EDP elliptiques et méthodes neuronales

Emmanuel Franck[*],

12 novembre 2024
Master CMSI, M2, Strasbourg

[*]MACARON project-team, Université de Strasbourg, CNRS, Inria, IRMA, France

Outline

Introduction aux PINNS et Deep Ritz

Objectifs

EDP elliptiques linéaires

On considérera ici des EDP elliptiques et linéaires de la forme:

$$\begin{cases} L(u(\mathbf{x})) = -\nabla \cdot (A(\mathbf{x})\nabla u(\mathbf{x})) + \nabla \cdot (\beta(\mathbf{x})u(\mathbf{x})) + c(\mathbf{x})u(\mathbf{x}) = f(\mathbf{x}), & \forall \mathbf{x} \in \Omega \subset \mathbb{R}^d \\ u(\mathbf{x}) = 0, & \forall \mathbf{x} \in \partial\Omega \end{cases}$$

Numérique Vs Apprentissage

L'apprentissage comme les méthodes numériques cherchent a construire des approximations de fonctions.
Dans les deux cas on utilise des fonctions paramétriques. L'un contrains par les données, l'autre par l'équation physique.

Idée

Utiliser les réseaux de neurones comme modèles paramétriques dans les méthodes numériques.

Espace d'approximation

Linéaires

- Espace d'approximation:

$$V_n = \left\{ \sum_{i=1}^{N} \theta_i \phi_i(\mathbf{x}), \quad \theta \in V \subset \mathbb{R}^n \right\}$$

- Opérateur de restriction $\mathcal{R}$:

$$\theta^* = \min_{\theta} \int_{\Omega} \mid u(\mathbf{x}) - \langle \theta, \mathbf{\Phi}(\mathbf{x}) \rangle \mid^2 dx,$$

- Lorsqu'on résout ce problème on obtient:

$$M\theta = b(u)$$

$M = \int_{\Omega} \Phi(\mathbf{x}) \otimes \Phi(\mathbf{x}) d\mathbf{x}, \quad b(u) = \int_{\Omega} u(\mathbf{x}) \Phi d\mathbf{x}$

- Opérateur de reconstruction $\mathcal{I}$:

$$\mathcal{I}(u) = \langle \theta^*, \mathbf{\Phi}(\mathbf{x}) \rangle = \sum_{i=1}^{N} \theta_i^* \phi_i(x)$$

- Projecteur: $\Pi_{V_n} = \mathcal{I} \circ \mathcal{R}$

Nonlinéaires

- Espace d'approximation:

$$M_{L,n} = \{ nn_\theta(\mathbf{x}), \quad \theta \in V \subset \mathbb{R}^n \}$$

un réseau à L couche et n neurones.

- Opérateur de restriction $\mathcal{R}$:

$$\theta^* = \min_{\theta} \int_{\Omega} \mid u(\mathbf{x}) - nn_\theta(\mathbf{x}) \mid^2 dx,$$

- Opérateur de reconstruction $\mathcal{I}$:

$$\mathcal{I}(u) = nn_{\theta^*}(\mathbf{x})$$

- Projecteur: $\Pi_{V_n} = \mathcal{I} \circ \mathcal{R}$
- Propriété du projecteur ?

Espace d'approximation II

Espace linéaire

- On choisit $f_1, f_2 \in V_n$:

$$f_1(\mathbf{x}) + f_2(\mathbf{x}) = \sum_{i=1}^{N} \theta_i \phi_i(\mathbf{x}) \in V_n$$

- V_n est un espace vectoriel.

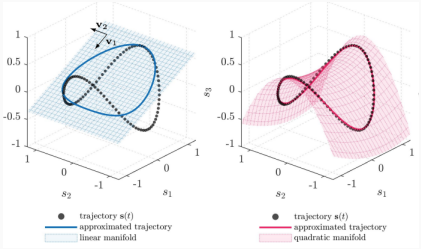
Espace nonlinéaire

- On choisit $f_1, f_2 \in M_{L,n}$:

$$f_1(\mathbf{x}) + f_2(\mathbf{x}) \not\subset M_{L,n}$$

- $M_{L,n}$ n'est pas espace vectoriel mais une variété

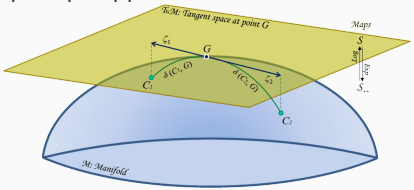
- Espace vectoriel Vs Variété



- Comment se comporte la projection sur une sous-variété.

Espace d'approximation III

- **Sous-Variété**: il s'agit d'un espace courbe (généralisant les surfaces en dimension n, qui se comporte localement comme un espace plat.
- **Espace tangent**: Il est associé a un point $\mathbf{x}$ de la variété.
  On peut le voir comme espace plat qui approche la variété localement autour de ce point.



- **Hyperplan**: $V = \{\mathbf{x}, \quad \text{tel que } \langle \mathbf{x}, \mathbf{a} \rangle = 0\}$ pour un $\mathbf{a} \in \mathbb{R}^d$ donné.
- **Sous-variété** gentille : $\mathcal{M} = \{\mathbf{x}, \quad \text{tel que } F(\mathbf{x}) = 0\}$.

Projecteur

- Est ce que le projecteur sur un espace vectoriel est unique ? oui, par le théorème de projection sur un convexe dans un Hilbert.
- Est ce que le projecteur sur une sous-variété est unique ? non. Exemple: la sphère.

Espace d'approximation IV

Exemples d'espaces linéaires

- Spectrale Fourier (globale):
$$f(\mathbf{x}) = \sum_{i=k}^{n} \alpha_k \sin(2k\pi x)$$
- Spectrale Polynômes orthogonaux (globale):
$$f(\mathbf{x}) = \sum_{i=k}^{n} \alpha_k P_k(\mathbf{x})$$
- Element finis (locale):
$$f(\mathbf{x}) = \sum_{i=k}^{n} \alpha_k \phi_{h,k}(\mathbf{x})$$
avec $\phi_{h,k}$ une fonction affine par morceaux.
- Base radiales:
$$f(\mathbf{x}) = \sum_{i=k}^{n} \alpha_k \phi(\epsilon \mid \mathbf{x} - \mathbf{x}_i \mid)$$
avec $\phi(r) = e^{-r^2}$, $\phi(r) = \sqrt{(1+r^2)}$.
- Réseaux aléatoires:
$$f(\mathbf{x}) = \sum_{i=k}^{n} \alpha_k nn_{\theta_k}(x)$$
ou les θ_k sont choisies aléatoirement.

Exemples d'espaces nonlinéaires

- Méthodes de tenseur:
$$f(\mathbf{x}) = \sum_{i=1}^{r} \left(\sum_{k=1}^{n} \alpha_{i,k} \phi_k(\mathbf{x}_1) \right) \left(\sum_{k=1}^{n} \beta_{i,k} \phi_k(\mathbf{x}_2) \right)$$
avec $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$.
- Spectrale Fourier (globale):
$$f(\mathbf{x}) = \sum_{i=k}^{n} \alpha_k \sin(2\omega_k \pi x)$$
- Base radiales (locale):
$$f(\mathbf{x}) = \sum_{i=k}^{n} \alpha_k \phi(\epsilon_k \mid \mathbf{x} - \mathbf{x}_i \mid)$$
- Base radiales anisotropes (globale):
$$f(\mathbf{x}) = \sum_{i=k}^{n} \alpha_k \phi(\mid \Sigma_k^{-1}(\mathbf{x} - \mathbf{x}_i) \mid)$$
- Réseaux de neurones type MLP (globale):
$$f(\mathbf{x}) = nn_\theta(\mathbf{x})$$
- Réseaux de neurones type KAN (globale):
$$f(\mathbf{x}) = kan_\theta(\mathbf{x})$$

Intégration

- L'intégration va dépendre de choix de l'espace. Pour les méthodes linéaires et tensorielles on choisira des quadratures locales (dans chaque maille d'un maillage) ou globales sur le domaine selon le type de base.
- On va t'intéresser ici au cas des espaces nonlinéaires notamment basés sur des réseaux de neurone dont les caractéristiques sont:
  - Modèles globales qui ne nécessite pas de maillage.
  - Bonne capacité d'approximation en grande dimension.
  - Précision limitée.

