Implémentation et test d'une paramétrisation stochastique de la convection dans ARPEGE (version optimisée)

Blanka Balogh[1], David Saint-Martin[1], Olivier Geoffroy[1], Aziz Bhouri[2] & Pierre Gentine[2]

[1]DESR/CNRM/GMGEC/ATMO
[2]Dept. of Earth & Environment Engineering, Columbia University

08/02/2024


Sommaire

Contexte

Data-driven parameterizations

- De plus en plus de paramétrisations IA performantes, mais encore peu de test *online*.
- Couplage fortran/python : pas de solution unanime.
- La plupart des tests online montrent que les paramétrisations IA sont **instables**.

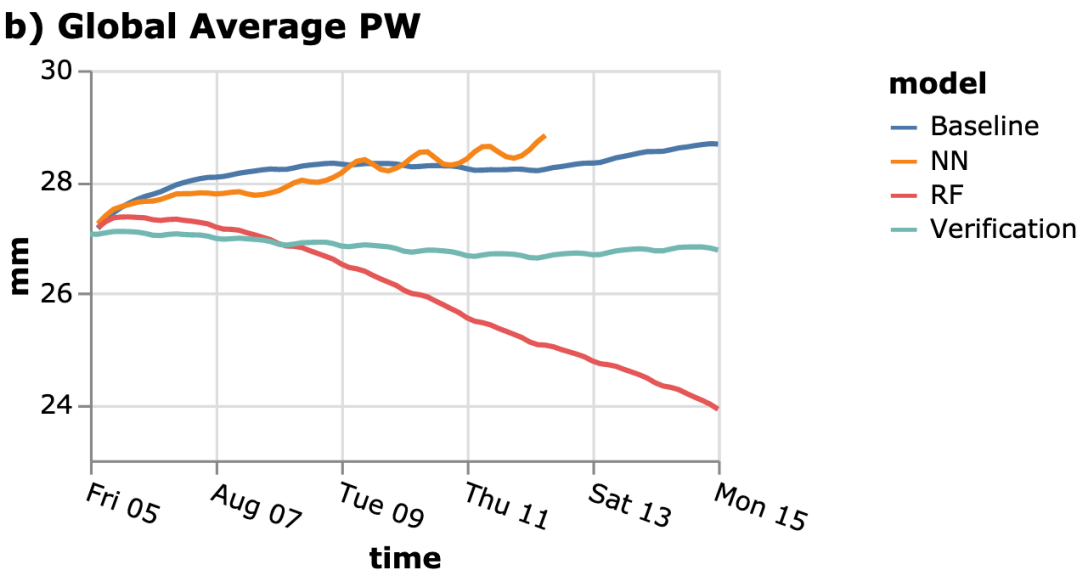


Figure 1: Évolution de l'eau précipitable dans un modèle de climat, lors d'un test *online*. (Brenowitz, Henn et al., 2020)

Contexte

NN utilisé : Bhouri et al., 2023 (preprint), pour la convection totale.

- Approche multi-fidélité : données d'une simulation SPCAM sans réchauffement, +4K et +8K ;
- *Random Prior Networks* ;
- Ensemble de 128 NNs feedforward.

