Méthodes numériques pour les EDP temporelles

Emmanuel Franck[*],

19 Novembre , 2024
Master CMSI, M2, Strasbourg

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

Outline

EDP temporelles linéaires et méthodes numériques linéaires

- EDP temporelles

Méthodes neuronales et EDP temporelles

- Approche Espace-temps
- Approches temporelles neuronales

EDP temporelles linéaires et méthodes numériques linéaires

EDP temporelles linéaires et méthodes numériques linéaires

EDP temporelles

Méthodes neuronales et EDP temporelles

Approche Espace-temps

Approches temporelles neuronales

EDP temporelles

- Ici on va considérer des EDP linéaires et scalaire.
- La théorie se généralise facilement aux systèmes. Le cas nonlinéaire nécessite des traitements spécifiques.

Exemple EDP

On considère une EDP de la forme:

$$\partial_t u(\mathbf{x}) + \nabla \cdot (\mathbf{a}(\mathbf{x})u(\mathbf{x})) - \nabla \cdot (\mathbf{D}(\mathbf{x})\nabla u(\mathbf{x})) + c(\mathbf{x})u(\mathbf{x}) = f(\mathbf{x})$$

qu'on peut réécrire sous la forme $\partial_t u(t, \mathbf{x}) = G(u(t, \mathbf{x}))$.

- Approche espace-temps: le temps est une dimension comme une autre.

$$V_n = \left\{ f_\theta(t, \mathbf{x}) = \sum_{i=1}^{n} \theta_i \phi_i(t, \mathbf{x}), \quad \theta \in \Theta \subset \mathbb{R}^n, \quad f_\theta(t, \mathbf{x}) = 0, \quad \mathbf{x} \in \partial\Omega \right\}$$

- Pas de formulation énergétique donc on utilise l'approche Galerkin moindre carrés.

Méthode DpO I

Approche classique

On va considérer que a chaque temps la solution est dans

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

ce qui est équivalent:

$$f(t, \mathbf{x}) = f_{\theta(t)}(\mathbf{x}) = \sum_{i=1}^{n} \theta_i(t) \phi_i(\mathbf{x}),$$

et on fait évoluer les paramètres.

- On va commencer par discrétiser l'équation (ici Euler explicite)

$$u(t^{k+1}, \mathbf{x}) = u(t^k, \mathbf{x}) + \Delta t G(u(t^k, \mathbf{x})) + O(\Delta t^2)$$

- On remplace la solution par sa projection dans V_n. On aimerait donc avoir

$$\mathcal{I}(\theta_{k+1}) = \Pi_{V_n}(u)(t^k, \mathbf{x}) + \Delta t G(\Pi_{V_n}(u(t^k, \mathbf{x})))$$

Méthode DpO II

- On suppose qu'on connait $\Pi_{V_n}(u(t^k, \mathbf{x}))$. Pour avoir l'expression précédente il faut donc que

$$\theta_{k+1} = \mathcal{R}(\Pi_{V_n}(u)(t^k, \mathbf{x}) + \Delta t G(\Pi_{V_n}(u(t^k, \mathbf{x}))))$$

  donc résoudre

$$A\theta_{k+1} = \mathbf{b}$$

  avec $\mathbf{b}$ associé à $f(\mathbf{x}) = \Pi_{V_n}(u)(t^k, \mathbf{x}) + \Delta t G(\Pi_{V_n}(u(t^k, \mathbf{x})))$ et A donnée par votre méthode de Collocation ou de Galerkin.

- Justification:
  - On part de
$$\theta_{k+1} = \mathcal{R}(\Pi_{V_n}(u)(t^k, \mathbf{x}) + \Delta t G(\Pi_{V_n}(u(t^k, \mathbf{x}))))$$
  - On applique l'opérateur de reconstruction
$$\mathcal{I}(\theta_{k+1}) = \mathcal{I}(\mathcal{R}(\Pi_{V_n}(u)(t^k, \mathbf{x}) + \Delta t G(\Pi_{V_n}(u(t^k, \mathbf{x})))))$$
  - ce qui est égale à:
$$\mathcal{I}(\theta_{k+1}) = \Pi_{V_n}(\Pi_{V_n}(u)(t^k, \mathbf{x}) + \Delta t G(\Pi_{V_n}(u(t^k, \mathbf{x}))))$$
  - Or si $f \in V_n$ on a $\Pi_{V_n}(f) = f$ et donc
$$\mathcal{I}(\theta_{k+1}) = \Pi_{V_n}(u)(t^k, \mathbf{x}) + \Delta t G(\Pi_{V_n}(u(t^k, \mathbf{x})))$$