Méthode d'integration

La méthode d'intégration la plus adaptée est la méthode de Monte Carlo.

$$\int_\Omega \|u_\theta(\mathbf{x}) - u(\mathbf{x})\|_2^2 dx = \mathbb{E}_{\mathcal{U}(\Omega)}[\|u_\theta(\mathbf{x}) - u(\mathbf{x})\|_2^2]$$

avec $\mathcal{U}(\Omega)$ une loi uniforme sur Ω. En appliquant la loi des grand nombres, on a

$$\int_\Omega \|u_\theta(\mathbf{x}) - u(\mathbf{x})\|_2^2 d\mathbf{x} \approx \frac{1}{N}\sum_{i=1}^{N} \|u_\theta(\mathbf{x}_i)) - u(\mathbf{x}_i)\|_2^2$$

Intégration et géométrie complexe

- On va donc résoudre en pratique

$$\theta^* = \int_\Omega \| u_\theta(\mathbf{x}) - u(\mathbf{x}) \|_2^2 \, d\mathbf{x}$$

- Comment échantillonner les points $\mathbf{x}_i$?
- Géométries tensorisées:
  - $\Omega = [a_1, b_1] \times \ldots \times [a_d, b_d]$. On utilise une loi uniforme dans chaque direction.
  - Le disque ou la sphère. On passe en coordonnées radiales ou sphériques et on utilise une loi uniforme pour chaque coordonnée.
  - Evidemment si Ω est une somme/différence de géométries tensorisées on peut facilement échantillonner des points dedans.
- Géométries complexes ?

Méthode de "mapping"

On suppose que $\Omega = m(\Omega_0)$ avec Ω_0 une géométrie simple. Dans ce cadre il suffit d'échantillonner Ω_0 et d'appliquer m a chaque point.

  - On peut apprendre le mapping $m(\mathbf{x})$

Fonction Niveau

Soit une domaine Ω de frontière Γ, on appelle une fonction niveau une fonction ϕ tel que

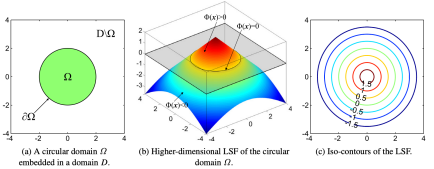
$$\phi(x) = \begin{cases} < 0, & \mathbf{x} \in \Omega \\ = 0, & \mathbf{x} \in \Gamma \\ > 0, & \mathbf{x} \in \mathbb{R}^d / \Omega \end{cases}$$

- Comment échantillonner ?
  - On tire un point aléatoirement dans $[a, d]^d$ tel que Ω soit inclus.
  - Si $\phi(\mathbf{x}) < 0$ on garde le point sinon on recommence.
- Pas d'unicité des fonctions niveaux. Exemple le disque:

$$\phi_1(\mathbf{x}) = \sqrt{x_1^2 + x_2^2} - r, \quad \phi_1(\mathbf{x}) = x_1^2 + x_2^2 - r^2$$

- La première est appelée La fonction distance signée car elle donne la distance entre chaque point et Γ. Il s'agit d'une fonction C^0 mais non C^1.

- **Somme de domaines**: $\phi_1(\mathbf{x}) < 0$ ou $\phi_2(\mathbf{x}) < 0$
- **Intersection de domaines**: $\phi_1(\mathbf{x}) < 0$ et $\phi_2(\mathbf{x}) < 0$
- **Domaines a trous**: $\phi_{domaine}(\mathbf{x}) < 0$ et $\phi_{trou}(\mathbf{x}) > 0$



(a) A circular domain Ω embedded in a domain D. (b) Higher-dimensional LSF of the circular domain Ω. (c) Iso-contours of the LSF.

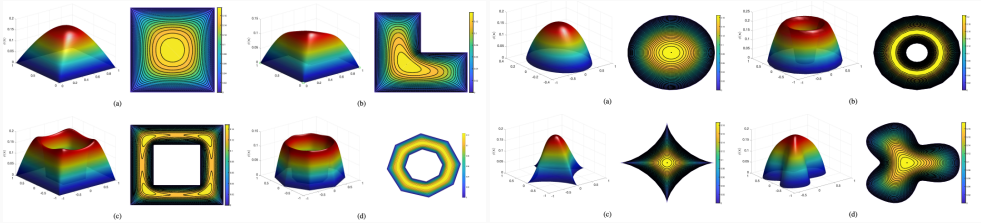
Intégration et géométrie complexe III

- Et si on ne connaît pas la fonction niveau ? On la calcule ou on l'apprend.
- Approximation de la fonction distance dans le cas polygonal :
  - On calcul

$$W(\mathbf{x}) = \sum_{i=1}^{n} \left(\frac{1}{r_i} + \frac{1}{r_{i+1}} \right) t_i, \quad t_i := \frac{\det(\mathbf{r}_i, \mathbf{r}_{i+1})}{r_i r_{i+1} + \mathbf{r}_i \cdot \mathbf{r}_{i+1}},$$

    avec $\mathbf{r}_i = \mathbf{x}_i - \mathbf{x}$ et $r_i = \|\mathbf{r}_i\|$.
  - L'approximation est donnée par $\phi(\mathbf{x}) = \frac{2}{W(\mathbf{x})}$.
- Approximation de la fonction distance dans le cas ou le bord est donné par une courbe paramétrique $\mathbf{c}(t)$:

$$\phi(\mathbf{x}) = \left(\frac{1}{W_p(\mathbf{x})} \right)^{1/p}, \quad W_p(\mathbf{x}) = \int_0^1 \frac{(\mathbf{c}(t) - \mathbf{x}) \cdot \mathbf{c}'^{\perp}(t)}{\|\mathbf{c}(t) - \mathbf{x}\|^{2+p}} dt$$



Intégration et géométrie complexe IV

- Comment apprendre une fonction niveau ? On a une équation pour la fonction distance signée appelée l'équation Eikonal.

$$\| \nabla \phi \| = 1, \quad \forall x \in \Omega$$

- On soit aussi qu'elle est nulle au bord et que le gradient de la fonction distance définie la normale au bord.
- Comment la résoudre ? Avec une méthode basée sur les réseaux (PINNs).
- On se donne des paires $(\mathbf{x}_b, \mathbf{n}_b)$ aux bords. On résout:

$$\min_{\phi \in C^0} (J_{eq}(\phi) + J_{di}(\phi) + J_{Neu}(\phi))$$

avec

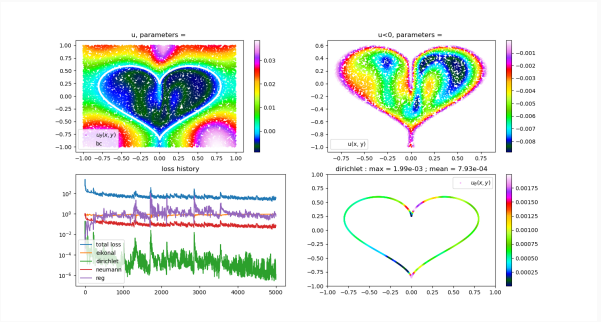
$$J_{eq}(\phi) = \int_{[a,b]^d} |\| \nabla \phi \| - 1|^2 \, d\mathbf{x},$$

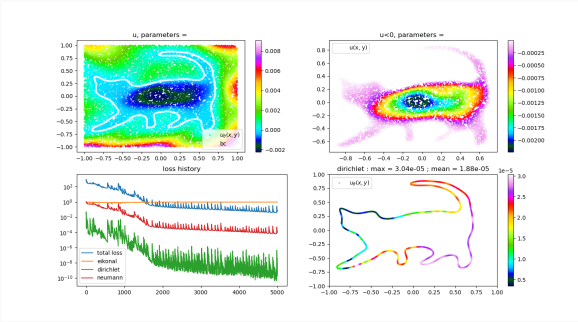
approchée par une méthode de Monte-Carlo et

$$\sum_{i=1}^{N_b} \| \phi(\mathbf{x}_i) \|^2, \quad \sum_{i=1}^{N_b} \| \nabla \phi(\mathbf{x}_i) - \mathbf{n}_b \|^2$$

Intégration et géométrie complexe IV

- Exemples:





- On connait les courbes paramétriques. On échantillonne des points et leurs normales sur le bord.
- Simulations de F. Lecourtier (Ex CSMI).

