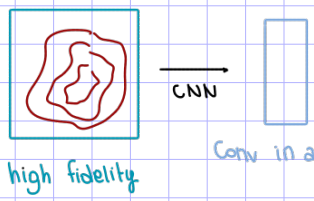
$$y = \gamma\hat{z} + \beta$$ with γ, β learnable parameters learn in the training phase

↓
capacity of the NN to get back z

9.5 Structure of a CNN



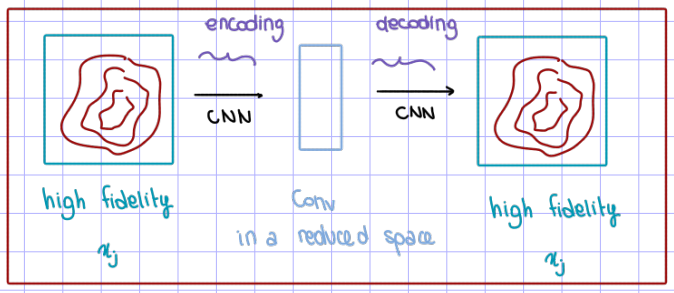
During this process, I do a sort of **conservation of the volume** during all this step
We start from the dimension of x_i and at the end we reach a smaller dimension
⟶ We can exploit this idea in PDE to reduce the image of the domain of PDE, i.e.



This process is called **encoding**, but we can see also the opposite one

↓

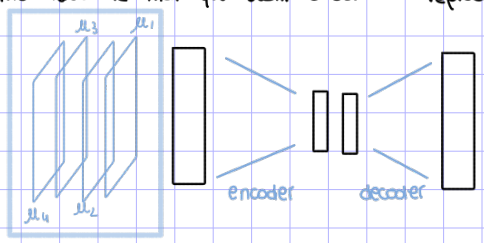
How we can encode the data in a supervised way?
↳ We add another convolutional neural network to enlarg again the dimension



⟶ This is an **AUTOENCODER NEURAL NETWORK**
After that we can use the decoders to build the final solution of the PDE

↓

The idea is that you want **exact** to reproduce the image. The scheme that you have to keep in mind is this one



1. Reduce the space ⟶ not-linear step — encoder
2. Enlarg again the space — decoder

What happen if I want to work with parametric stuff? We want to exploit the idea of NN, but for solving our problem. Hence we introduce another (NN) that takes the parameters μ and give in result the reduce space

- Offline you trane the net for each different parameter μ and build the encoder
- Online you exploit the decoder to obtain the result in the huge initial space

↓

The idea is to move from pixel to **graph neural network** to increase the accuracy of the encoder
Hence in the μ_i layer there are all the nodes and edge of the graph

NB We do not have any kind of idea of the error that we're doing during this step
↳ BIG PROBLEM: we have to fix this, but can we?

1° PHYSICS-INFORMED NEURAL NETWORK (PINN) ⟶ first way to apply NN to PDE's solver, train in unsupervision way

From the previous lessons, NN are just "magic operator" applies on number to compute classes.

↓

We're moving on unsupervision learning so we do NOT need snapshots to compute the solution

For more information ⟶ • 1998: Lagaris first implementation
• 2018: Karniadakis today best implementation

What is the idea beyond the PINN? Create a NN that is able to achieve $\hat{u}$ ⟶ INPUT to NN: point of domain
OUTPUT of NN: evaluation of $\hat{u}$ in the inpur point

Our goal is to solve a PDE

(1) $\begin{cases} A(\hat{u}) = f & \text{on } \Omega \subseteq \mathbb{R}^d \text{ open, bounded, Lipschitz} \\ B(\hat{u}) = g & \text{on } \Gamma \subseteq \partial\Omega \end{cases}$ ⟶ boundary condition (strong)

with $f: \Omega \to \mathbb{R}$ and $g: \Gamma \to \mathbb{R}$ given

($\hat{u}$: solution)

NB • $A(\cdot)$ operator can represents stationary & evolutionary problems, moreover it can be non-linear, time-dependent
• $B(\cdot)$ is a boundary operator, i.e. trace operator or normal derivate

The idea of PINN is to build a NN that learns how to solve PDE

Hence, we want $\mathcal{I}_{NN}: \Omega \longrightarrow \mathbb{R}$ with simulare FOM or using snapshots

$x \longmapsto u(x)$ approximation of $\hat{u}(x)$

$\underline{x} = \begin{pmatrix} x \\ y \\ t \\ \mu \end{pmatrix} \longmapsto u(x, y, t, \mu)$

it is a network with n inputs and 1 outputs

↓

We need physics to build the NN in an unsupervision way ⟶ we train $\mathcal{I}_{NN}$ unsupervised: we do NOT required snapshots

So, we create **loss function** using the residuals in the strong formulation

- $L_A^*(v, x) = \frac{1}{2} \left| f|_{\underline{x}} - A(v)|_{\underline{x}} \right|^2$
- $L_B^*(v, x) = \frac{1}{2} \left| g|_{\underline{x}} - B(v)|_{\underline{x}} \right|^2$

} Notice that we're NOT using weak formulation, but directly the strong formulation

and combined them via convex combination via λ Known a-priori

⟶ the **total loss function** is $\boxed{L^*(v, x) = L_A(v, x) + \lambda L_B(v, x)}$ that does not depends on snapshots

(λ: this is an hyperparameter (learn before the training) and balance the order of magnitude of the error; you can find this by hand or automatically somehow)

↓

The solution is $\boxed{u_{NN} := \underset{v \in \mathcal{I}_{NN}}{\operatorname{argmin}} L^*(v, x)}$, hence we have to minimize this function ⟶ $u_{NN} \in \mathcal{I}_{NN}$

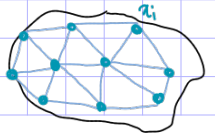
BUT it's NOT easy to find this solution in this way, due to non-linearity of equation or higher dimension (2D, 3D)

↳ The computation of L^* requires to strong hypothesis of f, g, A, B: one way to relax this is using integration
Thus we define a new loss function $L(v, x)$ given by $L(v) = L_A(v) + \lambda L_B(v)$ where $L_A(v) = \frac{1}{2}\int_\Omega |f - A(v)|^2$ & $L_B(v) = \frac{1}{2}\int_\Gamma |g - B(v)|^2$
and use a quadrature formula to approximate the integrals

How we can minimize the loss function? We have to compute some x to evaluate the loss function.
How to get quadrature points? There are different methods:

- **Meshless method** ⟶ similar to monte carlo method
We have the domain Ω, we randomly generate some $\{x_i\}$ and evaluate here the loss function

- **Mesh generation**
We discretize the space / domain and $\{x_i\}$ are the vertices of the mesh



1.1 Error PINN ⟶ what we can say about the error? The quality of u_{NN} depends on many factors

As we have something on the physics, we can measure the error that depends on 3 part:

1. Error capacity of NN : depends on the architecture of NN ⟶ number of layers, neurons

$$E_{NN} = \inf_{v \in \mathcal{Y}_{NN}} \|\hat{u} - v\|_X$$

⟶ best approximation of $\hat{u}$ in $\mathcal{Y}_{NN}$

2. Error associated to the loss function: associated to the minimization process

E_{Loss}

3. Error accuracy of the approximation of the residuals: ↗ related to the minimization problem

measure how good I am to compute L^*, so approximate correctly the physics ⇒ error integration

$$L^*(v) = \frac{1}{2}|f - A(v)|^2 + \frac{1}{2}|g - B(v)|^2$$

Here we need strong hypothesis because we use strong formulation

BUT we can relate the loss function ↘

$$L^*_W(v) = \frac{1}{2|\Omega|}\int_\Omega |f - A(v)|^2 + \frac{1}{2|\Omega|}\int_\Gamma |g - B(v)|^2$$

weak formulation

⇒ disadvantages: we have to solve integral

↘ use this formulation in the project!

Hence, the final error of the PINN is

$$\|u_{NN} - \hat{u}\|_X \leq E_{NN} + E_{Loss} + E_{QUAD}$$

To relax more the hypothesis, we can study the

1.2 Variational PINN ⟶ can we do something more? Yes, to relax the hypothesis we can use the weak form of (1)

We're introducing the variational (weak) form in the loss function

$$L^*_{WW}(v) = \frac{1}{2|\Omega|}\int_\Omega f(w) - a(v, w) + \cdots \qquad \forall w \in V_R$$

11 - POD NEURAL NETWORKS ⟶ reference paper: Non-intrusive reduced order modeling of non-linear problems using neural networks, Hesthaven & Ubbiali - 2018 JcP

The goal of the paper is to provide a fast tool to deal with non-linear problems
↳ with a non-affine structure

↓ How to build this tool?
They avoid projection online!

Problem formulation for non-linear problems.

Given a parameter $\mu \in \mathcal{P}$, find $u(\mu) \in V$ s.t. $g(u(\mu); \mu) = 0$ (1) strong version of the equation
(parametric problem)

i.e. Navie-Stocks $-\Delta u + (u \cdot \nabla u) + \nabla p = f$ (non-linear problem)

when we discretize ⟶ $Au + C(u)u + Bp = f$

there is this dependence that represents the non-linearity

↓

In weak formulation, we have $g(u(\mu), v; \mu) = 0 \quad \forall v \in V$ (2)
(assuming $f=0$, put that in the left-part)

Considering the discretized problem

⟶ $V_\delta \subset V$ and given the parameter $\mu \in \mathcal{P}$, find $u_\delta(\mu) \in V_\delta$ s.t. $g(u_\delta(\mu), v; \mu) = 0 \quad \forall v \in V_\delta$

Algebraically we can build the residual vector $G_\delta(u_\delta(\mu); \mu) = 0 \in \mathbb{R}^{N_\delta}$ where $\dim(V_\delta) = N_\delta$

How G is build? $G_\delta(\tilde{u}_\delta(\mu); \mu)_i = g(u_\delta(\mu), \varphi_i; \mu)$ where $\{\varphi_i\}_{i=1}^{N_\delta}$ is a FOM basis

End of FOM level

What should I do at the ROM level?