Méthode DpO II

- La méthode de Galerkin ca revient à:

$$\theta_{k+1} = \min_\theta \parallel \langle \theta, \Phi \rangle - \Pi_{V_n}(u)(t^k, \mathbf{x}) - \Delta t G(\Pi_{V_n}(u(t^k, \mathbf{x}))) \parallel_{L^2}$$

$$\theta_{k+1} = \min_\theta \parallel \langle \theta, \Phi \rangle - \langle \theta^k, \Phi \rangle - \Delta t G(\langle \theta^k, \Phi \rangle) \parallel_{L^2}$$

Une fois résolu on obtient:

$$M\theta_{k+1} = M\theta_k - \Delta t R(\theta_k)$$

avec $M = \int_\Omega \Phi(\mathbf{x}) \otimes \Phi(\mathbf{x}) d\mathbf{x}$, $\quad R(\theta_k) = \int_\Omega \Phi(\mathbf{x}) G(\langle \theta^k, \Phi \rangle) d\mathbf{x}$

- La méthode de collocation ca revient à:

$$\langle \theta^{k+1}, \Phi(\mathbf{x}_i) \rangle - \Pi_{V_n}(u)(t^k, \mathbf{x}_i) - \Delta t G(\Pi_{V_n}(u(t^k, \mathbf{x}_i))) = 0$$

$$\langle \theta^{k+1}, \Phi(\mathbf{x}_i) \rangle - \langle \theta^k, \Phi(\mathbf{x}_i) \rangle - \Delta t G(\langle \theta^k, \Phi(\mathbf{x}_i) \rangle) = 0$$

- Une stratégie similaire peut être utilisée en implicite.

Galerkin moindre Carré et OpD

- On peut appliquer la projection au niveau continu (avant discrétisation en temps)

$$\partial_t \Pi_{V_n}(u(t,x)) + G(\Pi_{V_n}(u(t,x)) = 0$$

- On remarque que $\Pi_{V_n}(u(t,x)) = \langle \theta(t), \Phi(\mathbf{x}) \rangle$ donc que

$$\partial_t \Pi_{V_n}(u(t,x)) = \langle \frac{d}{dt}\theta(t), \Phi(\mathbf{x}) \rangle, \quad \rightarrow \langle \frac{d}{dt}\theta(t), \Phi(\mathbf{x}) \rangle = -G(\Pi_{V_n}(u(t,x))$$

- On souhaite donc avoir $\frac{d}{dt}\theta(t) = -\mathcal{R}(G(\Pi_{V_n}(u(t,x)))$.
  - Méthode de Galerkin ca revient à chercher

$$\frac{d}{dt}\theta(t) = \min_\eta \parallel \langle \eta, \Phi(\mathbf{x}) \rangle - G(\Pi_{V_n}(u(t,\mathbf{x})) \parallel_{L^2}$$

$$M\frac{d}{dt}\theta(t) = R(\theta(t)), \quad M = \int_\Omega \Phi(\mathbf{x}) \otimes \Phi(\mathbf{x}) d\mathbf{x}, \quad R(\theta(t)) = \int_\Omega \Phi(\mathbf{x}) G(\Pi_{V_n}(u(t,\mathbf{x})) d\mathbf{x}$$

  - Par exemple pour la méthode de collocation ca revient à chercher

$$\langle \frac{d}{dt}\theta(t), \Phi(\mathbf{x}_i) \rangle + G(\Pi_{V_n}(u(t,\mathbf{x}_i)) = 0, \quad \forall \mathbf{x}_i \in X$$

Méthodes neuronales et EDP temporelles

PINNs

- Les approches espace temps sont moins utilisées dans le cadre des méthodes classiques sauf avec des bases discontinues (DG).
- A l'inverse les méthodes neuronales ont commencées avec des approches espace-temps.
- Soit une EDP de la forme
$$(\partial_t^\alpha - L)(u) = f$$
- On peut proposer une forme moindre carré:
$$\min_{u \in H} \left(\int_0^T \int_\Omega \| (\partial_t^\alpha - L)(u) - f \|_2^2 \right)$$