Adaptation

- Dans les méthodes d'approximation classiques on modifie souvent la discrétisation en raffinant le maillage ou la quadrature localement pour améliorer la précision.
- Comment faire ici? Repartons de la méthode Monte-Carlo

$$\int_\Omega f(x) = \mathbb{E}_p[f(X)] = \int_{\mathbb{R}^d} f(x)p(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i) = \bar{X}_N$$

avec $p(x) = 1_{\{x\in\Omega\}}$ une densité de probabilité uniforme.

- Les conséquence du théorème centrale limite nous dit

$$\mathbb{E}\left[\left(\bar{X}_N - \mathbb{E}_p[f(X)]\right)^2\right] = \frac{Var(f(X))}{N}$$

avec $Var(p)$ la variance de la loi associée à $p(x)$.

- Méthode de l'échantillonnage préférentielle.

$$\mathbb{E}_p[f(X)] = \int_{\mathbb{R}^d} f(x)p(x)dx = \int_{\mathbb{R}^d} \frac{f(x)p(x)}{\tilde{p}(x)}\tilde{p}(x) = \mathbb{E}_{\tilde{p}}\left[\frac{f(X)p(X)}{\tilde{p}(X)}\right]$$

- Si $Var(\frac{f(X)p(X)}{\tilde{p}(X)}) < Var(F(X))$ on réduit l'erreur. On a envi de prendre $\tilde{p}(X) \approx f(X)p(X)$.

Adaptation II

- On pose $r(\mathbf{x}) = u_\theta(x) - u(x)$
- On repart du résidu a minimiser et on applique la même stratégie :

$$\int_\Omega \|u_\theta \mathbf{x} - u(\mathbf{x})\|_2^2 d\mathbf{x} = \int_\Omega \frac{|\, r(\mathbf{x})\,|^2}{g_\theta(x)} g_\theta(x) d\mathbf{x} = \mathbb{E}_{g_\theta}\left[\frac{|\, r(\mathbf{x})\,|^2}{g_\theta(x)}\right]$$

- On souhaite donc déterminer $g_\theta(x) \approx |\, r(\mathbf{x})\,|^2$

Remarque

Puisqu'on souhaite minimiser $|\, r(\mathbf{x})\,|^2$ et puisque $g_\theta(x) > 0$ on est pas obligé de minimiser

$$\mathbb{E}_{g_\theta}\left[\frac{|\, r(\mathbf{x})\,|^2}{g_\theta(x)}\right]$$

On peut juste minimiser

$$\mathbb{E}_{g_\theta}\left[|\, r(\mathbf{x})\,|^2\right] = \int_\Omega |\, r(\mathbf{x})\,|^2\, g_\theta(x) d\mathbf{x}$$

si $g_\theta(\mathbf{x}) \approx |\, r(\mathbf{x})\,|^2$, les zones de haut résidu seront encore plus importantes dans l'intégrale et donc seront minimisées en priorités.

- Comment calculer g_θ ? Plusieurs approches.

Adaptation III

- En général
- Propriétés des méthodes:
  - **Explicite/implicite**: selon elles construisent explicitement g_θ ou échantillonne en mimant l'effet de g_θ.
  - **Re-échantillonnne (Re/Non-Re)**: on dit qu'une méthode re-échantillonne si on peut retirer l'ensemble des points à chaque fois à l'inverse de celles qui piochent une portion de points dans un ensemble pré-déterminer S_0.
  - **Pondéré (P/non-P)**: si elle divise le résidu par $g_\theta(\mathbf{x})$.
- Méthode RAR
  - Après k itération on sélectionne m points de plus aux résidus et on les ajoute à l'ensemble des points d'échantillonnages S_0.
  - Méthode implicite, non-Re, non-P.
- Méthode RAD
  - On construit $g_\theta(\mathbf{x}) = \frac{\|r(\mathbf{x})\|^2}{\sum_{\mathbf{x}_i \in S_0} \|r(\mathbf{x}_i)\|^2}$ avec S l'ensemble des points d'origine.
  - Après k itération on ré-échantillonne suivant $g_\theta(\mathbf{x})$
  - Méthode explicite, non-Re, non-P.
- On peut rendre les méthodes plus efficaces en appliquant cette méthode par sous-domaine.
- Le nombre de sous-domaines peut devenir grand avec la dimension.

Adaptation IV

- Méthode DAS/ASMM
  - On construit $g_{\theta_g}(\mathbf{x})$ un modèle génératif (modèles paramétriques pour estimer les lois de probabilités, cours fin semestre) en résolvant
    $$\theta_g = \min_{\theta} D_{KL}(p_{target}(\mathbf{x}) \mid\mid g_\theta)$$
    avec $p_{target}(\mathbf{x}) = \mid r(\mathbf{x}) \mid^2 1_{\{\phi(\mathbf{x})<0\}}$.
  - Toutes les k itérations, on effectue un petit apprentissage de g_{θ_g} et on ré-échantillonne les points.
  - Méthode explicite, non-Re ou Re, non-P ou P.
- Compatible avec des géométries complexes et la grande dimension.
- Détails:

  $$\theta_g = \min_{\theta} D_{KL}(p_{target}(\mathbf{x}) \mid\mid g_\theta) = \int_\Omega p_{target}(\mathbf{x}) log(p_{target}(\mathbf{x}))d\mathbf{x} - \int_\Omega p_{target}(\mathbf{x}) log(g_\theta \mathbf{x})d\mathbf{x}$$

  Puisqu'on minimise il ne reste que

  $$\theta_g = \min_{\theta} \int_\Omega p_{target}(\mathbf{x}) log(g_\theta(\mathbf{x}))d\mathbf{x} = \int_\Omega \frac{p_{target}(\mathbf{x}) log(g_\theta(\mathbf{x}))}{g_\theta(\mathbf{x})} g_\theta(\mathbf{x})d\mathbf{x}$$

  et donc après discrétisation

  $$\theta_g = \min_{\theta} \sum_{i=1}^{N} \frac{p_{target}(\mathbf{x}_i) log(g_\theta(\mathbf{x}_i))}{g_\theta(\mathbf{x}_i)} d\mathbf{x}$$

  ou les points sont échantillonner avec $g_\theta(\mathbf{x})$.

Résumé: Géométrie, intégration et adaptation

Classique

- Géométrie:
  - **Element finis** utilise des maillages. Plus récemment: frontières immergées et fonctions niveaux.
  - **Base radiales** utilise des fonctions niveaux et des échantillonnages.
- Adaptation:
  - **Élément finis**: On ajoute des mailles suivant un estimateur d'erreur. Les estimateurs dépendent du gradient (ou plus) et des éléments.