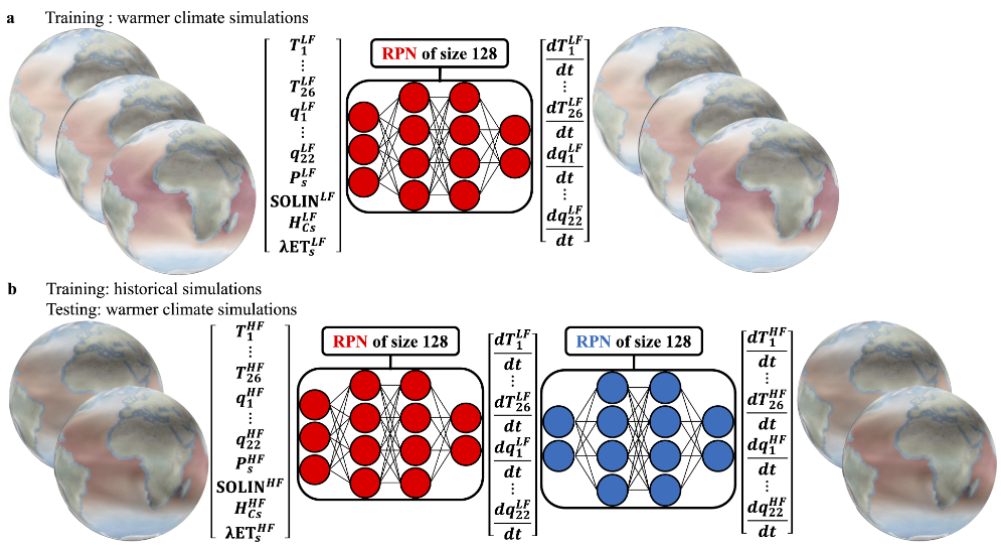


Figure 2: Entraînement de l'ensemble de NN. (Bhouri et al., 2023 (preprint))

Objectifs

Objectifs

Réaliser le test *online* d'un schéma de paramétrisation IA pour la convection, dans ARPEGE (version optimisée) (collab. Columbia University).

Principales difficultés anticipées

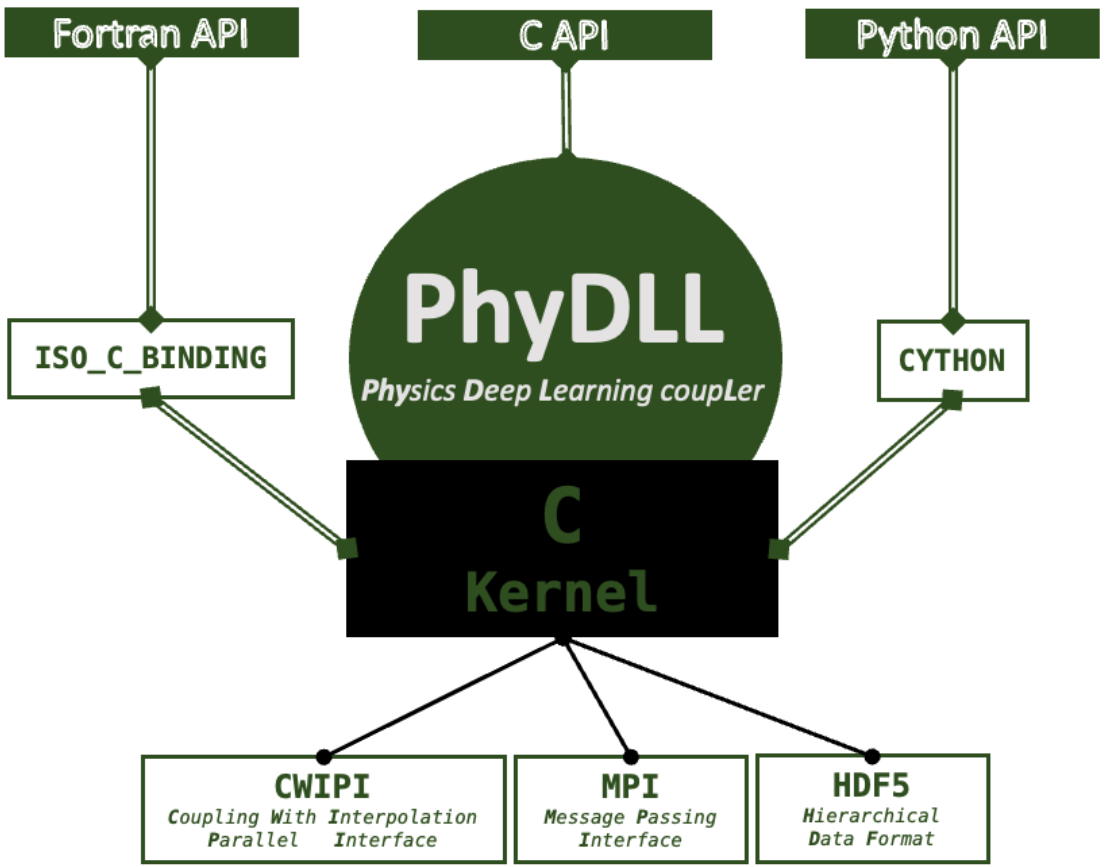
1. SPCAM 5 a une résolution verticale différente de celle d'ARPEGE.
2. Couplage python/fortran dans ARPEGE.
3. La paramétrisation IA a été entraîné avec des données SPCAM 5 : équivalence des variables ?