1. Build the space with POD:
   - Collect snapshots
   - Solve the generalized eigevalue problem over covariance matrix
   - Take eigenvectors related to the N-largest eigenvalues
   - Build the basis matrix
   - Project the matrices & the vectors ⟶ Attention! To build $\mathbb{R}^N \ni G_N(u_N(\mu); \mu) = B^T G_\delta(B u_N(\mu); \mu)$

   ISSUE: for each μ I have to assamble again the matrix

   Here, we cannot separare the variable, so we cannot perfor this

2. Solve the reduce systhem is NOT fast in N_δ ⟶ Not good

IDEA: can I avoid 2. using NL? We want to avoid the 2. step. How to do that?

POD-NN is an algorithm that exploits the direct projection of the snapshots in the reduce space to creare a map

Π^{NN} (Neural Network) s.t.
$\mu^* \longmapsto \Pi^{NN}(\mu^*) = u_N(\mu^*) \in \mathbb{R}^N$

I am learning the reduced coefficients

To build the ner we need a geometric interpreration of a reduced problem

- ROMs build a reduced basis matrix $B \in \mathbb{R}^{N_\delta \times N}$ where $B = [\varphi_1 \; | \; \varphi_N]$ ⟶ φ_i: basis function
- Let us consioler a linear problem: the simplest way to define direct projection

  i.e. $\begin{cases} -\Delta u = f & \forall v \in H_0^1(\Omega) \\ \text{boundary condition} \end{cases}$ ⟶ weak formulation $\int_\Omega \nabla u \nabla v \, dx = \int f u \, dy$

  This can be seen as minimization problem ↓ FOM $Au = f$ where $A_{ij} = \int_\Omega \nabla\varphi_i \nabla\varphi_j \, dx$ for $\{\varphi_i\}_{i=1}^{N_\delta}$ basis of the FOM

  matrix related to the grad-grad operator

  ↳ H_0^1-semi norm equivalent to H_0^1 norm

Considering, $A = X_\delta \in \mathbb{R}^{N_\delta, N_\delta}$ this is the inner product matrix, that during the labs we called sniffness matrix

↓

A small version of my matrix & vector can be written like that: $B^T A B = A_N$, $B^T f = f_N$

11.1 Algebraic properties of the Galerkin-ROM

The goal here is to find u_N without solving the system

↓

Hence, let us define the vector representation of the error $\boxed{e_\delta(\mu) = u_\delta(\mu) - B^T u_N(\mu)}$

The FOM residual, on the contrary, can be written as $\boxed{r_\delta(\mu) = f - AB\, u_N(\mu)}$ → take the small vector and project back

↳ depends on u_N and μ

The following hold

1. $Ae_\delta(\mu) = Au_\delta(\mu) - AB u_N(\mu) = f - AB u_N(\mu) = r_\delta(u_N;\mu)$ → The error goes to zero if $r_\delta(\mu)$ of the FOM solution is zero
2. $B^T A u_\delta(\mu) = B^T f = f_N$
3. $B^T r_\delta(u_N;\mu) = B^T f - B^T AB\, u_N = f_N - A_N u_N$ → $B^T r_\delta(u_N,\mu) \to 0$ if B is a "good" representation

$f_N - A_N u_N \to 0$ is the reduced system (RPI)

I do NOT want to solve $f_N - A_N u_N = 0$, but I want to express u_N in terms of $u_\delta(\mu)$ using the properties 1.,2.,3.

Let us take the FOM problem evaluated in u_N in the FOM space

$AB\, u_N(\mu) = f = Au_\delta(\mu)$

(if B is good; by FOM definition)

↓

Now we project, everything, in the reduced space

$B^T AB\, u_N(\mu) = B^T f = B^T A u_\delta(\mu)$

→ Then, $\boxed{u_N(\mu) = (B^T AB)^{-1} B^T A u_\delta(\mu)}$ → if we have $u_\delta(\mu)$, we have a direct way to compute u_N

↓ IP

and IP is the direct projection of $u_\delta(\mu)$ in the ROM space

Finally, we can train our net, because we can define the loss

$\mu^* \xrightarrow{\pi^{NN}} u_N(\mu^*)$ BUT How to train it?

- The input is μ
- The output is $u_N(\mu) \in \mathbb{R}^N$
- The loss could be $\mathcal{L} = \sum_\mu \frac{1}{N} \| \pi^{NN}(\mu) - u_N(\mu) \| = \sum_\mu \frac{1}{N} \| \pi^{NN} - IP\, u_\delta(\mu) \| = \sum_\mu \frac{1}{N} \| \pi^{NN} - (B^T X_\delta B)^{-1} B^T X_\delta\, u_\delta(\mu) \|$

↳ number of the snapshots; X_δ: this is the stiffness matrix; basis function matrix related to POD

- Hence, now we have all we need!

OFFLINE

- Snapshots collection
- POD
- Train the net → You're loosing the theorem that tells us that the projection is the best thing (including NON linearity)

\+ I exploit POD

\- The offline phase is longer

ONLINE

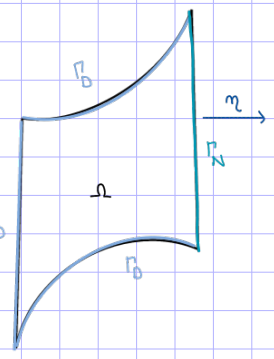
- Evaluation of the net: $\mu^* \longmapsto \pi(\mu^*)$

\+ Very fast: give me the parameter μ, I'll give you the representation

\- No control on the accuracy

12 - STOKES EQUATIONS in HDR ⟶ we're in a linear setting

Consider the domain Ω and we want to study (find)
- The fluid velocity $u = (u_1, u_2)$ on Ω (vectorial variable)
- The pressure



Make some assumption

H1. The fluid is **incompressible** ⟶ volume costant in time, it holds $\nabla \cdot u = 0$

H2. Do NOT consider the time: **stationary system**

Let us define the system

$$(1)\quad \begin{cases} -\gamma \Delta u + \nabla p = f & \text{in } \Omega \\ \nabla \cdot u = 0 & \text{in } \Omega \\ u = g & \text{in } \Gamma_D \subseteq \partial\Omega \neq \emptyset \\ -\gamma (\nabla u)^T \eta + p\eta = \psi & \text{in } \Gamma_N := \partial\Omega \setminus \Gamma_D \end{cases}$$

γ := viscosity, that for us it is a constant

⟶ strong boundary condition: we need to have a unique solution (Dirichlet)

⟶ Neuman boundary condition

If this boundary exist, we fix some condition on the pressure

So if this boundary does NOT exist, we solve the system and find the solution of p up to a constant

i.e. If $\Gamma_N = \emptyset$, for example you can state
- $p = 0$ on a point of Ω to solve the uniqueness of the solution
- $\int_\Omega p = 0$ on Ω

↓ Definition of the term

The definition of $\nabla u \in \mathbb{R}^{2,2}$ ⟶ $(\nabla u)_{ji} = \dfrac{\partial u_i}{\partial x_j} = (Ju)_{ij}$ (because it's the transpose of the jacobian)

The divergence is $\nabla \cdot u = \sum_{i=1}^{2} \partial_{x_i} u_i = \partial_x u_1 + \partial_y u_2$

The last definitio is $\Delta u := \nabla(\nabla \cdot u) - \nabla \times (\nabla \times u) = -\nabla \times (\nabla \times u)$, where the vectorial product is defined as

$$\nabla \times u := \begin{vmatrix} i & j & k \\ \partial_x & \partial_y & \partial_z \\ u_1 & u_2 & 0 \end{vmatrix} = i(-\partial_z u_2) - j(-\partial_z u_1) + k(\partial_x u_2 - \partial_y u_1) = (0, 0, \partial_x u_2 - \partial_y u_1)$$

↓ Hence, doing the computation we have $\nabla \times (\nabla \times u) = \begin{vmatrix} i & j & k \\ \partial_x & \partial_y & \partial_z \\ 0 & 0 & \partial_x u_2 - \partial_y u_1 \end{vmatrix} = i(\partial_y \partial_x u_2 - \partial_y \partial_y u_1) - j(\partial_x \partial_x u_2 - \partial_x \partial_y u_1) + k0$

Then $\nabla \cdot u = 0 \iff \partial_x u_1 + \partial_y u_2 = 0 \iff \partial_x u_1 = -\partial_y u_2$ (*)

⟶ $(-\partial_y^2 u_1 + \partial_y \partial_x u_2, -\partial_x^2 u_2 + \partial_x \partial_y u_1, 0) = (-\partial_y^2 u_1 - \partial_x^2 u_1, -\partial_x^2 u_2 - \partial_y^2 u_2, 0) = -(\Delta u_1, \Delta u_2)$

(we remove u_2 / we remove u_1)