Réseaux

- Géométrie:
  - Utilise des fonctions niveaux ou des mapping et des échantillonnage de points.
- Adaptation:
  - On ajoute des points la ou l'estimateur l'indique. Souvent liés aux résidus.

Comment calculer l'opérateur de restriction

Espaces linéaires

- **Calcul du gradient**: analytique,
- **Résolution de** $\nabla J = 0$: équation normale.
  - Dans le cas linéaire:

$$\nabla J = 0 \longleftrightarrow A\theta - \mathbf{b} = 0$$

  - on résout donc un système linéaire (LU, CG, GMRES etc).
- **Calcul des dérivées des fonction de bases**: analytique

Espaces nonlinéaires

- **Calcul du gradient**: différentiation automatique
- **Résolution de** $\nabla J = 0$: descente de gradient stochastique/méthode de type Newton
- **Calcul des dérivées des fonction de bases**: différentiation automatique

Cas nonlinéaire: méthodes de Gradient et de Newton

- On veut donc déterminer θ^* solution de

$$\nabla_\theta \mathcal{J}(\theta^*) = \sum_{i=1}^{N} \nabla_\theta \mathcal{J}_i(\theta^*) = 0$$

avec $\mathcal{J}_i$ la fonction de coût locale (ici une norme L^2) a chaque échantillon

- Puisque $\mathcal{J}$ est nonlinéaire on a potentiellement plusieurs solutions.
- Le gradient est calculé par différentiation automatique.
- **Méthode de Gradient**:

$$\nabla_\theta \mathcal{J}(\theta) = 0 \iff -\eta \nabla_\theta \mathcal{J}(\theta) = 0 \iff -\eta \nabla_\theta \mathcal{J}(\theta) + \theta = \theta$$

par méthode de point fixe on a: $\theta_{k+1} = \theta_k - \eta \nabla_\theta \mathcal{J}(\theta_k)$.

- **Méthode de Newton**:

$$\nabla_\theta \mathcal{J}(\theta) = 0 \underbrace{\Longrightarrow}_{\text{linéarisation}} Jac(\nabla_\theta \mathcal{J}(\theta_0))(\theta - \theta_0) + \nabla_\theta \mathcal{J}(\theta_0) \approx \nabla_\theta \mathcal{J}(\theta) = 0$$

donc

$$H_\theta(J(\theta_k))(\theta_{k+1} - \theta_k) = -\nabla_\theta \mathcal{J}(\theta_k) \iff \theta_{k+1} = \theta_k - H_\theta^{-1}(\mathcal{J}(\theta_k)) \nabla_\theta \mathcal{J}(\theta_k)$$

avec $H_\theta(\mathcal{J}(\theta_k))$ la Hessienne de $\mathcal{J}$.

Cas nonlinéaire: méthode de. Gauss-Newton et Leverberg-Marquardt

- **Méthode de Gauss-Newton**:
  - On pose $\mathcal{J}(\theta) = \langle \boldsymbol{\mathcal{J}}(\theta), \mathbf{1} \rangle$. On propose de minimiser

$$\min_\theta \parallel \boldsymbol{\mathcal{J}}(\theta) \parallel_2^2$$

  - On linéarise autour de θ_0 ce qui donne

$$\min_\theta \parallel \partial_\theta \boldsymbol{\mathcal{J}}(\theta_0)(\theta - \theta_0) + \boldsymbol{\mathcal{J}}(\theta)(\theta_0) \parallel_2^2$$

  - Ca donne un problème aux moindre carrés sur $(\theta - \theta_0)$ qui peut définir l'incrément du méthode itérative:

$$\theta_{k+1} = \theta_k + \delta\theta_k, \quad \text{avec } \min_{\delta\theta_k} \parallel (\partial_\theta \boldsymbol{\mathcal{J}}(\theta)(\theta_k))\delta\theta_k + \boldsymbol{\mathcal{J}}(\theta_k) \parallel_2^2$$

    avec $\partial_\theta \boldsymbol{\mathcal{J}}(\theta_k)$ la Jacobienne de $\boldsymbol{\mathcal{J}}$.
  - On résout le problème aux moindre carrés par l'équation normale (cours 1):

$$\theta_{k+1} = \theta_k + \delta\theta_k, \quad \text{avec } \partial_\theta \boldsymbol{\mathcal{J}}(\theta_k)^T \partial_\theta \boldsymbol{\mathcal{J}}(\theta_k)\delta\theta_k = -\partial_\theta \boldsymbol{\mathcal{J}}(\theta_k)^T \boldsymbol{\mathcal{J}}(\theta_k)$$

- **Méthode de Leverberg-Marquardt** :
  - Parfois la matrice $\partial_\theta \boldsymbol{\mathcal{J}}(\theta_k)^T \partial_\theta \boldsymbol{\mathcal{J}}(\theta_k)$ peut être difficile à inverser.
  - On utilise donc

$$\theta_{k+1} = \theta_k + \delta\theta_k, \quad \text{avec } (\partial_\theta \boldsymbol{\mathcal{J}}(\theta_k)^T \partial_\theta \boldsymbol{\mathcal{J}}(\theta_k) + \lambda_k I_d)\delta\theta_k = -\partial_\theta \boldsymbol{\mathcal{J}}(\theta_k)^T \boldsymbol{\mathcal{J}}(\theta_k)$$

  - Si l'erreur décroit vite on baisse a chaque itération λ_k on devient proche de la méthode de Gauss-Newton.
  - Si la convergence est lente on augmente λ. Puisque $\partial_\theta \boldsymbol{\mathcal{J}}(\theta_k)^T \boldsymbol{\mathcal{J}}(\theta_k) = \nabla_\theta \parallel \boldsymbol{\mathcal{J}}(\theta) \parallel_2^2$ on se rapproche d'une méthode de gradient.

Erreur de projection: cas général

- On considère que $u \in H_x(\Omega)$ avec H_x un espace de Hilbert. On chercher calculer l'opérateur de restriction:
$$\hat{\theta} = \min_\theta \mathcal{E}(\theta) = \min_\theta \int_\Omega \parallel u_\theta - u \parallel_x^2$$
- En pratique on calcul numériquement $\theta^*(S)$ par un algorithme numérique qui minimise $\mathcal{E}_N(\theta) = \sum_{i=1}^N w_i \parallel u_\theta(\mathbf{x}_i) - u(\mathbf{x}_i) \parallel_x^2$ avec $S = \{\mathbf{x}_1, ...\mathbf{x}_N\}$.
- On chercher a estimer l'erreur du modèle calculé donc $\mathcal{E}(\theta^*(S))$.

Majoration de l'erreur

$$\mathcal{E}^* \leqslant C(\underbrace{\mathcal{E}(\widehat{\theta})}_{\text{erreur d'approximation}} + \underbrace{2\text{sup}_{\theta\in\Theta} \mid \mathcal{E}(\theta) - \mathcal{E}_N(\theta) \mid}_{\text{erreur d'intégration}} + \underbrace{\mathcal{E}_N(\theta^*) - \min_\theta \mathcal{E}_N(\theta)}_{\text{erreur d'optimisation}})$$

- On remarque facilement que $\mathcal{E}(\theta^*(S)) = \mathcal{E}(\hat{\theta}) + \Big(\mathcal{E}(\theta^*(S)) - \mathcal{E}(\hat{\theta})\Big)$
- On fait intervenir l'intégration
$$\mathcal{E}(\theta^*(S)) = \underbrace{\mathcal{E}(\hat{\theta})}_{\text{erreur d'approximation}} + \mathcal{E}(\theta^*(S)) - \mathcal{E}_N(\theta^*) + \mathcal{E}_N(\theta^*) - \mathcal{E}(\hat{\theta}) + \min_\theta \mathcal{E}_N(\theta) - \min_\theta \mathcal{E}_N(\theta)$$
- On remarque l'erreur d'optimisation est donnée par: $\mathcal{E}_N(\theta^*) - \min_\theta \mathcal{E}_N(\theta)$. Il reste, après regroupement donc
$$\mathcal{E}(\theta^*(S)) - \mathcal{E}_N(\theta^*) + \min_\theta \mathcal{E}_N(\theta) - \mathcal{E}(\hat{\theta}) \leqslant 2sup_\theta \mid \mathcal{E}(\theta) - \mathcal{E}_N(\theta) \mid$$

Erreur de projection: cas général II

Erreur d'approximation

Elle quantifie la capacité de votre espace d'approximation a approcher la fonction cible u. Si la fonction cible est dans l'espace de dimension fini choisi cette erreur est nulle.