→ Dans quelle mesure et comment peut-on surmonter ces difficultés ?

Sommaire

Couplage fortran/python – 1/3

Avant : réseaux simples, implémentation manuelle.
Maintenant : implémentation de **PhyDLL** (A. Serhani, C. Lapeyre & al., CERFACS)



Couplage fortran/python – 2/3

PhyDLL simple d'utilisation dans python...

```python
def main():
    """
    Main coupling routine
    """
    phydll = PhyDLL(coupling_scheme="DS",
                    mesh_type="NC",
                    phy_nfields=52,
                    dl_nfields=48)

    while phydll.fsignal:
        # Recv Phy fields
        phy_fields = phydll.receive_phy_fields()

        # Inference
        dl_fields = your_predict_fct(phy_fields)

        # Send DL fields
        phydll.send_dl_fields(dl_fields)

    # Finalize
    phydll.finalize()


def your_predict_fct(fields):
    """
    Dummy inference function
    """
    pred_dT_dq = predict_H(fields.T)*sigma_MF_out[None,:,:]+mu_MF_out[None,:,:]
    pred_dT_dq = onp.median(pred_dT_dq,axis=0)

    return pred_dT_dq.T


if __name__ == "__main__":

    main()
```

... Mais nécessite de nombreux ajouts et modifications dans ARPEGE (version optimisée) → choix d'utiliser la version optimisée et améliorée d'ARPEGE (Geoffroy & Saint-Martin, 2024).

Couplage fortran/python – 3/3

Bilan

On peut faire tourner n'importe quel NN en python dans ARPEGE (version optimisée), mais :

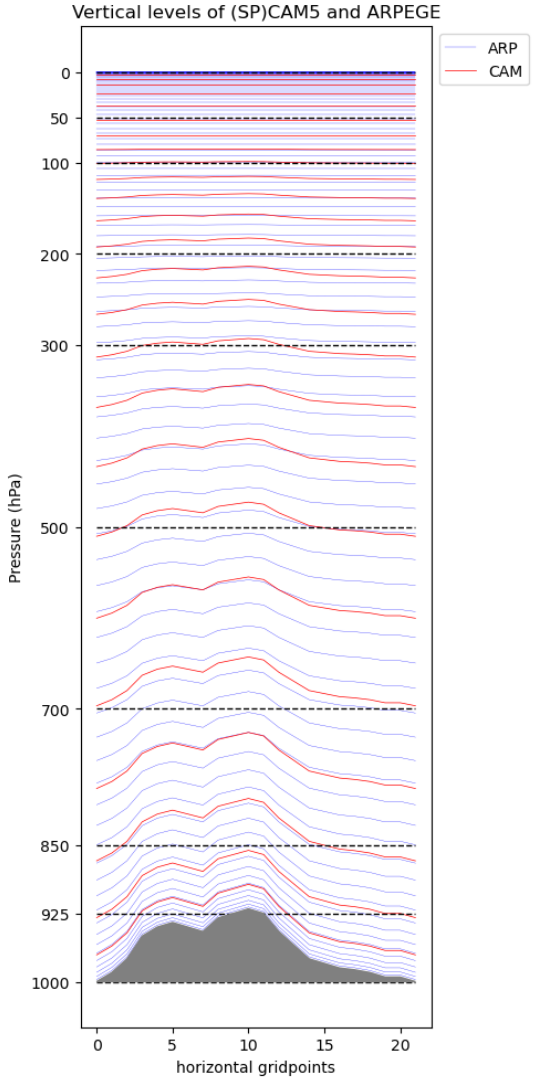
- Exécution du script python : **très** coûteux avec le NN utilisé (20 fois plus coûteux que la paramétrisation dans ARPEGE (version optimisée)).
- Communication MPI entre ARPEGE-Clim, PhydII et XIOS impossible ? $\rightarrow$ Utilisation de XIOS sur les procs ARPEGE.