Hence, returning to $\Delta u := \nabla(\nabla \cdot u) - \nabla \times (\nabla \times u) = (\Delta u_1, \Delta u_2)$ ⟶ $\Delta u = (\Delta u_1, \Delta u_2)$ ⟶ we can split and compute the laplacian of the 2 variable separately

↓

Can we go to the weak form? Yes, rewrite the system

$$(2)\quad \begin{cases} -\int_\Omega \gamma \Delta u \cdot v + \int_\Omega \nabla p \cdot v = \int_\Omega f \cdot v & \text{in } \Omega \quad \forall v \in ? \\ -\int (\nabla \cdot u) q = 0 & \text{in } \Omega \quad \forall q \in ? \end{cases}$$

in which space? we now do NOT know, it is a test function for velocity

another space that we do NOT know, it is a test function for the pressure

Now, we want to define the **Green's identities**

(1) $\int_{\partial\Omega} \psi \nabla\phi \cdot \eta = \int_\Omega \psi \Delta\phi + \int_\Omega \nabla\psi \cdot \nabla\phi$ ⟶ in Stokes equation: $\nabla\phi = u$, $\psi = p$

(2) (for vectorial variables) $\int_{\partial\Omega} v \cdot [(\nabla u)^T \eta] = \int_\Omega v \cdot \Delta u + \int_\Omega \nabla u : \nabla v$ where $A : B := \sum_{ij=1}^{d} A_{ij} B_{ij}$ double dot product (for square matrix, and obtain a scalar)

Applying to the system, we obtain

$$\begin{cases} \int_\Omega \gamma \nabla u : \nabla v - \int_\Omega p \nabla \cdot v - \int_{\partial\Omega} \gamma v \cdot ((\nabla u)^T \eta) + \int_{\partial\Omega} p v \cdot \eta = \int_\Omega f \cdot v & \forall v \\ -\int_\Omega (\nabla \cdot u) q = 0 & \forall q \end{cases}$$

BUT we are in a weak formulation, hence we want that $v|_{\Gamma_D} = 0 = (0,0)^T$ so we can consider that

$$-\int_{\partial\Omega} v\cdot((\nabla u)^T n) + \int_{\partial\Omega} p\, v\cdot n \longrightarrow \begin{cases} 0 \text{ in } \Gamma_D \\ \neq 0 \text{ in } \Gamma_N \end{cases}$$

↓ Rewrite the system considering this observations ⟶ **Weak form stokes equations**

$$\begin{cases} \int_\Omega \nu \nabla u : \nabla v - \int_\Omega \underbrace{p}_{\text{velocity}} \underbrace{\nabla\cdot v}_{\text{pressure}} = \int_\Omega f\cdot v - \int_{\Gamma_N} \psi\cdot v & \forall v \in H^1_0(\Omega) \\ -\int_\Omega (\nabla\cdot u) q = 0 & \forall q \in L^2(\Omega) \end{cases}$$

↳ This terms are "the same"

Let's now deal with the space:

- **preassure space** $p, q \in L^2(\Omega)$, and also the divergence has to belong to the same space ↳ we have NOT to derive it, no gradient, very weak hypothesis
- **forcing term** $f \in [L^2(\Omega)]^2$ ↳ because they are array
- **velocity space** $u \in [H^1_g(\Omega)]^2 := \{ v \in [H^1(\Omega)]^2 : v|_{\Gamma_D} = g \}$
  $v \in [H^1_0(\Omega)]^2 := \{ v \in [H^1(\Omega)]^2 : v|_{\Gamma_D} = 0 \}$

Remark -
- **R1** If $\Gamma_N = \Phi$, we have to set conditions to have a unique solution ⟶ $p \in L^2_0(\Omega) := \{ v \in L^2(\Omega) : \int_\Omega v = 0 \}$
- **R2** Implementing this problem, we define $u = u_0 + G$, where $u_0 \in [H^1_0(\Omega)]^2$, $G|_{\Gamma_D} = g$ is the derection of the Dirichler condition

Make all these informations together...

$$\begin{cases} \underbrace{\int_\Omega \nu \nabla u_0 : \nabla v}_{a(\cdot,\cdot)} - \underbrace{\int_\Omega p \nabla\cdot v}_{b(\cdot,\cdot)} = \underbrace{\int_\Omega f\cdot v - \int_{\Gamma_N} \psi\cdot v - \int_\Omega \nu \nabla G : \nabla v}_{F(\cdot)} & \forall v \in [H^1_0(\Omega)]^2 \equiv V \\ \underbrace{-\int_\Omega (\nabla\cdot u) q}_{b(\cdot,\cdot)} = 0 & \forall q \in L^2(\Omega) \equiv Q \end{cases}$$

Introducing the bilinear form, to write everything in a contract way, we have $\begin{cases} a(\cdot,\cdot) - b(\cdot,\cdot) = F(\cdot) & \forall v \in V \\ -b(\cdot,\cdot) = 0 & \forall q \in Q \end{cases}$ [VP] ↳ variational problem

where $a(\cdot,\cdot) : V\times V \to \mathbb{R}$, $a(u,v) = \int_\Omega \nu \nabla u : \nabla v$, $b(\cdot,\cdot) : V\times Q \to \mathbb{R}$, $b(v,q) := -\int_\Omega q \nabla\cdot v$

$F(\cdot) : V \to \mathbb{R}$, $F(v) = \int_\Omega f\cdot v - \int_{\Gamma_N} \psi\cdot v - \int_\Omega \nu \nabla G : \nabla v$

↓

Hence we obtain the **saddle point problem**

$$\begin{cases} \text{Find } u \in V,\ p \in Q \text{ s.t.} \\ a(u,v) + b(v,p) = {}_{V'}\langle F, v\rangle_V & \forall v \in V \quad \text{VSP} \\ b(u,q) = 0 & \forall q \in Q \end{cases}$$

Theorem - Brezzi's Theorem

Hypothesis -
- **H1.** $a(\cdot,\cdot)$ is continuous in V ⟶ $\exists\, \gamma_a > 0 : |a(u,v)| \le \gamma_a \|u\|_V \|v\|_V \quad \forall u, v \in V$
- **H2.** $a(\cdot,\cdot)$ is coercive in V_0 ⟶ $V_0 := \{ v \in V : b(v,q) = 0 \ \forall q \in Q \} \subset V$
  $\exists\, \alpha_a > 0 : a(u,u) \ge \alpha_a \|u\|^2_V \ \forall u \in V_0 \iff \sup_{w \in V_0} \frac{a(w,v)}{\|w\|_V} \ge \alpha_a \|v\|_V \ \forall v \in V_0$ (equivalence)
- **H3.** $b(\cdot,\cdot)$ is continuous on $V\times Q$ ⟶ $\exists\, \gamma_b > 0 : b(v,q) \le \gamma_b \|v\|_V \|q\|_Q \quad \forall v \in V \ \forall q \in Q$
- **H4.** $b(\cdot,\cdot)$ is inf-sup stable in $V\times Q$ ⟶ $\exists\, \beta_b > 0 : \inf_{q\in Q} \sup_{v\in V} \frac{b(v,q)}{\|v\|_V \|q\|_Q} \ge \beta_b > 0 \iff \sup_{v\in V} \frac{b(v,q)}{\|v\|_V} \ge \beta_b \|q\|_Q \ \forall q \in Q$ (equivalence)

↳ extension of coercivity for bilinear form in 2 dimension

Thesis - Then $\exists!\, (u,p) \in V\times Q$ solution of [VP] ↳ Unique continuous solution

1. $\|u\|_V \le \frac{1}{\alpha_a} \|F\|_{V'}$
2. $\|p\|_Q \le \frac{1}{\beta_b}\left(1 + \frac{\gamma_a}{\alpha_a}\right) \|F\|_{V'}$ (*)

What happen in the Galerkin approximation? See Lecture 11 for more details

12.1 Supermizer operator

Is the operator define such that $T: Q \longrightarrow V$, where $\forall q \in Q \quad Tq \in V: (Tq, w)_V = b(w,q) \quad \forall w \in V$ that is an implicite definition of the **supermizer operator** T

↓ This is usefull only in the discrete space, not in the continuity one

Property - $\forall Tq \in \arg\sup_{w \in V} \frac{b(w,q)}{\|w\|_V}$ when you apply the operator to the pression you obtain the velocity

12.2 What happen when we want to study the stokes equation in a discrete space?

We discretize the domain V_δ, Q_δ with the same tassellation δ to study the discrete space

↓

What happen to the Hp of the Brezzi Theorem?