Erreur d'intégration ou de généralisation

Puisque la quantité $\int_\Omega \| u_\theta - u \|_x^2$ est approchée par une intégrale discrète, il s'agit d'une erreur d'intégration. En apprentissage, on parle d'erreur de généralisation car les échantillons de l'intégrale discrète correspondent aux données et donc cette erreur à quel point le modèle peut traiter de nouvelle données.

Erreur d'optimisation

Il s'agit de l'erreur induite par votre algorithme de résolution (souvent optimisation).

Erreur de projection: espace linéaire

- **Erreur d'approximation**:
  - Dans le cadre des méthodes basées sur des espaces approximations linéaires, l'erreur d'approximation Peut-être déterminer analytiquement.
  - La méthode élément finis utilise la convergence de l'opérateur d'interpolation. Pour cela, on utilise des formules de Taylor.
- **Erreur d'intégration/généralisation**:
  - La méthode d'intégration est souvent choisie tel que :

$$sup_{\theta} \mid \mathcal{E}(\theta) - \mathcal{E}_N(\theta) \mid << \mathcal{E}(\hat{\theta})$$

  - En élément finis, on utilise des quadratures suffisamment précise pour intégrer exactement les polynômes utilisées dans les bases.
- **Erreur d'optimisation**:
  - Cette erreur est nulle si on utilise un solveur linéaire exact type LU.
  - Avec les méthodes itératives, elle est donnée par la tolérance du solveur.

Erreur de projection: espace non-linéaire

- L'erreur d'approximation est donnée par les théorèmes d'approximation universelle des réseaux de neurones.
- L'erreur d'optimisation est très difficile à analyser. On va donc ici se concentrer sur l'erreur d'intégration.
- Si on utilise une méthode de quadrature classique les théorèmes d'approximation nous donne naturellement:

$$| \mathcal{E}(\theta) - \mathcal{E}_N(\theta) | \leqslant C_{quad} N^{-\alpha}$$

Cas Monte-Carlo

La difficulté vient du fait que l'apprentissage dépend du jeux de données (ici les points d'intégration MC).

- Pour traiter le cas, il faut donc considérer l'erreur

$$\bar{\mathcal{E}}(\theta) = \mathbb{E}_S[\mathcal{E}(\theta(S))] = \int_{\Omega^N} \int_{\Omega} \| u_\theta - u \|_x^2$$

- En faisant une analyse semblable a précédemment, mais plus technique, on aboutit.

Méthodes d'approximation pour les EDP elliptiques

Méthodes d'approximation

Espaces linéaires

- Ritz-Galerkin:

$$\theta^* = \min_{v \in V_n} (a(v,v) - f(x)v)$$

- Least square Galerkin:

$$\theta^* = \min_{v \in V_n} \int_\Omega |\, L(u) - f \,|^2$$

Espace nonlinéaires

- Deep-Ritz:

$$\theta^* = \min_{v \in M_n} (a(v,v) - f(x)v)$$

- PINNs:

$$\theta^* = \min_{v \in M_n} \int_\Omega |\, L(u) - f \,|^2$$

- L'idée est la même. On restreint les fonctionnelles à minimiser à l'espace d'approximation.
- Entre les méthodes classiques et les méthodes neuronales ce qui diffère c'est l'espace d'approximation. Le choix de l'approximation des intégrales et de la résolution en découle.

Cas nonlinéaire: Résolution, géométrie et adaptation

Résolution

Pour résoudre les problèmes de minimisation des PINNs ou de Deep-Ritz, on utilise les mêmes méthodes que pour calculer l'opérateur de restriction :

- Méthodes de type gradient stochastique (Adam, ResProp, etc)
- Méthodes de Newton et Quasi-newton (L-BFGS, Leverberg-Marquardt).
- Méthodes de type gradient stochastique Préconditionné.

- Il est assez fréquent de combiner deux méthodes. Les méthodes de Newton/quasi Newton converge lentement mais sont moins robuste a une mauvaise initialisation et sont plus coûteuses.
- **Approche classique**: On commence avec une méthode de gradient et on finit avec un algorithme de type quasi-Newton.

Intégration, géométrie et adaptation

Les stratégies pour intégrer les fonction coûts, traiter les géométries complexe et faire de l'adaptation sont les mêmes que pour l'opérateur de restriction.

- Pour l'adaptation, on essaie de construire $g_\theta(\mathbf{x}) \approx \| r(\mathbf{x}) \|^2$ avec $r(\mathbf{x}) = L(u_\theta)(\mathbf{x}) - f(\mathbf{x})$

Conditions limites faibles et équilibrage de coûts I

- Comme pour les méthodes linéaires usuelles on peut imposer les conditions limites faiblement.
- Pour les méthodes EF on parle souvent de pénalisation.
- On nomme $\mathcal{J}_r(u)$ la fonctionnelle à minimiser (PINNs ou Deep-Ritz). On nomme le résidu des conditions limites $B(u) = 0$ (Dirichlet, Neumann ou autre).

Conditions limites faibles pour les méthodes neuronales

Le problème de minimisation devient

$$\min_{u_\theta \in W_n} \left(\mathcal{J}_r(u_\theta) + \lambda_{bc} \int_\Omega \| B(u_\theta) \|_2^2 \, d\mathbf{x} \right)$$

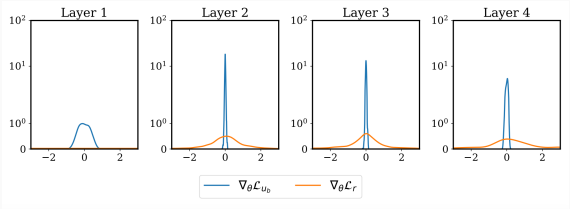
- Les méthodes linéaire prennent souvent $\lambda_{bc} = \frac{1}{\epsilon}$ avec ϵ petit. Ici l'optimisation étant non-linéaire on utilise des coefficients plus raisonnable.
- **Echantillonnage**: On doit échantillonner des points de bords. On peut faire ca avec des courbes paramétrique $h(\mathbf{t})$ avec $\mathbf{t}$ qui sera échantillonné uniformément.

Question

Comment choisir/adapter λ_{bc} ?