PINNs

Soit un espace nonlinéaire M_n, la méthode PINNs est une méthode de Galerkin moindre carré espace-temps ce qui revient a résoudre:
$$\min_{u \in H} \left(\int_0^T \int_\Omega \| (\partial_t^\alpha - L)(u) - f \|_2^2 \right)$$

- le temps étant traité comme une dimension comme les autres la condition initiale devient une condition aux bords (gauche seulement)
- On peut donc, comme précédemment l'imposer fortement ou faiblement.

NTK et causalité

1er défaut

Lorsque la condition initiale est imposée faiblement si la fonction de coût residue est trop importante l'apprentissage peut apprendre une solution triviale.

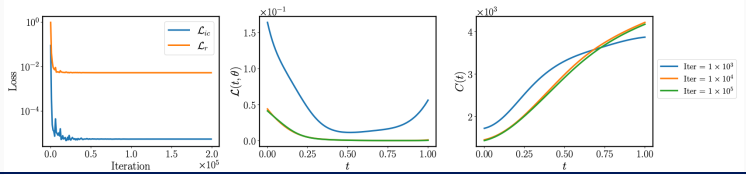
- On reprend le problème de minimisation précédent et on discrétise les intégrales.
- On doit minimiser:

$$\mathcal{J}\theta) = \sum_{i=1}^{N_t} J(\theta, t_i), \qquad \mathcal{J}(\theta, t) = \sum_{j=1}^{N_x} \| \ (\partial_t^\alpha - L)(u(t, \mathbf{x}_j)) - f(t, \mathbf{x}_j) \ \|_2^2$$

- On peut calculer le noyau NTK suivant: $K_\theta(t) = \langle \frac{\partial \mathcal{J}(\theta,t)}{\partial \theta}, \frac{\partial \mathcal{J}(\theta,t)}{\partial \theta} \rangle$ et définir le taux de convergence de la descente de gradient dépendant du temps:

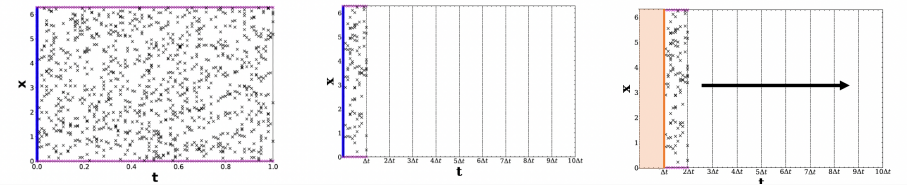
$$C(t) = \frac{Trace(K_\theta(t))}{N_t}$$

- Exemple sur une EDP classique:



Causalité

- Deux stratégies pour remettre de la causalité.
- **Stratégie 1**:
  - On commence par résoudre le PINNs sur l'intervalle $[0, \Delta t]$ pendant m itératiions
  - Puis on passe à l'intervalle $[0, 2\Delta t]$ pendant m iterations et on itère.



- **Stratégie 2**:
  - On minimise:

$$\mathcal{J}_r(\theta) = \frac{1}{N_t} \sum_{i=1}^{N_t} \underbrace{\exp\Big(-\epsilon \sum_{k=1}^{i-1} \mathcal{J}_r(t_k, \theta)\Big)}_{w_i} \mathcal{J}_r(t_i, \theta).$$

  - Les poids w_i sont proportionnels a la fonction de cout cumulative jusqu'au temps t_i.
  - Par conséquent, $\mathcal{J}_r(t_i, \theta)$ ne sera pas minimisé à moins que tous les résidus précédents ne diminuent jusqu'à une petite valeur telle que w_i soit suffisamment grand.