Consider $\beta_b := \inf_{q \in Q} \sup_{v \in V} \frac{b(v,q)}{\|v\|_V \|q\|_Q} > 0$, if I restrict Q

↳ We have to study the discrete inf-sup stability

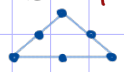
- •) The pressure resulting from the inf sup belongs to Q_δ: ✓
- •) The pressure is NOT in Q_δ: ✓ it is inf
- •) The velocity is in V_δ: ✓
- •) The velocity is NOT in V_δ: this is really a big problem! ✗
  The value β_δ could be 0 ⟶ HUGE PROBLEM (0)

↓

Hence, NOT ALL THE DISCRETIZATIONS ARE GOOD, then the usual space used is the **Taylor-Hood FEM spaces**

$\begin{cases} v \in FEM \quad K=2 \\ q \in FEM \quad K=1 \end{cases}$ ⟶ 

basis function φ_i: { 1 in the vertex i, 0 in the other points



basis also in the middle of the edge

This is a good and stable tassellation, moreover the error of the velocity goes to zero faster wrt the one on the q

$\{p_N = p(\mu)\}_N$

12.3 Reduced Model Order for Stokes equation

We want to reduce V_{SP_δ} in the classic way, hence we can take the snapshots $\{u_N = u_\delta(\mu)\}_N$

The same observation as before for the inf sup value, How can we effort? Add ψ to the space so that we have nor problem with the value of infsup ⟶ We have to build the reduced space in order that the inf-sup stability holds. Indeed in high fidelity $\{w_i, p_i\}_N$ the inf-sup stability is given by the choice of Q_δ, V_δ

↓

Add other velocity to the ones find with POD until the new basis $\{\psi_i\}_N \{\phi_i\}_N$ satisfies the condition of infsup

To do that we exploit the **supermizer operator**. Let's see the discrete version

To obtain the stability we enlarge the space with T-operator

$T: Q_\delta \longrightarrow V_\delta$ s.t. $\forall q_\delta \in Q_\delta \quad Tq_\delta \in V_\delta$

$\forall w_\delta \in V_\delta \quad \boxed{(Tq_\delta, w_\delta)_V = b(w_\delta, q_\delta)}$ ⟶ This now is a linear system

But, it holds the same property ⟶ $Tq_\delta \in \arg\sup_{w_\delta \in V_\delta} \frac{b(w_\delta, q_\delta)}{\|w_\delta\|_V}$ ⟶ This is good because in that way it garantees the existance of the infsup

Corollary - The constrants $\beta_\delta := \inf_{q_\delta \in Q_\delta} \sup_{w_\delta \in V_\delta} \frac{b(w_\delta, q_\delta)}{\|w_\delta\|_V \|q_\delta\|_Q} := \inf_{q_\delta \in Q_\delta} \frac{\|Tq_\delta\|_V}{\|q_\delta\|_Q}$ ⟶ This holds thanks to the properties

Then $\beta_N \geq \beta_\delta \geq \beta_0 > 0$ ✓

So that we can build the reduced spaces

Fixing $Q_N \subset Q_\delta$, $\boxed{Q_N := \text{span}\{p_\delta^i\}_{i=1}^m}$ find, i.e. with POD, the right choice for the space of the velocity is

$\boxed{V_N := \text{span}\{w_\delta^i\}_{i=1}^m \cup \text{span}\{Tp_\delta^i\}_{i=1}^m}$ ⟶ this is infsup stable & $\beta_N \geq \beta_\delta \geq \beta_0 > 0$ ✓

NB This approach can be done also with other model of PDE to deal with the same problems

12.4 Remark on Stokes High-Fidelity implementation

We have this specific problem, that is the high-fidelity problem

$$\begin{cases} a(u_\delta, v_\delta) - b(v_\delta, p_\delta) = F(v_\delta) & \forall v_\delta \in V_\delta \\ -b(u_\delta, q_\delta) = 0 & \forall q_\delta \in Q_\delta \end{cases} \qquad a(u_\delta, v_\delta) := \int_\Omega \gamma \nabla u_\delta : \nabla v_\delta \;,\; b(u_\delta, q_\delta) := \int_\Omega q_\delta \nabla \cdot u_\delta \qquad F(v_\delta) = \int_\Omega f \cdot v_\delta$$

In FEM, we fix the basis function
- Q_δ: $\{\varepsilon_k\}_{k=1}^{N_\delta^p}$
- V_δ: $\{\psi_j\}_{j=1}^{N_\delta^u}$ → vectorial basis $\psi_j = \{\psi_j^1, \psi_j^2\} \in \{[\varphi_i, 0], [0, \varphi_i]\}$ → standard FEM basis $P_2(\Omega)$

$$\left.\begin{matrix} [\varphi_1, 0] \\ [\varphi_2, 0] \\ \vdots \\ [0, \varphi_{N_\delta^u}] \\ [0, \varphi_{N_\delta^u}] \end{matrix}\right\} = \psi_j$$

where $\{\varphi_i\}_{i=1}^{N_\delta^u}$ is the FEM standard basis $N_\delta^u = \frac{N_\delta^u}{2}$

How to discretize with the FEM?
↳ We need to define a smart basis

Consider now (1) $A_{ij} = a(\psi_j, \psi_i) = \int_\Omega \gamma \nabla \psi_i : \nabla \psi_j$ ⇒ if $k \in (0, \dots, N_\delta)$ $\psi_k = (\varphi_j, 0)$

we can split the computation thanks to the basis that we choose

$i \in (0, \dots, N_\delta)$, $j \in (0, \dots, N_\delta)$

$$A_{ij}^1 = \int_\Omega \gamma \nabla \begin{pmatrix} \varphi_i \\ 0 \end{pmatrix} : \nabla \begin{pmatrix} \varphi_j \\ 0 \end{pmatrix} = \int_\Omega \gamma \begin{bmatrix} \partial_x \varphi_i & 0 \\ \partial_y \varphi_i & 0 \end{bmatrix} : \begin{bmatrix} \partial_x \varphi_j & 0 \\ \partial_y \varphi_j & 0 \end{bmatrix} = \int_\Omega \gamma \left(\partial_x \varphi_i \partial_x \varphi_j + \partial_y \varphi_i \partial_y \varphi_j\right)$$

$$= \int_\Omega \gamma \nabla \varphi_i \cdot \nabla \varphi_j \quad \text{gradient } \varphi_j \text{ dot gradient } \varphi_j$$

$i \in (0, \dots, N_\delta)$, $j \in (N_\delta+1, \dots, 2N_\delta)$ $A_{ij} = \int_\Omega \gamma \begin{bmatrix} \partial_x \varphi_i & 0 \\ \partial_y \varphi_i & 0 \end{bmatrix} : \begin{bmatrix} 0 & \partial_x \varphi_j \\ 0 & \partial_y \varphi_j \end{bmatrix} = 0$ by definition dot product

$i \in (N_\delta+1, \dots, 2N_\delta)$ $j \in (0, \dots, N_\delta)$ $A_{ij} = 0$

$i \in (N_\delta+1, \dots, 2N_\delta)$ $j \in (N_\delta+1, \dots, 2N_\delta)$ $A_{ij}^2 = \int_\Omega \gamma \nabla \varphi_i : \nabla \varphi_j$

(2) $B_{kj} = b(\psi_j, \varepsilon_k) = \int_\Omega \varepsilon_k \nabla \cdot \psi_j$

$$= \underbrace{\left(\int_\Omega \varepsilon_k \partial_x \varphi_j^1\right)_{j \in (1, \dots, N_\delta)}}_{(B_{FEM}^1)_{kj}} \underbrace{\left(\int_\Omega \varepsilon_k \partial_y \varphi_j^2\right)_{j \in (N_\delta+1, \dots, 2N_\delta)}}_{(B_{FEM}^2)_{kj}}$$

$$= \underbrace{\left(\int_\Omega \varepsilon_k [1,0] \cdot \nabla \varphi_j^1\right)_{j \in (1, \dots, N_\delta)}}_{(B_{FEM}^1)_{kj}} \underbrace{\left(\int_\Omega \varepsilon_k [0,1] \cdot \nabla \varphi_j^2\right)_{j \in (N_\delta+1, \dots, 2N_\delta)}}_{(B_{FEM}^2)_{kj}}$$

(3) $q_i = F(\psi_i) = \int_\Omega f \cdot \psi_i$

$$= \underbrace{\left(\int_\Omega f_1 \varphi_i^1\right)_{i \in (1, \dots, N_\delta)}}_{q_{FEM}^1} \underbrace{\left(\int_\Omega f_2 \varphi_i^2\right)_{i \in (N_\delta, \dots, 2N_\delta)}}_{q_{FEM}^2}$$

Hence, the stokes linear system becomes

$$\begin{pmatrix} A^1 & 0 & B_{FEM}^{1\,T} \\ 0 & A^2 & B_{FEM}^{2\,T} \\ B_{FEM}^1 & B_{FEM}^2 & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ p \end{pmatrix} = \begin{pmatrix} q_{FEM}^1 \\ q_{FEM}^2 \\ 0 \end{pmatrix}$$

(N_δ^u rows/columns for A^1, A^2, $B_{FEM}^{1\,T}$, $B_{FEM}^{2\,T}$; N_δ^p for B_{FEM}^1, B_{FEM}^2; N_δ for u_1, u_2)

This is the solution that we want to find

$:= J$

13 - NON LINEAR PROBLEMS

The non-linearity is in the bilinear form $a(u,v) = F(v)$, hence $a(u,v)$ is non linear in u, i.e. $a(u,v) = \max(u,v)$ or $a(u,v) = u \cdot |u| \cdot v$

↓ We have to linearize the problem, i.e. using Newton Method

13.1 Newton Method: linear system resolver

It's an iterative method in which we assume $f(u)=0$ and the iterative step is $u_{n+1} = u_n + \delta u$ (what we have to compute)

Exploits the Taylor expansion centered in u_n, we can compute, using $\delta u = u_{n+1} - u_n$

$$f(u_{n+1}) = f(u_n) + J_f(\delta u)|_{u_n} \delta u + O(\|\delta u\|^2)$$

- This term has to go to zero
- Jacobian first order difference
- negligible second order term

→ IDEA: solve a non-linear PDE by iteratively solving a sequence of linear PDE. At each step, you linearize the non-linear PDE around your current best guess for the solution & solving the resulting linear PDE to find update to go closer to the solution