SPCAM vs. ARPEGE (version optimisée)

→ Grille verticale :

- 50 niveaux dans ARPEGE (version optimisée) ;
- 26 niveaux dans SPCAM.

Interpolation ? ARPEGE (version optimisée) avec la grille verticale de SPCAM ?



SPCAM vs. ARPEGE (version optimisée)

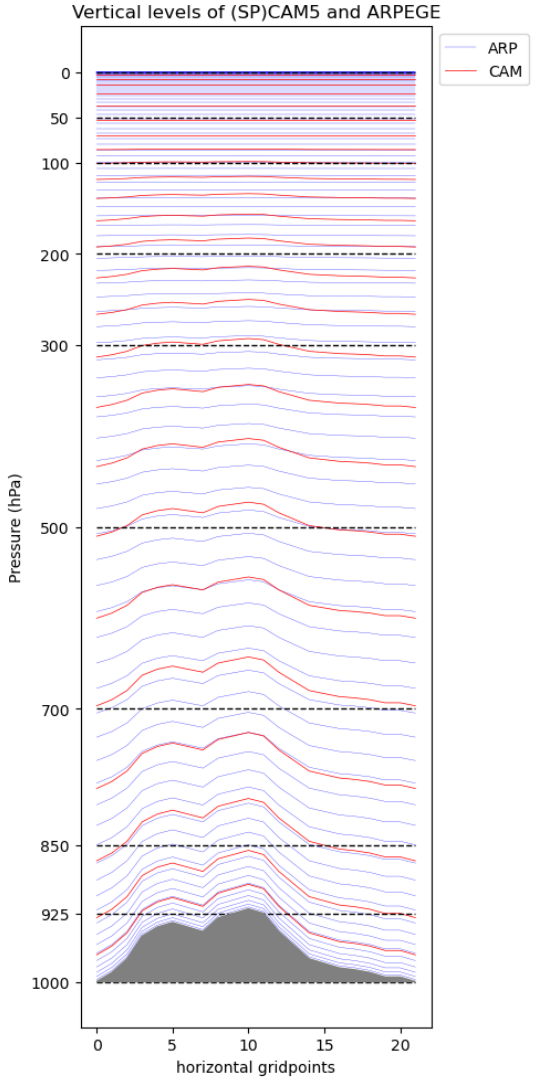
→ Grille verticale :

- 50 niveaux dans ARPEGE (version optimisée) ;
- 26 niveaux dans SPCAM.

Interpolation ? ARPEGE (version optimisée) avec la grille verticale de SPCAM ?