PINNs discret

- On se donne un espace nonlinéaire (ici de réseau de neurones)

$$M_n = \{nn_\theta(\mathbf{x}), \quad \theta \in \Theta\}$$

- On considère une EDP $\partial_t u = G(u)$ ou G dépendra d'opérateurs différentiels.
- On discrétise en temps:

$$u(t^{k+1}, \mathbf{x}) = u(t^k, \mathbf{x}) + \Delta t G(u(t^k, \mathbf{x})) + O(\Delta t^2)$$

- Comme précédemment on on va supposer que les paramètres de notre modèle évolue en temps donc que $u(t, \mathbf{x}) = nn_{\theta(t)}(\mathbf{x})$.
- Comme précédemment, on aimerait donc avoir

$$\mathcal{I}(\theta_{k+1}) = \Pi_{V_n}(u)(t^k, \mathbf{x}) + \Delta t G(\Pi_{V_n}(u(t^k, \mathbf{x})))$$

- et pour cela on va résoudre

$$\theta_{k+1} = \mathcal{R}(\Pi_{V_n}(u)(t^k, \mathbf{x}) + \Delta t G(\Pi_{V_n}(u(t^k, \mathbf{x}))))$$

ce qui donne le PINNs discret associé au schéma d'Euler. Ce qui donne:

$$\theta_{k+1} = \min_\theta \int_\Omega \| nn_\theta(\mathbf{x}) - nn_{\theta_n}(\mathbf{x}) - \Delta t G(nn_{\theta_n}(\mathbf{x})) \|_2^2$$

PINNs discret II

PINNs discret associé au schéma d'Euler explicite

- A l'instant initiale on résout:

$$\theta_0 = \min_\theta \int_\Omega \| nn_\theta(\mathbf{x}) - u_0(\mathbf{x})) \|_2^2$$

- A chaque étape en temps on résout:

$$\theta_{k+1} = \min_\theta \mathcal{J}_r(\theta) = \min_\theta \int_\Omega \| nn_\theta(\mathbf{x}) - nn_{\theta_n}(\mathbf{x}) - \Delta t G(nn_{\theta_n}(\mathbf{x})) \|_2^2$$

- A chaque étape on initialise l'étape de gradient avec θ_n la valeur de θ au temps précédent. On fait donc de petit entrainement a chaque étape.
- Plus Δt est petit plus les entrainements sont rapides mais nombreux.
- Vous pouvez appliquer des schémas implicites ou RK (un entrainement par sous étape).

Théorie

Supposant l'équation dissipative et les solutions régulières, l'erreur entre la solution $u(t_n, x)$ et le modèle $u_{\theta^n}(x)$ noté $e^n(x) = u(t_n, x) - u_{\theta^n}(x)$ peut être bornée par

$$\|e^n\| \leqslant C\sqrt{1+t_n}\left(\tau^2 + \max_{1\leqslant i\leqslant n} \sqrt{\mathcal{J}^i} + N_r^{\frac{1}{4}}\right), \quad n = 1, \ldots, N_t$$

ou C dépend de $u(t_n, x)$ and $u_{\theta^n}(x)$ et N_r the nombre de points de quadrature.

Méthode Neural Galerkin I

- Si on reprend:

$$\theta_{k+1} = \min_{\theta} \int_{\Omega} \| nn_{\theta}(\mathbf{x}) - nn_{\theta_n}(\mathbf{x}) - \Delta t G(nn_{\theta_n}(\mathbf{x})) \|_2^2$$

- Normalement $nn_{\theta}(\mathbf{x})$ a vocation à approcher la solution au temps t_{n+1}. On propose donc de linéariser autour de θ_n qui ne devrait pas être éloigné.
- On obtient $nn_{\theta}(\mathbf{x}) = (\nabla_{\theta_n} u_{\theta_n})(\theta - \theta_n) + nn_{\theta_n}(\mathbf{x}) + O((\theta - \theta_n)^2)$.
- En injectant cela dans l'optimisation on obtient:

$$\theta_{k+1} = \min_{\theta} \int_{\Omega} \| (\nabla_{\theta_n} u_{\theta_n})(\theta - \theta_n) - \Delta t G(nn_{\theta_n}(\mathbf{x})) \|_2^2$$

Méthode neural Galerkin associé au schéma d'Euler explicite

- A l'instant initiale on résout:

$$\theta_0 = \min_{\theta} \int_{\Omega} \| nn_{\theta}(\mathbf{x}) - u_0(\mathbf{x}) \|_2^2$$