Hence → while $\delta u \neq 0$ do → $\delta u \neq 0$ or $f(u_{n+1}) \neq 0$ are two equivalent stopping criteria

$$J_f(\delta u)|_{u_n} \delta u = -f(u_n)$$

Definition of the Jacobian: (direction v)

$$J_f[v]|_{x_0} := \lim_{h \to 0} \frac{f(x_0 + hv) - f(x_0)}{h}$$

if we can compute this, we can solve the system

12 Gateaux differentiable: if exists the limit of the incremental ratio

In the laboratory we'll study the **burger equation** $-\Delta u + u \nabla \cdot u = g$, where $\nabla \cdot u := \partial_x u + \partial_y u$ where u is scalar

- Sort of with scalar variable
- definition of this strange divergence

↓

The first thing to do is to go to the **variational form** to write the correct form of f

$$f: \underbrace{\int_\Omega \nabla u \cdot \nabla v}_{f_1} + \underbrace{\int_\Omega u \nabla \cdot u\, v}_{f_2} - \underbrace{\int_\Omega g v}_{f_3} = 0$$

Now we can split f in f_i so that we can compute the Jacobian

- Both f_1 & f_3 are linear function, so let's try to compute the Jacobian

Diffusion $J_{f_1}(\delta u)|_{u_N} = \lim_{h \to 0} \frac{f_1(u_N + h\delta u) - f_1(u_N)}{h} = \lim_{h \to 0} \frac{f_1(u_N) + h f_1(\delta u) - f_1(u_N)}{h} = f_1(\delta u) = \int_\Omega \nabla u \cdot \nabla v$ → This holds for all linear function (f_1 is linear)

$J_{f_3}(\delta u)|_{u_N} = 0$ because it doesn't appear u in the function

- For f_2, that is not linear, we have to compute the Jacobian in a different way

$$J_{f_2}(\delta u)|_{u_N} = \lim_{h \to 0} \frac{f_2(u_N + h\delta u) - f_2(u_N)}{h} = \lim_{h \to 0} \frac{\int_\Omega (u_N + h\delta u)\nabla(u_N + h\delta u)v - \int_\Omega (u_N)\nabla(u_N)v}{h} \overset{\text{the integral is linear}}{=} \text{we can split}$$

$$= \lim_{h \to 0} \frac{1}{h}\Big(\underbrace{\int_\Omega u_N \nabla u_N v - \int_\Omega u_N \nabla \cdot u_N}_{=0} + \underbrace{\frac{h^2}{h}(\cdot\cdot)}_{\text{neglectable}} + \underbrace{\int_\Omega u_N \nabla \cdot (\delta u) v}_{\text{advection term}} + \underbrace{\int_\Omega \delta u \nabla \cdot u_N v}_{\text{reaction term}}$$

advection term: $\int_\Omega (u_N[1,1]) \nabla \cdot (\delta u) v$ — reaction term: $\int_\Omega (\nabla \cdot u_N) \delta u v$

Newton Schema $J_f[\delta u]|_{u_N} \cdot \delta u = -f(u_N)$

$$\rightarrow \left(\int_\Omega \nabla \delta u \cdot \nabla v + \int_\Omega (u_N[1,1]) \cdot \nabla \delta u\, v + \int_\Omega (\nabla \cdot u_N) \delta u v \right) = -\int_\Omega \nabla u_N \nabla v - \int_\Omega u_N \nabla \cdot u_N v + \int_\Omega g v \qquad \forall v \in V = H_0^1(\Omega)$$

With respect to Lab 12. $\int_\Omega h(u_N)\, e(\delta u)\, m(v) \simeq \sum_{q=1}^{Q} h(u_N, x_q)\, e(\delta u, x_q)\, m(v, x_q)\, w_q$ → quadrature formulae

$h(u_N)$: burger_non_linear_c => $h(u_N) = \nabla \cdot (u_N) = \partial_x u_N|_{x_q} + \partial_y u_N|_{y_p}$

Algorithm 3 Newton's Method for Nonlinear PDEs

Input: Nonlinear PDE operator R, Tolerance ϵ, Maximum iterations m_{max}, Initial guess u_0.
Output: Approximate solution u.
Initialize $m = 0$.
Choose an initial guess u_0.
for $m = 0, 1, 2, \ldots, m_{max} - 1$ **do**
  Evaluate the residual $R(u_m)$. *#Often in variational form:* $F(u_m)(v)$
  Evaluate the linearized operator $R'(u_m)$. *#Often in variational form: find* $J_F[\cdot](\cdot)$
  Solve the linear equation $R'(u_m)[\delta u_m] = -R(u_m)$ for the correction δu_m. *#Often in variational form:* $J_F[\delta u_m](v) = -F(u_m)(v)$
  Update the solution: $u_{m+1} = u_m + \delta u_m$.
  Check for convergence.
  if $||R(u_{m+1})|| < \epsilon$ or $||\delta u_m|| < \epsilon$ **then**
    Set $u = u_{m+1}$.
    break
  end if
end for
if $m == m_{max} - 1$ and convergence not reached **then**
  Warning: Maximum number of iterations reached without convergence.
end if
Set $u = u_{m+1}$.
Return: u.

13.2 Evaluation of the error of the solution

u_{ex} := exact solution
u_R := high fidelity approximation
u_{PINN} := PINN solution
} → compute error with different norm, i.e.

L_2

1. Discretize the domain to take mesh
2. Compute integral on each triangle via quadrature formula

$$\int_\Omega (u_n - u_{PINN})^2 \simeq \sum_T \int_T (u_R - u_{PINN})^2 \simeq \sum_T^1 \sum_{x_0}^1 (u_R - u_{PINN})^2_{|_{xq}} w_q$$

quadrature Points

3. Sol_x := derivatives of numeric solution in quadrature Points
   Sol_y := derivatives wrt y of numeric solution in quadrature Points

14 – NAVIER-STOKES EQUATIONS

These equations describes the motion of fluids. We work with **steady** NS equations for **incompressible** flows.
(the property of the fluid in a fixed point do not change in the time; the density of the fluid is constant in every point at in every time)

The problem in **strong formulation** is: (non linear term)

- find $(u,p) \in \mathcal{U} \times Q$ s.t.

$$\begin{cases} -\mu \Delta u + (u\cdot\nabla)u + \nabla p = f & \text{in } \Omega \subset \mathbb{R}^d \\ \operatorname{div}(u) = 0 & \text{in } \Omega \subset \mathbb{R}^d \\ u = g & \text{in } \Gamma_D \subset \partial\Omega \\ -p\eta + \mu(\nabla u)\eta = h & \text{in } \Gamma_N \subset \partial\Omega \end{cases} \qquad \begin{cases} \Gamma_D \cap \Gamma_N = \phi \\ \Gamma_D \cup \Gamma_N = \partial\Omega \end{cases}$$

(u: velocity $\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$, p: pressure)

with d is the dimension of the problem
f is the forcing term $f \in [L^2(\Omega)]^d$
$\mu \in P$ is the parameter
g, h have proper regularity ⟶ for semplicity, $g = h = 0$ we can study the homogeneous case

We can pass to **weak formulation**

- find $(u,p) \in \tilde{\mathcal{U}} \times \tilde{Q}$ s.t. (non linear part)

$$\begin{cases} \mu a(u,v) + c(u,w,v) + b(u,p) = f(v) & \forall v \in \tilde{\mathcal{U}} \\ b(u,q) = 0 & \forall q \in \tilde{Q} \end{cases}$$

with $\tilde{\mathcal{U}} := [H^1_{\Gamma_D}(\Omega)]^d$
$\tilde{Q} := L^2(\Omega)$ if $\Gamma_N = \phi$, $\tilde{Q} := L^2_0(\Omega)$ to fix an added condition to have unique solution $\int_\Omega p\,dx = 0$

$a(u,v) := +\int_\Omega \nabla u : \nabla v \, dx$ ⟶ Laplacian wrt x of u : Laplacian wrt y of u ($u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$)

$b(u,p) := -\int_\Omega p \operatorname{div}(v)\, dx$

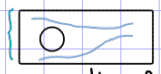
$c(u,w,v) := \int (u\cdot\nabla) w \cdot v \, dx$ ⟶ Scalar product between u & the gradient operator that produces two equations

Equation 1 $\int_\Omega \left(u_1 \frac{\partial}{\partial x} w_1 v_1 \right) dx + \int_\Omega \left(u_2 \frac{\partial}{\partial y} w_1 v_1 \right) dx$

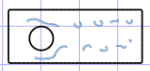
Equation 2 $\int_\Omega \left(u_1 \frac{\partial}{\partial x} w_2 v_2 \right) dx + \int_\Omega \left(u_2 \frac{\partial}{\partial y} w_2 v_2 \right) dx$

14.1 Flow Regime

We have two different kind of flow regime



- **Laminar**: the flows goes around the obstacle and nothing happen
- **Turbulent**: the flows creates a lot of vortices after obstacle



It is determined by the **Reynolds Number** $Re := \frac{LU}{\mu}$ where

$\mu :=$ Kinematic viscosity
$L :=$ characteristic lenght of Ω
$U :=$ characteristic velocity of Ω ⟶ related to boundary condition

How can I solve this problem numerically?