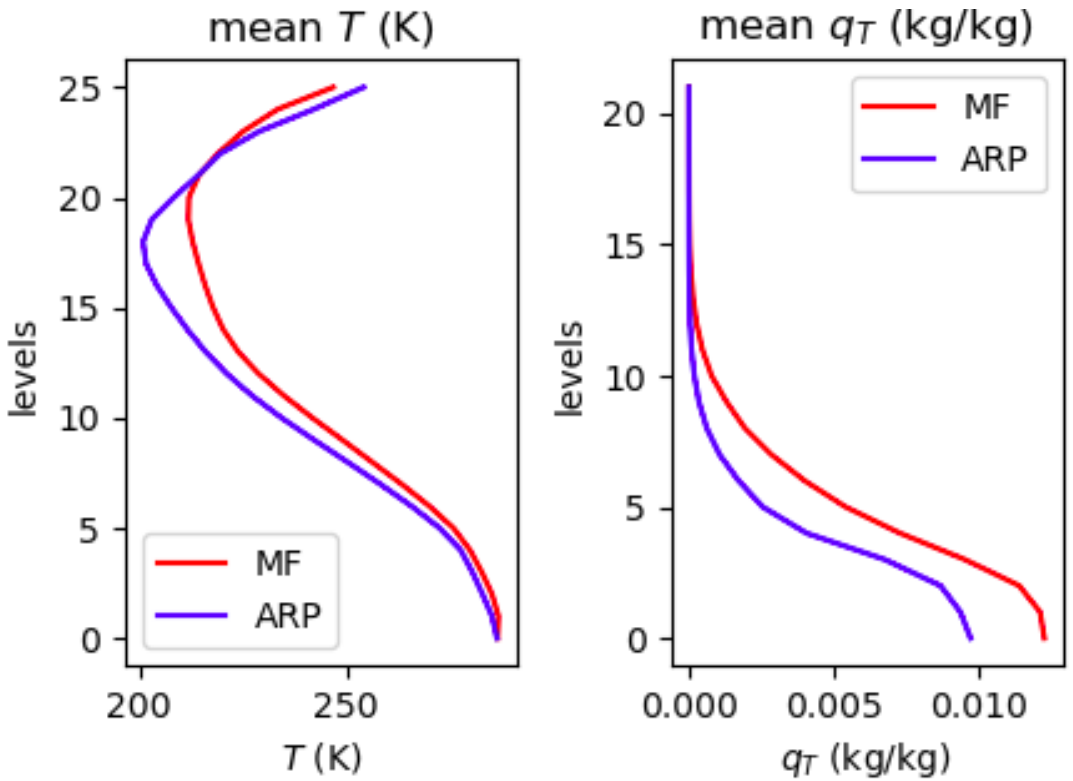
Solution retenue

Nouvelle configuration d'ARPEGE (version optimisée) TCo179 avec 26 niveaux verticaux.



Équivalence de variables – 1/2

En entrée du NN. Moyennes globales (SP)CAM vs. ARPEGE (version optimisée) 26 niveaux.



mean SURFP MF : 9.68e+04 hPa
mean SURFP ARP: 9.86e+04 hPa

mean RSDT MF : 298 W.m^(-2)
mean RSDT ARP: 300 W.m^(-2)

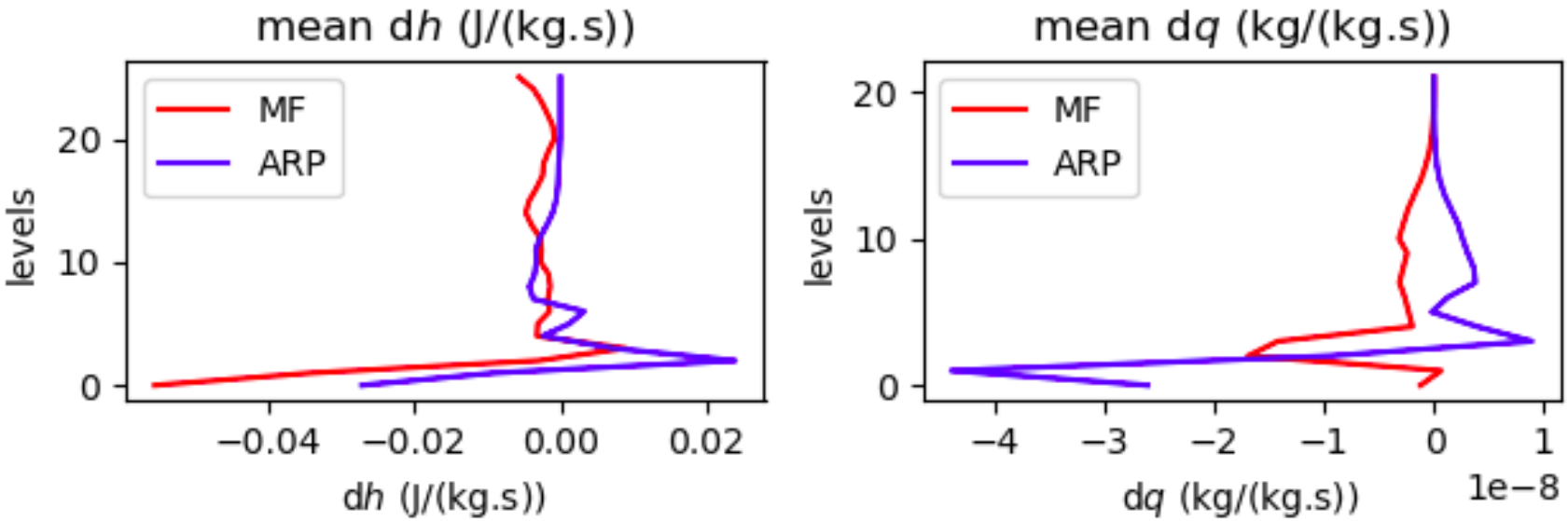
mean SHFLUX MF : 11.5 W.m^(-2)
mean SHFLUX ARP: 29.7 W.m^(-2)

mean LHFLUX MF : 81.2 W.m^(-2)
mean LHFLUX ARP: 85.3 W.m^(-2)

A peu près OK, sauf pour les flux de surface de chaleur sensible (ARP: 29.7 W/m², (SP)CAM : 11.5 W/m²). Humidité ?