Conditions limites faibles et équilibrage de coûts II

- Problème d'entrainement avec les BC faibles. Exemple: Helmholtz equation.
- Référence: *Understanding and mitigating gradient pathologies in PINNs.*



- Sur cette exemple, le gradient de J_{bc} est très souvent localisé autour de zéro.
- Si l'entrainement voit pas les BC on peut capturer des solutions triviales. Exemple:

$$\begin{cases} -\partial_{xx} u = f, & \forall x \in \Omega \\ u = g & \forall x \in \partial\Omega \end{cases}$$

- Si la solution est de la forme $u(x) = \sin(2\pi k x)$ la source est de la forme $4\pi^2 k^2 u(x)$.
- Supposons que $u_\theta(x) = h_\theta(x) u(x)$ avec $\left\| \frac{\partial^k h_\theta(x)}{\partial x^k} \right\|_{L^\infty} < \epsilon$. On obtient:

$$\|\nabla_\theta \mathcal{J}_{bc}(\theta)\|_{L^\infty} \leqslant 2\epsilon \|\nabla_\theta h_\theta(x)\|_{L^\infty} \qquad \|\nabla_\theta \mathcal{J}(\theta)\|_{L^\infty} \leqslant O\left(16\pi^4 k^4\right) \cdot \epsilon \|\nabla_\theta h_\theta(x)\|_{L^\infty}$$

Conditions limites faibles et équilibrage de coûts III

- Comment équilibrer les fonctions de coûts ?
  - Descente de gradient

$$\theta_{n+1} = \theta_n - \eta \nabla_\theta \mathcal{J}(\theta_n) = \theta_n - \eta [\nabla_\theta \mathcal{J}_r(\theta_n) + \nabla_\theta \mathcal{J}_{bc}(\theta_n)]$$

  - Développement limité:

$$\mathcal{J}(\theta_{n+1}) = \mathcal{J}(\theta_n) + (\theta_{n+1} - \theta_n) \cdot \nabla_\theta \mathcal{J}(\theta_n) + \frac{1}{2}(\theta_{n+1} - \theta_n)^T \nabla^2_\theta \mathcal{J}(\xi)(\theta_{n+1} - \theta_n)$$

    avec $\xi = t\theta_n + (1-t)\theta_{n+1}$ et $t \in [0, 1]$
  - On combine les deux expressions:

$$\begin{aligned}
\mathcal{J}(\theta_{n+1}) - \mathcal{J}(\theta_n) &= -\eta \nabla_\theta \mathcal{J}(\theta_n) \cdot \nabla_\theta \mathcal{J}(\theta_n) + \frac{1}{2}\eta^2 \nabla_\theta \mathcal{J}(\theta_n)^T \nabla^2_\theta \mathcal{J}(\xi) \eta \nabla_\theta \mathcal{J}(\theta_n) && (1)\\
&= -\eta \|\nabla_\theta \mathcal{J}(\theta_n)\|_2^2 + \frac{1}{2}\eta^2 \nabla_\theta \mathcal{J}(\theta_n)^T \nabla^2_\theta \mathcal{J}(\xi) \nabla_\theta \mathcal{J}(\theta_n) && (2)\\
&= -\eta \|\nabla_\theta \mathcal{J}(\theta_n)\|_2^2 + \frac{1}{2}\eta^2 \nabla_\theta \mathcal{J}(\theta_n)^T \nabla^2_\theta \mathcal{J}(\xi) \nabla_\theta \mathcal{J}(\theta_n) && (3)
\end{aligned}$$

  - On appelle Q la matrice de orthogonale qui diagonalise le Hessienne. Avec quelques calculs on obtient

$$\begin{aligned}
\nabla_\theta \mathcal{J}(\theta_n)^T \nabla^2_\theta \mathcal{J}(\xi) \nabla_\theta \mathcal{J}(\theta_n) &= \|\nabla_\theta \mathcal{J}(\theta_n)\|_2^2 y^T \operatorname{diag}(\lambda_1, \lambda_2 \dots \lambda_M)\mathbf{y} && (4)\\
&= \|\nabla_\theta \mathcal{J}(\theta_n)\|_2^2 \sum_{i=1}^{M} \lambda_i y_i^2 && (5)
\end{aligned}$$

    avec $\mathbf{y} = Q \frac{\nabla_\theta \mathcal{J}(\theta_n)}{\|\nabla_\theta \mathcal{J}(\theta_n)\|}$.

Conditions limites faibles et équilibrage de coûts IV

- Suite:
  - En appliquant au deux fonctions de coûts on a:

$$\mathcal{J}(\theta_{n+1}) - \mathcal{J}(\theta_n) = \eta \|\nabla_\theta \mathcal{J}(\theta_n)\|_2^2 \left(-1 + \frac{1}{2}\eta \sum_{i=1} \lambda_i y_i^2 \right)$$

    avec λ_i les valeurs propres de $\nabla_\theta^2 \mathcal{J}(\xi) = \nabla_\theta^2 \mathcal{J}_r(\xi) + \nabla_\theta^2 \mathcal{J}_{bc}(\xi)$

- Si les valeurs propres sont grandes et η pas assez petit le gradient peux remonter.
- En pratique on peut constater dans des cas non convergent qu'il y a des valeurs propres très grandes venant de $\nabla_\theta^2 \mathcal{J}_r(\xi)$.
- On considère la fonction de coût

$$\mathcal{L}(\theta) := \mathcal{L}_r(\theta) + \sum_{i=1}^{M} \lambda_i \mathcal{L}_i(\theta)$$

- **Algorithme** (étape k):
  - on recalculer les nouveaux poids

$$\hat{\lambda}_i = \frac{\max_\theta \{|\nabla_\theta \mathcal{J}_r(\theta_n)|\}}{\frac{1}{N}\sum_{i=1}^{N} |\nabla_\theta \mathcal{J}_i(\theta_n)|}, i = 1, \ldots, M$$

  - On met a jour les poids

$$\lambda_i = (1-\alpha)\lambda_i + \alpha \hat{\lambda}_i, \quad i = 1, \ldots, M$$

  - On fait la descente de gradient:

$$\theta_{n+1} = \theta_n - \eta \nabla_\theta \mathcal{J}_r(\theta_n) - \eta \sum_{i=1}^{M} \lambda_i \nabla_\theta \mathcal{J}_i(\theta_n)$$

Conditions limites fortes I

- En général dans les méthodes linéaires classiques, des conditions comme celles de Dirichlet sont imposées en dur dans les fonctions de bases et donc dans l'espace.
- Comment faire de même ici ?

Fonctions de Rvachev

On se donne une fonction niveau ϕ. On introduit fonctions normalisés de Rvachev:

$$u(\mathbf{x}) = u_0^*(\mathbf{x}) + \sum_{k=1}^{m} \frac{u_k^*(\mathbf{x})}{k!}\phi^k(\mathbf{x}) + \phi^{m+1}(\mathbf{x})\Psi(\mathbf{x})$$

avec $u_k^*(\mathbf{x}) = \psi_k(\mathbf{x} - \phi\nabla\phi)$ ou ψ_k et Ψ sont des fonctions inconnues. Elles satisfont

$$u(\mathbf{x}) = \psi_0(\mathbf{x}), \quad \forall \mathbf{x} \in \partial\Omega$$

et

$$\frac{\partial^k u(\mathbf{x})}{\partial \nu^k} = \psi_k(\mathbf{x}), \quad \forall \mathbf{x} \in \partial\Omega$$

avec $\nu = \nabla\phi$ la normale rentrante dans le domaine.

Conditions limites fortes II

Condition de Dirichlet

Soit $u_\theta(\mathbf{x})$ un modèle paramétrique nonlinéaire. Une fonction normalisée de Rvachev avec $\psi_0(\mathbf{x}) = g(\mathbf{x})$ et $\psi_1(\mathbf{x}) = u_\theta(\mathbf{x})$ satisfaisant une condition de Dirichlet du type $u(\mathbf{x}) = g(\mathbf{x})$ sur le bord du domaine Ω défini par ϕ. En pratique on peut l'approcher à l'ordre un par:

$$u(\mathbf{x}) = g(\mathbf{x}) + \phi(\mathbf{x})u_\theta(\mathbf{x})$$

- On prend fonction normalisée de Rvachev avec $m = 0$ on a donc

$$u(\mathbf{x}) = \psi_0(\mathbf{x} - \phi\nabla\phi) + \phi(\mathbf{x})\psi_1(\mathbf{x})$$
$$u(\mathbf{x}) = \psi_0(\mathbf{x}) - (\phi\nabla\phi, \nabla\psi_0(\mathbf{x})) + \phi(\mathbf{x})\psi_1(\mathbf{x}) + O((\phi\nabla\phi)^2)$$
$$u(\mathbf{x}) = \psi_0(\mathbf{x}) - (\phi\nabla\phi, \nabla\psi_0(\mathbf{x})) + \phi(x)\psi_1(\mathbf{x}) + O((\phi\nabla\phi)^2)$$

- on utilise que $\nabla\phi = -\mathbf{n}$

$$u(\mathbf{x}) = \psi_0(\mathbf{x}) + \phi\frac{\partial\psi_0(\mathbf{x})}{\partial\mathbf{n}} + \phi(\mathbf{x})\psi_1(\mathbf{x}) + O((\phi\nabla\phi)^2).$$

Puisque est nul au bord on peut simplifier et obtenir:

$$u(\mathbf{x}) = \psi_0(\mathbf{x}) + \phi(\mathbf{x})\psi_1(\mathbf{x}) + O((\phi\nabla\phi)^2).$$

et utiliser cette relation en négligeant les termes du second ordre.