14.2 General formulation of nonlinear problems

The problem becomes

find $u \in \mathcal{U}$ s.t. $g(u(\mu), v; \mu) = 0 \quad \forall v \in \mathcal{U}$

At the FOM level I introduce $\mathcal{U}_\delta \subset \mathcal{U}$ and the problem now is

find $u_\delta \in \mathcal{U}_\delta$ s.t. $g(u_\delta(\mu), v_\delta; \mu) = 0 \quad \forall v_\delta \in \mathcal{U}_\delta$

The problem is well posed if the **Frechet derivative** of g at $u_\delta(\mu)$ is $dg(u_\delta(\mu))$ (applied to 2 variable) — continuous, inf-sup stable

We solve the problem with Newton Method: given $u^0_\delta(\mu) \in \mathcal{U}_\delta$ we iterate over K

$$dg\left[u^K_\delta(\mu)\right](\delta^K u_\delta, v_\delta; \mu) = -g(u^K_\delta(\mu), v_\delta; \mu)$$

Newton step

Jacobian Matrix — Unkown — Residual Vector

$$u^{K+1}_\delta(\mu) = u^K_\delta(\mu) + \delta^K_{u_\delta}$$

More precisely, for NSE:

find $(\delta u^k, \delta p^k) \in \tilde{u}_\delta \times \tilde{Q}_\delta$ s.t.

$$\begin{cases} (1)\ a(\delta u^k_\delta(\mu), u_\delta;\mu) + c(u^k_\delta, \delta u^k_\delta, v_\delta) + c(\delta u^k_\delta, u^k_\delta, v_\delta) + b(v_\delta, \delta p^k_\delta) = f(u_\delta) - a(u^k_\delta, v_\delta) - c(u^k_\delta, u^k_\delta, v_\delta) - b(v_\delta, p^k_\delta) \\ (2)\ b(\delta u^k_\delta, q_\delta) = -b(u^k_\delta, q_\delta) \end{cases}$$

(unknown: u^k_δ; unknown: δu^k_δ)

with

$$(u^{k+1}_\delta, p^{k+1}_\delta) = (u^k_\delta, p^k_\delta) + (\delta u^k_\delta, \delta p^k_\delta) \longrightarrow X^{k+1} = X^k + \delta^k X \qquad X = (u_\delta, p_\delta)$$

Algebraically

$J_\delta(X^k, \mu)\, \delta^k X = -G_\delta(X^k_\delta, \mu)$, where the jacobian matrix for NSE is $J_f = \begin{pmatrix} A + C_1(u^k_\delta) + C_2(u^k_\delta) & B^T \\ B & 0 \end{pmatrix}$

where $A :=$ stiffness matrix

$B_{ij} := -\int_\Omega \psi_i \operatorname{div}(\varphi_j)\, dx \longrightarrow \{\psi_i\}_{i=1}^{N^p_\delta}$ span $\tilde{Q}$ & $\{\varphi_j\}_{j=1}^{N^u_\delta}$ span $\tilde{U}$ different from Vicini's notes

$C_1(u^k_\delta)_{ij} = c(u^k_\delta, \varphi_j, \varphi_i)$

$C_2(u^k_\delta)_{ij} = c(\varphi_j, u^k_j, \varphi_i)$

What about the ROM level? We have to create the basis, i.e. with POD and compute the supremizer

To compute the basis we use POD and

1. We proceed standard for pressure to obtain $\mathbb{V}_p \in \mathbb{R}^{N^p_\delta, N}$
2. We enrich the velocity space with supremizer obtaining $\mathbb{V}_u \in \mathbb{R}^{N^u_\delta, 2N}$ $\mathbb{V}_u = [\mathbb{V}_{\text{standard } u} \mid \mathbb{V}_S]$ a global stabilized basis $\mathbb{V} = \begin{bmatrix} \mathbb{V}_u & 0 \\ 0 & \mathbb{V}_p \end{bmatrix} \in \mathbb{R}^{N^u_\delta + N^p_\delta, 3N}$

We want to apply a Reduced Newton

Algo For every k

$J_N(X^k_N, \mu)\, \delta X_N = -G_N(X^k_N, \mu)$, where $J_N(X_N;\mu) = \begin{pmatrix} A_N + C_{1N}(u^k_N) + C_{2N}(u^k_N) & B_N^T \\ B_N & 0 \end{pmatrix}$

$A_N = \mathbb{V}_u^T A \mathbb{V}_u$

$B_N = \mathbb{V}_p^T B \mathbb{V}_u$

$C_{1N} = \mathbb{V}_u^T C_1(\mathbb{V}_u^T u^k_N; \mu)\mathbb{V}_u$ $\longrightarrow$ depends on FOM level

$C_{2N} = \mathbb{V}_u^T C_2(\mathbb{V}_u^T u^k_N; \mu)\mathbb{V}_u$

↓

PROBLEM! No separation of the variable, at each step at the ROM level you have to compute FOM level

How to go from ROM to FOM during the iteration?

Considering k-th iteration

- I want to solve $J_N(X^k_N, \mu)\, \delta X_N = -G_N(X^k_N, \mu)$
- Namely at each iteration of the reduced solver $\mathbb{V}^T J_\delta(\mathbb{V} X^k_N; \mu)\mathbb{V}\, \delta X^k_N = -\mathbb{V}^T G_\delta(\mathbb{V} X^k_N; \mu)$
  1. Compute the FOM representation of X^k_N
  2. Assemble $J_\delta(\mathbb{V} X^k_N; \mu)$ and $G_\delta(\mathbb{V} X^k_N; \mu)$
  3. Project jacobian and residual vector
  4. Solve reduced system obtaining X^{k+1}_N

But, all these steps are so consuming! To avoid this you can apply some ML strategy that we already study

15 - NOT AFFINE PROBLEM

Affinity property: consider $g: \bar{\Omega} \times \mathcal{P} \longrightarrow \mathbb{R}$, $(x, \mu) \longmapsto g(x, \mu)$ that can be split in this formulation $g(x,\mu) = \sum_{q=1}^{Q} \vartheta_q(\mu) g_q(x)$

↓ be able to do offline vs online phase

BUT if we do not have this hypothesis?
1. NN
2. Interpolation with Empirical Interpolation Method

15.1 EIM: empirical interpolation method

We want to exploit a NOT-AFFINE approximation $g(x,\mu) = \sum_{q=1}^{Q} \vartheta_q(\mu) g_q(x)$

↓

To do that, we have to define the interpolation operator $\mathcal{I}_Q^x : \mathcal{G} \longrightarrow X_Q^{EIM}$ where

$\mathcal{G} := \{ g(\cdot,\mu) : \mu \in \mathcal{P} \} \subset C^0(\bar{\Omega})$

X_Q^{EIM} of dimension Q generated by $\{h_1(x), \dots, h_Q(x)\}$

such that, the operator can be written as

$$\mathcal{I}_Q^x : \mathcal{G} \longrightarrow X_Q^{EIM}, \quad g(x,\mu) \longmapsto (\mathcal{I}_Q^x g)(x,\mu) := \sum_{q=1}^{Q} \vartheta_q(\mu) h_q(x) \quad \forall \mu \in \mathcal{P}$$

IDEA: via interpolation, find an approximation of g that is affine so that we can do offline/online

$\{\vartheta_q(\mu)\}_{q=1,\dots,Q}$ are computed $(\mathcal{I}_Q^x g)(t^q, \mu) = g(t^q, \mu) \quad \forall \mu \in \mathcal{P}, \ \forall q \in \{1, \dots, Q\}$

This interpolation points are called magic points := points in which the rule of interpolation holds

AIM: minimize the error of the interpolator operator

We do that with a greedy strategy

↓

Initialization

1. Select some parameter $\mu_{EIM}^1 := \underset{\mu \in \mathcal{P}}{\operatorname{argmax}} \| g(\cdot,\mu) \|_{L^\infty(\Omega)}$ (can be also the forcing term)
2. Create a set to collect the parameter $S_1 := \{\mu_{EIM}^1\}$
3. Define the function $\xi_1(x) := g(x, \mu_{EIM}^1)$
4. Define the magic point $t^1 := \underset{x \in \bar{\Omega}}{\operatorname{argmax}} |\xi_1(x)|$
5. Define a set to collect the magic point $T_1 = \{t^1\}$
6. Define the basis function $h_1(x) := \frac{\xi_1(x)}{\xi_1(t^1)} \Rightarrow X_1^{EIM} = \operatorname{span}\{h_1\}$

Now, this is an iterative method, hence we have to define the q-th step

Iterative orthogonalization steps

7. We have $S_q = \{\mu_{EIM}^1, \dots, \mu_{EIM}^q\}$, $\{\xi_1(x), \dots, \xi_q(x)\}$, $T_q = \{t^1, \dots, t^q\}$, $X_q^{EIM} = \{h_1(x), \dots, h_q(x)\}$
8. Compute new parameter $\mu_{EIM}^{q+1} := \underset{\mu \in \mathcal{P}}{\operatorname{argmax}} \| g(\cdot,\mu) - (\mathcal{I}_q^x g)(\cdot,\mu) \|_{L^\infty(\Omega)}$
9. Enlarge $S \longrightarrow \xi_{q+1}(x) := g(x, \mu_{EIM}^{q+1}), \ S_{q+1} := S_q \cup \{\mu_{EIM}^{q+1}\}$
10. Update the magic point $\longrightarrow t^{q+1} := \underset{x \in \bar{\Omega}}{\operatorname{argmax}} |\xi_{q+1}(x) - (\mathcal{I}_q^x \xi_{q+1})(x)|$ the parameter is fixed
11. Update the basis $h_{q+1}(x) := \frac{\xi_{q+1}(x) - (\mathcal{I}_q^x \xi_{q+1})(x)}{\xi_{q+1}(t^{q+1}) - (\mathcal{I}_q^x \xi_{q+1})(t^{q+1})}$

12. Than $X_{q+1}^{EIM} := X_q^{EIM} \cup \text{span}\{h_{q+1}\}$

13. Hence, the stopping criteria is $\max_{\mu \in P} \| g(x,\mu) - (\mathcal{I}_q^x g)(x,\mu) \|_{L^\infty(\Omega)} < \varepsilon_{tol}$

How fast this approximation converges to the real function?