- A chaque étape en temps on résout:

$$\theta_{k+1} = \min_{\theta} \int_{\Omega} \| (\nabla_{\theta_n} nn_{\theta_n})(\theta - \theta_n) - \Delta t G(nn_{\theta_n}(\mathbf{x})) \|_2^2$$

Méthode Neural Galerkin II

- Il s'agit d'un problème au moindre carrés. On peut le résoudre numériquement par une méthode de résolution.
- On peut le résoudre analytiquement (équivalent de l'équation normale pour le produit scalaire L^2).
- On obtient:

$$M(\theta_n)\theta_{n+1} = M(\theta_n)\theta_n - \Delta t R(\theta_n)$$

avec $M(\theta_n) = \int_\Omega (\nabla_{\theta_n} nn_{\theta_n}) \otimes (\nabla_{\theta_n} nn_{\theta_n}) d\mathbf{x}$, $\quad R(\theta_n) = \int_\Omega (\nabla_{\theta_n} nn_{\theta_n}) G(nn_{\theta_n}(\mathbf{x})) d\mathbf{x}$.

- il peut avoir des problèmes d'inversion donc résoudra plutôt:

$$(M(\theta_n) + \epsilon I_d)\theta_{n+1} = M(\theta_n)\theta_n - \Delta t R(\theta_n)$$

Méthode neural Galerkin

On a appliqué une approche DpO. Comme dans le cas linéaire on peut utiliser l'approche OpD.

Méthode Neural Galerkin III

- On peut appliquer la projection au niveau continu (avant discrétisation en temps)

$$\partial_t \Pi_{V_n}(u(t,x)) + G(\Pi_{V_n}(u(t,x)) = 0$$

- On remarque que $\Pi_{V_n}(u(t,x)) = nn_{\theta(t)}(\mathbf{x})$ donc que

$$\partial_t \Pi_{V_n}(u(t,x)) = \langle \frac{d}{dt}\theta(t), (\nabla_\theta nn_{\theta(t)})\rangle, \quad \rightarrow \langle \frac{d}{dt}\theta(t), (\nabla_\theta nn_{\theta(t)})\rangle = -G(\Pi_{V_n}(u(t,x))$$

- On souhaite donc avoir $\frac{d}{dt}\theta(t) = -\mathcal{R}(G(\Pi_{V_n}(u(t,x))))$

Méthode neural Galerkin

- Méthode de neural Galerkin ca revient à chercher

$$\frac{d}{dt}\theta(t) = \min_\eta \parallel \langle \eta, (\nabla_{\theta(t)} nn_{\theta(t)})\rangle - G(\Pi_{V_n}(u(t,\mathbf{x})) \parallel_{L^2}$$

$$M(\theta(t))\frac{d}{dt}\theta(t) = R(\theta(t))$$

avec

$$M(\theta(t)) = \int_\Omega (\nabla_\theta nn_{\theta(t)}) \otimes (\nabla_\theta nn_{\theta(t)}) d\mathbf{x}, \quad R(\theta(t)) = \int_\Omega \nabla_\theta nn_{\theta(t)} G(\Pi_{V_n}(u(t,\mathbf{x}))d\mathbf{x}$$

Intégration et conditions limites

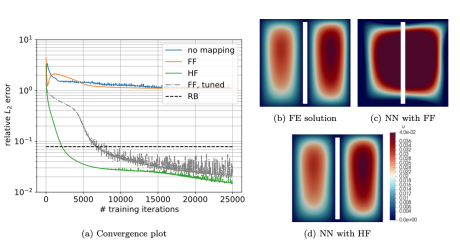
Intégration

Comme attendu on procède comme précédemment. On intègre avec une méthode de Monte Carlo. Pour l'adaptation on utilise une méthode spécifique au problème temporelle qu'on détaillera pas ici.

Condition limite

Pour les PINNs discret on peut les imposer en dur ou en faible. Pour la méthode Neural Galerkin on les imposera en dur afin de garder un problème au moindre carré.