Conditions limites fortes III

Condition de Neumann

Soit $u_\theta(\mathbf{x})$ un modèle paramétrique nonlinéaire. On se une fonction normalisée de Rvachev à l'ordre 2 satisfaisant les conditions de Neumann $\frac{\partial u_\theta}{\partial \mathbf{n}} = h(\mathbf{x})$. En pratique on peut l'approcher par:

$$u(\mathbf{x}) = \left(1 + \phi(\mathbf{x})\frac{\partial}{\partial \mathbf{n}}\right) u_{\theta,1}(\mathbf{x}) - \phi(\mathbf{x})h(\mathbf{x}) + \phi^2(\mathbf{x})u_{\theta,2}(\mathbf{x})$$

avec $u_{\theta,1}(\mathbf{x})$ et $u_{\theta,2}(\mathbf{x})$ des modèles paramètriques

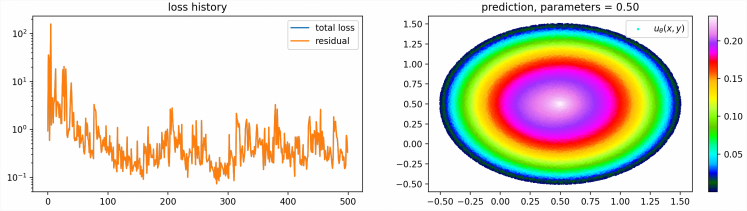
- On commence par développer notre fonction de Rvachev:
$$u(\mathbf{x}) = \psi_0(\mathbf{x} - \phi\nabla\phi) + \phi(\mathbf{x})\psi_1(\mathbf{x} - \phi\nabla\phi) + \phi^2(\mathbf{x})\psi(\mathbf{x})$$
- On linéarise le premier terme
$$u(\mathbf{x}) = \psi_0(\mathbf{x}) - \phi(\nabla\phi, \nabla\psi_0) + o(\nabla\phi^2) + \phi(\mathbf{x})\psi_1(\mathbf{x} - \phi\nabla\phi) + \phi^2(\mathbf{x})\psi(\mathbf{x})$$
- On utilise que $\mathbf{n} = -\nabla\phi$ donc
$$u(\mathbf{x}) = \psi_0(\mathbf{x}) + \phi(\nabla\psi_0, \mathbf{n}) + o(\nabla\phi^2) + \phi(\mathbf{x})\psi_1(\mathbf{x} - \phi\nabla\phi) + \phi^2(\mathbf{x})\psi(\mathbf{x})$$
- ce qui donne
$$u(\mathbf{x}) = \psi_0(\mathbf{x}) + \phi\frac{\partial\psi_0}{\partial\mathbf{n}} + o(\nabla\phi^2) + \phi(\mathbf{x})\psi_1(\mathbf{x} - \phi\nabla\phi) + \phi^2(\mathbf{x})\psi(\mathbf{x})$$
Par définition des fonction de Rvachev on a $\psi_1 = \frac{\partial}{\partial\mathbf{v}} = -h$ donc
$$u(\mathbf{x}) = \psi_0(\mathbf{x}) + \phi\frac{\partial\psi_0}{\partial\mathbf{n}} + o(\nabla\phi^2) - \phi(\mathbf{x})h(\mathbf{x}) + \phi^2(\mathbf{x})\psi(\mathbf{x})$$
- Il suffit ensuite de paramétriser ψ_0 et ψ.

Conditions limites fortes IV

- Les conditions de Robin peuvent se traiter de façon similaire.
- **Référence**: *Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks.* Il est détaillé le cas de multiples conditions limites sur le même domaine.
- Est ce que toutes les fonctions niveaux fonctionnent ?
- On résout

$$-\Delta u = 1, \quad \forall \mathbf{x} \in B(0.5, 1), \qquad u = 0, \quad \forall \mathbf{x} \in \partial B(0.5, 1)$$

- On utilise la fonction distance signée $\phi(\mathbf{x}) = \sqrt{x^2 + y^2} - 1$ et on impose en dur les conditions de Dirichlet.

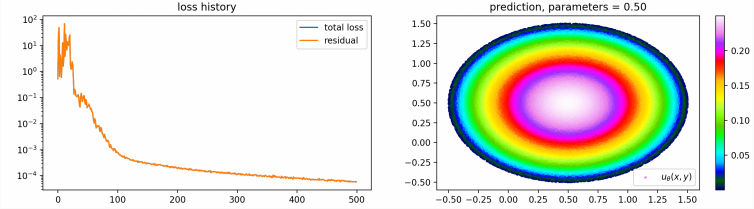


Conditions limites fortes IV

- Les conditions de Robin peuvent se traiter de façon similaire.
- **Référence**: *Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks.* Il est détaillé le cas de multiples conditions limites sur le même domaine.
- Est ce que toutes les fonctions niveaux fonctionnent ?
- On résout

$$-\Delta u = 1, \quad \forall \mathbf{x} \in B(0.5, 1), \qquad u = 0, \quad \forall \mathbf{x} \in \partial B(0.5, 1)$$

- On utilise
  la fonction distance signée $\phi(\mathbf{x}) = x^2 + y^2 - 1$ et on impose en dur les conditions de Dirichlet.



- La dérivée seconde de la sdf est est singulière. Donc le Laplacien de f $u(\mathbf{x}) = g(\mathbf{x}) + \phi(\mathbf{x}) u_\theta(\mathbf{x})$ est singulier et l'apprentissage ne converge pas ca ils se concentre sur les points singulier.

Conditions limites fortes V

- Comment apprendre une fonction niveau régulière ?

Apprentissage de fonction distance signée régularisée

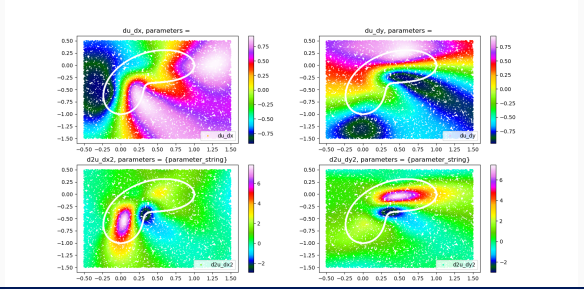
On se donne des paires $(\mathbf{x}_b, \mathbf{n}_b)$ aux bords. On résout:

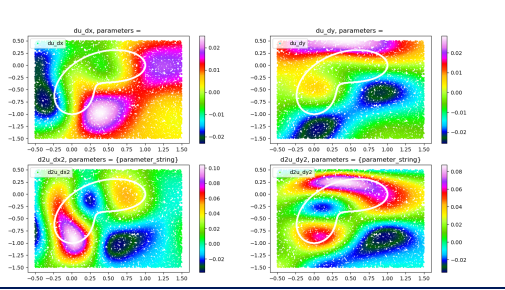
$$\min_{\phi \in C^0} (\mathcal{J}_{eq}(\phi) + \mathcal{J}_{di}(\phi) + \mathcal{J}_{Neu}(\phi) + \mathcal{J}_{reg}(\phi))$$

avec

$$\mathcal{J}_{reg}(\phi) = \int_{[a,b]^d} |\Delta\phi|^2 d\mathbf{x}.$$

- Apprentissage sans/avec régularisation





Résumé sur les conditions aux limites

Classique

- BC forte: V_n construit tel que $f_\theta(\mathbf{x}) = 0, \quad \mathbf{x} \in \partial\Omega, \forall f_\theta \in V_n$
- BC pénalisée:

$$\theta^* = \min_{v \in V_n} \left(J(v) + \frac{1}{\epsilon} \int_\Omega | u - g |^2 \right)$$

Réseaux

- BC forte: $W_n = \{ nn_\theta(x) * h(x) \}$ avec $h(x) = 0$ au bords.
- Il faut une fonction niveau régulière.
- BC pénalisée:

$$\theta^* = \min_{v \in W_n} \left(J(v) + \frac{1}{\epsilon} \int_\Omega | u - g |^2 \right)$$

- $\frac{1}{\epsilon}$ est remplacé par λ. Il peut être adapté automatiquement pendant la descente de gradient.

Problèmes paramétriques et malediction de la dimension

- En optimisation, propagation d'incertitude etc on souhaite résoudre des problèmes du type

$$L_\alpha(u(\mathbf{x})) - f(\mathbf{x}, \beta)$$

avec $\mu = (\alpha, \beta)$ des paramètres qui vivent dans un espace V_μ.

- Les méthodes usuelles sont trop couteuse en grande dimension donc on résout pas ce problème dans l'espace $\Omega \times V_\mu$.
- En général on fait des simulations pour différent μ et on construit un modèle réduit.

Méthodes neuronales paramétriques

Les espaces de réseaux de neurones étant plus efficace en grande dimension ont peut essayer de résoudre dans l'espace $\Omega \times V_\mu$

- Dans ce cas l'opérateur de restriction devient:

$$\theta^* = \min_\theta \int_{V_\mu} \int_\Omega \mid u(\mathbf{x}, \mu) - nn_\theta(\mathbf{x}, \mu) \mid^2 dx,$$

- La méthodes PINNs devient:

$$\theta^* = \min_\theta \int_{V_\mu} \int_\Omega \mid L_\alpha(u(\mathbf{x}, \mu)) - f(\mathbf{x}, \beta) \mid^2 dx,$$

Erreur numérique et stabilité

- On considère que $u \in H_x(\Omega)$ avec H_x solution de $L(u) - f = 0$ avec $f(\mathbf{x}) \in H_y$

$$\hat{\theta} = \min_\theta \mathcal{R}(\theta) = \min_\theta \int_\Omega \parallel L(u_\theta) - f \parallel_y^2$$

- En pratique on calcul numériquement $\theta^*(S)$ par un algorithme numérique qui minimise $\mathcal{R}_N(\theta) = \sum_{i=1}^N w_i \parallel L(u_\theta(\mathbf{x}_i)) - f(\mathbf{x}_i) \parallel_x^2$ avec $S = \{\mathbf{x}_1, ...\mathbf{x}_N\}$
- On chercher a estimer l'erreur du modèle calculé donc $\mathcal{E}(\theta^*(S))$.

Stabilité

On dit que l'EDP est stable si : $\parallel u - v \parallel_X \leqslant C_{\mathrm{PDE}} \| \mathcal{L}(u) - \mathcal{L}(v) \|_Y$

Erreur numérique

Soit u la solution de l'EDP stable $Lu(\mathbf{x}) = f(\mathbf{x})$ alors on

$$\mathcal{E}(\theta^*(S)) \leqslant C_{\mathrm{PDE}} C(\mathcal{R}(\widehat{\theta}) + 2 \sup_{\theta \in \Theta} \mid \mathcal{R}(\theta) - \mathcal{R}_N \mid + \mathcal{R}_N(\theta^*) - \min_\theta \mathcal{R}_N(\theta))$$

- Par continuité on a

$$\mathcal{E}(\theta^*(S)) = \int_\Omega \parallel u - u_{\theta^*} \parallel_x^2 \leqslant \int_\Omega L(u_{\theta^*}) - L(u) \parallel_y^2 = \int_\Omega L(u_{\theta^*}) - f \parallel_y^2 - \mathcal{R}(\theta^*(S))$$

- On applique ensuite le même raisonnement a $\mathcal{R}(\theta^*(S))$ que pour $\mathcal{E}(\theta^*(S))$ dans la section projection.

Méthodes d'approximation et espace linéaire de réseaux

Méthodes des réseaux aléatoires I

Modèles linéaires basés sur des réseaux aléatoires

On peut aussi utilisées des réseaux de neurones comme des fonctions de bases globales. On retrouve l'espace d'approximation

$$V_n = \left\{ \sum_{i=1}^{N} \theta_i \phi_i(\mathbf{x}), \quad \theta \in V \subset \mathbb{R}^n \right\}$$

avec $\phi_i(\mathbf{x}) = nn_{\theta_i}$ ou les poids sont choisis aléatoirement. On parle d'extreme ML ou de méthode "random features".

- Cela peut se combiner aux méthodes de Galerkin ou Galerkin-moindre carrés.
- Puisque nos bases sont des réseaux on garde la quadrature de Monte-Carlo.
- Les stratégies utilisées pour les BC fortes et faibles pour les espaces nonlinéaires de réseaux marche aussi pour cette approche.

Résolution

Si le nombre de base et de points sont égaux on se retrouve avec une matrice a inverser sinon avec un problème de moindre carré à résoudre.

Détails pour les méthodes des "réseaux aléatoire"

- Premier choix de base:

$$\phi_i(\mathbf{x}) = \sigma(A_i\mathbf{x} + \mathbf{b}_i)$$

avec A_i, $\mathbf{b}_i$ choisis aléatoirement.

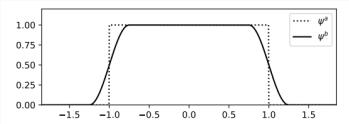
- Deuxième choix de base:

$$\phi_i(\mathbf{x}) = \sigma\left(A_i \frac{1}{\mathbf{r}}(\mathbf{x} - \mathbf{x}_i) + \mathbf{b}_i\right)$$

avec A_i, $\mathbf{b}_i$ choisis aléatoirement, $\mathbf{x}_i$ des points du domaine, et $\mathbf{r} \in \mathbb{R}^d$.

- Partition de l'unité:
  - On se donne une fonction support
  - en 1D

$$\psi_n^d(x) = \begin{cases} \frac{1+\sin(2\pi\tilde{x})}{2} & -\frac{5}{4} \leqslant \tilde{x} < -\frac{3}{4}, \\ 1 & -\frac{3}{4} \leqslant \tilde{x} < \frac{3}{4}, \\ \frac{1-\sin(2\pi\tilde{x})}{2} & \frac{3}{4} \leqslant \tilde{x} < \frac{5}{4}, \\ 0 & \text{sinon} \end{cases}$$



  - en dimension d:

$$\psi_n(\mathbf{x}) = \prod_{k=1}^{d} \psi_n^d(x_k)$$

  - **Base multi-échelles**:

$$u_\theta(\mathbf{x}) = \sum_{i=1}^{M} \theta_i \phi_i(\mathbf{x})) + \sum_{n=1}^{M_p} \psi_n(\mathbf{x}) \sum_{j=1}^{J_n} \theta_{nj} \phi_{nj}(\mathbf{x})$$

Temporary page!

LaTeX was unable to guess the total number of pages correctly. As there was some unprocessed data that should have been added to the final page this extra page has been added to receive it.

If you rerun the document (without altering it) this surplus page will go away, because LaTeX now knows how many pages to expect for this document.