This property holds

$$\sup_{\mu \in P} \| g(\cdot,\mu) - (\mathcal{I}_q^x g)(\cdot,\mu) \|_{L^\infty(\Omega)} < c e^{-\alpha q}$$

Is this the best method?

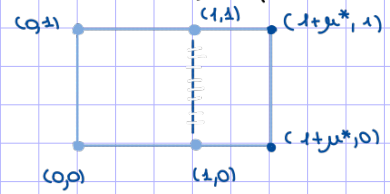
If the non-linearity is very complex, this method is NOT a good method via computation error of approximation

Indeed every time we do

- $\arg\max_{\mu \in P}$ we have to discretize the space P with $P_{train}^{EIM} \subset P$
- $\arg\max_{x \in \Omega}$ with $\bar{\Omega}_h \subseteq \Omega$

16 – GEOMETRIC PARAMETRIZATION

Let us assume $\mu \in \mathbb{P}$ one input parameter. The parameter can change the physics of the domain, but also the geometry. The μ-dependent domain is $\tilde{\Omega}(\mu)$ that is called **original domain**



To solve the problem we need to define a **reference domain** Ω that is μ-indipendent! The original domain Ω is obtained by means of a parametric map $\Phi: \Omega \times \mathbb{P} \to \tilde{\Omega}$ and namely $\tilde{\Omega}(\mu) = \Phi(\Omega; \mu) \ \forall \mu \in \mathbb{P}$

$x \longmapsto \tilde{x}(\mu)$

Let us specify the nature of μ: $\mu = (\mu_1^{ph}, \dots, \mu_p^{ph}, \mu_g)$ where μ_g is the only parameter interacting with the domain.

Our PDE problem will be:

> given $\mu \in \mathbb{P}$, find $\tilde{u}(\mu) \in \tilde{V}(\mu_g)$: $\tilde{a}(\tilde{u}(\mu), v; \mu) = \tilde{f}(v; \mu) \quad \forall v \in \tilde{V}(\mu_g)$

Thanks to the map Φ, we can pull-back the problem onto the reference domain Ω: given $\mu \in \mathbb{P}$, find $u(\mu) \in V$: $a(u(\mu), v; \mu) = f(v; \mu) \quad \forall v \in V$

⚠ The two problems are different! There are some geometric factors induced by the map Φ

Let us recall that $x \in \Omega$ and $\tilde{x} \in \tilde{\Omega}(\mu)$. Moreover, both the subdomains are subsets of $\mathbb{R}^d$. We can define the JACOBIAN MATRIX of Φ, $J_\Phi(x, \mu) \in \mathbb{R}^{d,d}$ as

$$[J_\Phi(x,\mu)]_{ke} = \frac{\partial \Phi_k(x,\mu)}{\partial x_e}, \quad k,e = 1, \dots, d$$

If for any $\mu \in \mathbb{P}$ the determinant $|J_\Phi(x;\mu)| \neq 0$, then Φ is well-posed! For any $\tilde{\Psi}: \tilde{\Omega} \to \mathbb{R}$ it holds:

$$\int_{\tilde{\Omega}(\mu)} \tilde{\Psi}(\tilde{x})\, d\tilde{x} = \int_\Omega \Psi(x)\, |J_\Phi(x;\mu)|\, dx, \quad \text{where } \Psi = \tilde{\Psi} \circ \Phi$$

16.1 Affine transformation

We will always assume Φ being an affine map of the form

$$\Phi(x, \mu) = A(\mu) x + c(\mu) \quad \text{with } A(\mu) \in \mathbb{R}^{d,d} \text{ and } c(\mu) \in \mathbb{R}^d$$

In this case $J_\Phi(x;\mu) = A(\mu)$ and the determinant depends only on the parameter μ. If Φ is well-posed, $\Phi^{-1}: \tilde{\Omega} \to \Omega$ is defined.

Recall: $\Omega = \Phi^{-1}(\tilde{\Omega}(\mu); \mu)$ and $J_{\Phi^{-1}}(\tilde{x}; \mu) = (J_\Phi(x;\mu))^{-1}$ for $\tilde{x} = \Phi(x;\mu)$

↳ by means of the inverse function theorem:

Φ continuous and non-singular in x than Φ is invertible in a neighborhood of x and $J_{\Phi^{-1}} \circ \Phi = (J_\Phi)^{-1}$

Namely, the determinant verifies: $|J_{\Phi^{-1}}(\tilde{x}, \mu)| = \dfrac{1}{|J_\Phi(\tilde{x};\mu)|}$.

More details in Parametric Geometry with additional examples

17 — OPTIMAL CONTROL PROBLEMS

In this setting the PDE is a state system needs to be OPTIMIZED with some "external help", i.e. other variables called CONTROLS

The goal of OCP is minimize an objective functional to reach beneficial configuration (related to known data)

INPUT (data of the PDE) ⟶ FUNCTIONAL TO MINIMIZE ⟶ OPTIMAL STATE

Then, our problem is: given a control $u \in U$ (control, no more a snapshot) with U a Hilbert space, the state variable $y(u) \in Y$, with Y an Hilbert space, verifies the strong form of the controlled PDE

(1) $G(y(u), u; \mu) = 0$ in Y' with $\mu :=$ physic or geometric parameter

The main goal is to minimize over $Y \times U$

$J : Y \times U \times P \longrightarrow \mathbb{R}$ defined as

$$J(y(u), u; \mu) = \frac{1}{2} \|y(u) - y_d\|^2_{Y_d} + \frac{\alpha}{2} \|u\|^2_U$$

such that the strong form holds ⟶ The function is quadratic to garantee the uniqueness

where
- $y_d \in Y_d \supset Y$ is the observation (Y_d is bigger than Y because you required less property)
- $\alpha > 0$ is the penalization parameter ⟶ $\alpha :=$ small: big control; $\alpha :=$ big: small control

In an abstract way, an OCP can be recast as

$$\text{find } u^* := \underset{u \in U}{\operatorname{argmin}} J(y(u), u; \mu) \quad \text{s.t. } G(y(u), u; \mu) = 0$$

$u^* :=$ the optimal control

$y^*(u^*) :=$ the optimal state

fixing a parameter, if you change μ you have to recompute everything

From now on, we work with linear PDEs:

$$G(y(u), u; \mu) = K(\mu) y - C(\mu) u - f(\mu) = 0$$

where
- $C(\mu) : U \longrightarrow Y'$, continuous
- $K(\mu) : Y \longrightarrow Y'$, continuous, positive definite (coercive problem)
- $f(\mu) \in Y'$

The related bilinear form are
- $a : Y \times Y \longrightarrow \mathbb{R}$
- $c : U \times Y \longrightarrow \mathbb{R}$
- $f : Y \longrightarrow \mathbb{R}$

and the weak formulation of (1) is

(2) $a(y, v; \mu) - c(u, v; \mu) - f(v; \mu) = 0 \quad \forall v \in Y$

But also the functional can be written in "weak form", rewriting the norm into integral

$$J(y, u; \mu) = \frac{1}{2} m(y - y_d, y - y_d; \mu) + \frac{\alpha}{2} n(u, u; \mu)$$

where
- $m : Y_d \times Y_d \longrightarrow \mathbb{R}$ is Y_d-norm product
- $n : U \times U \longrightarrow \mathbb{R}$ is U-inner product

Example. $Y_d = L^2(\Omega_o)$ $\Omega_o \subseteq \Omega$, $U = L^2(\Omega_u)$ $\Omega_u \subseteq \Omega$

Then J in strong form is $J(y, u; \mu) = \frac{1}{2} \|y - y_d\|^2_{L^2(\Omega_o)} + \frac{\alpha}{2} \|u\|^2_{L^2(\Omega_u)}$

$\|y - y_d\|^2_{L^2(\Omega_o)} = (y - y_d, y - y_d)_{L^2(\Omega_o)} = \int_{\Omega_o} (y - y_d)(y - y_d)\, dx$

What happen if I add another term? Consider $J(y, u; \mu) = \frac{1}{2} m(y - y_d, y - y_d; \mu) + \frac{\alpha}{2} n(u, u; \mu) - \frac{1}{2} m(y_d, y_d)$, but the solution remain the same

To solve I use **The Lagrangian Approach**

Let us define $p \in Y$ (same regularity of y) and define the lagrangian functional:

③ $$\mathcal{L}(y,u,p;\mu) : Y \times U \times Y \times P \longrightarrow \mathbb{R}$$
$$(y,u,p;\mu) \longmapsto J(y,u;\mu) + a(y,p) - c(u,p) - f(p)$$

testing the weak formulation on p

↓ in strong formulation

$$\underbrace{\langle K(\mu)y - C(\mu)u - f(\mu), p\rangle_Y}_{\in Y'}$$

where p is the **adjoint variable** and the role is the Lagrangian multiplier

We can derive the necessary conditions differentiating the Lagrangian and imponing the quantities to zero.