Équivalence de variables – 2/2

En sortie du NN



- Tendances de température (gauche) OK
- Tendances d'humidité (droite) – mauvaise équivalence ?
- Il manque les précipitations : à diagnostiquer.

Test offline – exemple

Intégrale des tendances du NN $\sim$ précipitations.

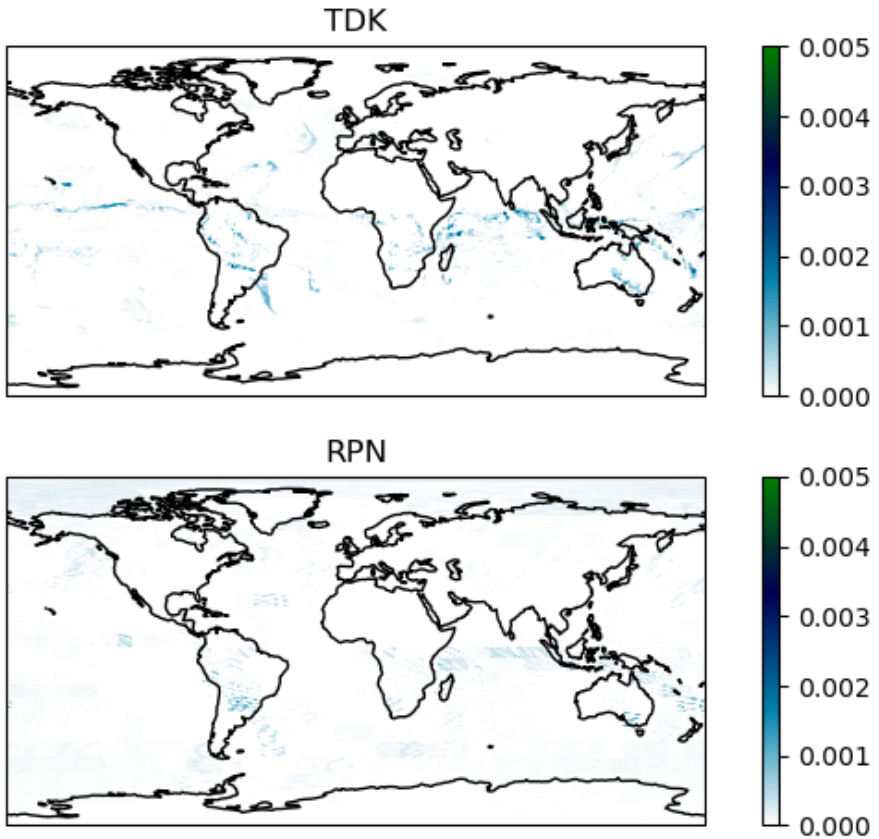


Figure 3: Précipitations calculées avec la paramétrisation physique d'ARPEGE (version optimisée) (haut) et le NN (bas).

Sommaire

Conclusion – 1/2

Préparation de l'implémentation

- Nous avons implémenté un NN compliqué écrit en python dans ARPEGE (version optimisée), grâce au module PhyDLL.
- Pour réaliser le test *online*, une nouvelle configuration d'ARPEGE (version optimisée) a été développée, plutôt que d'interpoler les I/O du NN.
- L'exécution du script python *online* est coûteux – à améliorer.
- Le problème de communication entre les 3 acteurs (ARPEGE, XIOS, PhyDLL) suppose plusieurs ajouts dans chacun des modèles : fastidieux.

Conclusion – 2/2

Vers un test online ?

- Les résultats *offline* montrent des tendances et précipitations plus intenses avec la paramétrisation NN.
- Résultats offline encourageants, test *online* prochainement.

→ Jeudi du climat (séminaire interne au CNRM/GMGEC) le 29/02 sur les aspects techniques.

Merci pour votre attention !