- Approche pour les imposer:
  - Comme pour les Pinns: $u_{\theta(t)}(\mathbf{x}) = g(\mathbf{x}) + \phi(\mathbf{x}) nn_{\theta(t)}(\mathbf{x})$ avec ϕ une fonction niveau. Ca peut parfois être rigide.
  - On peut aussi prendre $u_{\theta(t)}(\mathbf{x}) = nn_{\theta(t)}(\phi(\mathbf{x}))$ avec un réseau qui est nul en zéro.
  - On peut aussi prendre $u_{\theta(t)}(\mathbf{x}) = nn_{\theta(t)}(\phi_1(\mathbf{x}), \ldots, \phi_n(\mathbf{x}))$ avec un réseau qui est nul en zéro et ϕ_i un vecteur propre du Laplacien sur la géométrie calculés par élément finis au préalable.



Référence: *Positional Embeddings for Solving PDEs with Evolutional Deep Neural Networks*, M. Kast, J. Hesthaven.

Schémas en temps. I

- Schéma explicite pour $M(\theta(t))\frac{d\theta(t)}{dt} = \mathbf{F}(\theta(t))$.
- Exemple: RK4

$$\Delta\theta_k = \frac{1}{6}\left(\Delta\theta_k^{(1)} + 2\Delta\theta_k^{(2)} + 2\Delta\theta_k^{(3)} + \Delta\theta_k^{(4)}\right),$$

$$\mathbf{M}_k(\theta_k)\Delta\theta_k^{(1)} = \mathbf{F}_k(\theta_k),$$

$$\mathbf{M}_k\left(\theta_k + \frac{\delta t_k}{2}\Delta\theta_k^{(1)}\right)\Delta\theta_k^{(2)} = \mathbf{F}_k\left(\theta_k + \frac{\delta t_k}{2}\Delta\theta_k^{(1)}\right),$$

$$\mathbf{M}_k\left(\theta_k + \frac{\delta t_k}{2}\Delta\theta_k^{(2)}\right)\Delta\theta_k^{(3)} = \mathbf{F}_k\left(\theta_k + \frac{\delta t_k}{2}\Delta\theta_k^{(2)}\right),$$

$$\mathbf{M}_k\left(\theta_k + \delta t_k\Delta\theta_k^{(3)}\right)\Delta\theta_k^{(4)} = \mathbf{F}_k\left(\theta_k + \delta t_k\Delta\theta_k^{(3)}\right).$$

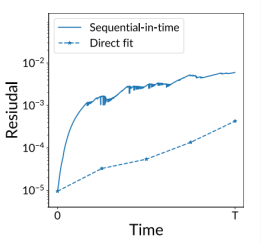
- Assez proche d'un RK classique mais la matrice de masse doit être recalculer a chaque fois.
- Schémas implicite ?
- un type de schéma assez simple appelé linéairement implicite peuvent être utilisé ici.

Méthode de "Randomization"

- On repart de

$$\theta_{k+1} = \min_{\theta} \int_{\Omega} \| \left(\nabla_{\theta_n} nn_{\theta_n}\right)(\theta - \theta_n) - \Delta t G(nn_{\theta_n}(\mathbf{x})) \|_2^2$$

- En pratique, puisqu'on utilise souvent des réseaux sur-paramétrés on a des paramètres $\theta(t)$ redondant. Imaginons qu'on 2 fois trop de poids on va avoir une matrice 4 fois plus grosse à inverser ou le cout de résolution du problème au moindre carrés sera 4 fois plus grande.



- **Référence**: *Randomized Sparse Neural Galerkin Schemes for Solving Evolution Equations with Deep Networks.*
- Entrainement successif d'un réseau vs entrainement point a point en temps.

Méthode de ”Randomization” II

- L'idée des méthodes de ”randomization” consiste a ne pas faire évoluer tout les poids a chaque étape en temps.

Avantage

- On résout un problème de plus petite taille.
- On évite les instabilités en temps du aux paramètres redondants.

- Algorithme:
  - On se donne une matrice S_k qui sélectionne m paramètres.
  - On résout
$$\min_{\Delta\theta_s^{(k)}} \left\| (\nabla_{\theta_n} nn_{\theta_n})(\theta^{(k)}) S_k \Delta\theta_s^{(k)} - G(\theta^{(k)}) \right\|_2^2$$
  - La mise a jour des poids sélectionnée devient celle de l'ensemble des poids $\Delta\theta^{(k)} = S_k \Delta\theta_s^{(k)}$.
  - On met a jour les poids: $\theta_{k+1} = \theta_k + \Delta\theta_s^{(k)}$