Explicitly,

$$\begin{aligned}\mathcal{L}(y,u,p;\mu) &= \tfrac{1}{2}m(y-y_d, y-y_d) + \tfrac{\alpha}{2}n(u,u) + a(y,p;\mu) - c(u,p;\mu) - f(p;\mu) - \tfrac{1}{2}m(y_d,y_d)\\ &= \tfrac{1}{2}m(y,y) - m(y_d,y) + \tfrac{1}{2}m(y_d,y_d) + \tfrac{\alpha}{2}n(u,u) + a(y,p;\mu) - c(u,p;\mu) - f(p;\mu) - \tfrac{1}{2}m(y_d,y_d)\\ &= \tfrac{1}{2}m(y,y) - m(y_d,y) + \tfrac{\alpha}{2}n(u,u) + a(y,p;\mu) - c(u,p;\mu) - f(p;\mu)\end{aligned}$$

The conditions are related to the derivatives:

① $D_y\mathcal{L}(y,u,p;\mu)[q] = \tfrac{1}{2}m(q,y) + \tfrac{1}{2}m(y,q) - m(y_d,q) + a(q,p;\mu)$ → all the other terms does not depend on y

direction of differentiation

This is the way to variational differentiation $y \cdot y = 1 \cdot y + y \cdot 1$ (Identity), but we compute on q so that is the reason why a put q in $m(q,y)$ or $m(y,q)$

$= m(y,q) - m(y_d,q) + a(q,p;\mu)$ → $a(q,p;\mu)$ the test function is on the left, this is not good, we have to switch

L^2 product between y & q

If the operator is not symmetric, we have to built the adjoint operator of a to put q on the right

Hence

$D_y\mathcal{L}(y,u,p;\mu)[q] = m(y-y_d, q) + a^*(p,q;\mu)$ with a^* the adjoint operator, i.e. $a^*(p,q) = a(q,p)$

Example. Linear operators $\langle Aq, p\rangle = \langle q, A^T p\rangle \longleftrightarrow a(q,p) = a^*(p,q)$

i.e. $A := \text{grad}\cdot\text{grad} \longrightarrow \int_\Omega \nabla q \nabla p \, dx = \int_\Omega \nabla p \nabla q \, dx \quad a(p,q) = a(q,p)$ symmetric

$A := \text{convection} \longrightarrow \int_\Omega \frac{\partial q}{\partial x} p \, dx = \cancel{\int_\Omega \frac{d(qp)}{dx}} - \int_\Omega \frac{\partial p}{\partial x} q \, dx = a^*(q,p) \neq a(p,q)$

$= 0$ for boundary condition

② $D_u\mathcal{L}(y,u,p;\mu)[v] = \frac{\alpha}{2}n(v,u) + \frac{\alpha}{2}n(u,v) - c(v,p) = \alpha n(u,v) - c(v,p) = \alpha n(u,v) - c^*(p,v) = 0$

③ $D_p\mathcal{L}(y,u,p;\mu)[z] = a(y,z;\mu) - c(u,z;\mu) - f(z) = 0$

we obtain the constrain equal to zero, hence we obtain the week formulation

The minimum has reached at the stationary point of the Lagrangian, imposing

$$\begin{cases} D_y\mathcal{L}(y,u,p;\mu)[q] = 0 & \forall q \in Y \quad \text{adjoint equation}\\ D_u\mathcal{L}(y,u,p;\mu)[v] = 0 & \forall v \in U \quad \text{optimality equation}\\ D_p\mathcal{L}(y,u,p;\mu)[z] = 0 & \forall z \in Y \quad \text{state equation}\end{cases}$$

Namely

$$\begin{cases} m(y,q) + a^*(p,q;\mu) = m(y_d,q) & \forall q \in Y\\ \alpha n(u,v) - c^*(p,v;\mu) = 0 & \forall v \in U\\ a(y,z;\mu) - c(u,z;\mu) = f(z,\mu) & \forall z \in Y\end{cases}$$

Hence, matrix wise (FOM)

$$\begin{cases} M_\delta(\mu)y + K_\delta^T(\mu)p = M_\delta(\mu)y_d & \text{finite element matrix}\\ \alpha N_\delta(\mu)u - C_\delta^T(\mu)p = 0\\ K_\delta(\mu)y - C_\delta(\mu)u = f_\delta(\mu)\end{cases}$$

Algebrically, in compact form (observation vector)

$$\begin{bmatrix} M_\delta(\mu) & 0 & K_\delta^T(\mu) \\ 0 & \alpha N_\delta(\mu) & -C_\delta^T(\mu) \\ K_\delta(\mu) & -C_\delta(\mu) & 0 \end{bmatrix} \begin{bmatrix} y \\ u \\ p \end{bmatrix} = \begin{bmatrix} M_\delta(\mu) y_d \\ 0 \\ f_\delta \end{bmatrix}$$

Saddle-point structure. Consider the problem that is in this structure, we can write

$$\begin{bmatrix} A(\mu) & B^T(\mu) \\ B(\mu) & 0 \end{bmatrix} \begin{bmatrix} z \\ p \end{bmatrix} = \begin{bmatrix} g_\delta \\ f_\delta \end{bmatrix} \quad \text{where} \quad z = \begin{bmatrix} y \\ u \end{bmatrix}, \quad g_\delta = \begin{bmatrix} M_\delta(\mu) y_d \\ 0 \end{bmatrix}$$

- $A(\mu)$ is always well-poseness
- For $B(\mu)$, the well-poseness is related to the inf-sup stability and it is provable if $p \in Y$

↓

ROM level. Collect three sets of snapshots and build three different basis matrices $\mathbb{B}_y, \mathbb{B}_u, \mathbb{B}_p$

- y_N spanned by $\mathbb{B}_y$
- u_N spanned by $\mathbb{B}_u$
- p_N spanned by $\mathbb{B}_p \neq B_y$ ⟶ How can I recover the inf-sup aggregated spaces?

We define $\mathbb{B}_{yp} = [\mathbb{B}_y \,|\, \mathbb{B}_p]$ the **aggregated space** and consider y_N, p_N spanned by $\mathbb{B}_{yp}$

↓

In order to garantee well-poseness, we need $\mathcal{N}$ basis wit N basis without control!

Once we build the basis, assuming affinity, we can pre-compute

$$M_N = \mathbb{B}_{yp}^T M_\delta \mathbb{B}_{yp}$$
$$N_N = \mathbb{B}_u^T N_\delta \mathbb{B}_u$$
$$K_N = \mathbb{B}_{yp}^T K_\delta \mathbb{B}_{yp}$$
$$C_N = \mathbb{B}_{yp}^T C_\delta \mathbb{B}_u$$

and solve

$$\begin{pmatrix} M_N & 0 & K_N^T \\ 0 & \alpha N_N & C_N^T \\ K_N & C_N & 0 \end{pmatrix} \begin{pmatrix} y_N \\ u_N \\ p_N \end{pmatrix} = \begin{pmatrix} M_N y_d^N \\ 0 \\ f_N \end{pmatrix}$$

where $y_d^N = \mathbb{B}_{yp}^T y_d$, $f_N = \mathbb{B}_{yp}^T f_\delta$

Ex: A very simple practicle example

Consider $\Omega = [0,1] \times [0,1]$, $y_d \in L^2(\Omega_0)$, $\Omega_0 \subseteq \Omega$, $\mu \in \mathcal{P} \subset \mathbb{R}$ $\alpha > 0$

Given a $\mu \in \mathcal{P}$, find $(y, u) \in H_0^1(\Omega) \times L^2(\Omega)$ s.t. they verify

$$\min_{(y,u)} \frac{1}{2} \| y - y_d \|^2_{L^2(\Omega_0)} + \frac{\alpha}{2} \|u\|^2_{L^2(\Omega)}$$

$$\text{s.t.} \begin{cases} -\mu \Delta y = u + f & \text{in } \Omega \\ y = 0 & \text{in } \partial\Omega \end{cases}$$

Let us define $p \in H_0^1(\Omega)$ and the Lagrangian

$$\mathcal{L}(y,u,p;\mu) = \frac{1}{2}\int_{\Omega_0} (y - y_d)^2 \, dx + \frac{\alpha}{2}\int_\Omega u^2 dx + \underbrace{\mu \int_\Omega \nabla y \nabla p \, dx}_{a(y,p)} \underbrace{- \int_\Omega up \, dx}_{-c(u,p)} - \int_\Omega fp \, dx$$

Now we have to differenciate

$$D_y \mathcal{L}(y,u,p;\mu)[q] = \int_{\Omega_0} (y - y_d) q \, dx + \mu \int_\Omega \nabla q \nabla p \, dx \quad \text{(self adjoint)}$$

$$D_u \mathcal{L}(y,u,p;\mu)[v] = \alpha \int_\Omega uv \, dx - \int_\Omega pv \, dx$$

$$D_p \mathcal{L}(y,u,p;\mu)[z] = \mu \int_\Omega \nabla y \nabla z \, dx - \int_\Omega uz \, dx - \int_\Omega fz \, dx \quad \text{(self adjoint)} \longrightarrow \text{This is the variational formulation}$$

⚠ If you have an advection term $a(y,z;\mu) = \int_\Omega \beta \nabla y \, z \, dx = \int_\Omega \text{div}(\beta y z) \, dx - \int \beta \nabla z y \, dx = \int_{\partial\Omega} \beta y z n \, ds - \int_\Omega \beta \cdot \nabla z y \, dx$

$= 0$ b.c.

$= a^*(z,y;\mu)$