Outil de comparaison de paramétrisations en « offline »

Sébastien Riette
AMA, 11/13 mars 2019

Motivations

- Dépendance au pas de temps de simulations réalisées avec Méso-NH
- Dépendance vue en particulier sur les champs de pluie au sol et/ou lorsque la grêle pronostique était activée
  => microphysique suspectée
- Tests avec des simulations 2D mais interactions trop fortes avec les autres paramétrisations et la dynamique
- Besoin d'un outil pour caractériser la dépendance et aider à la corriger

METEO FRANCE

Plan de la présentation

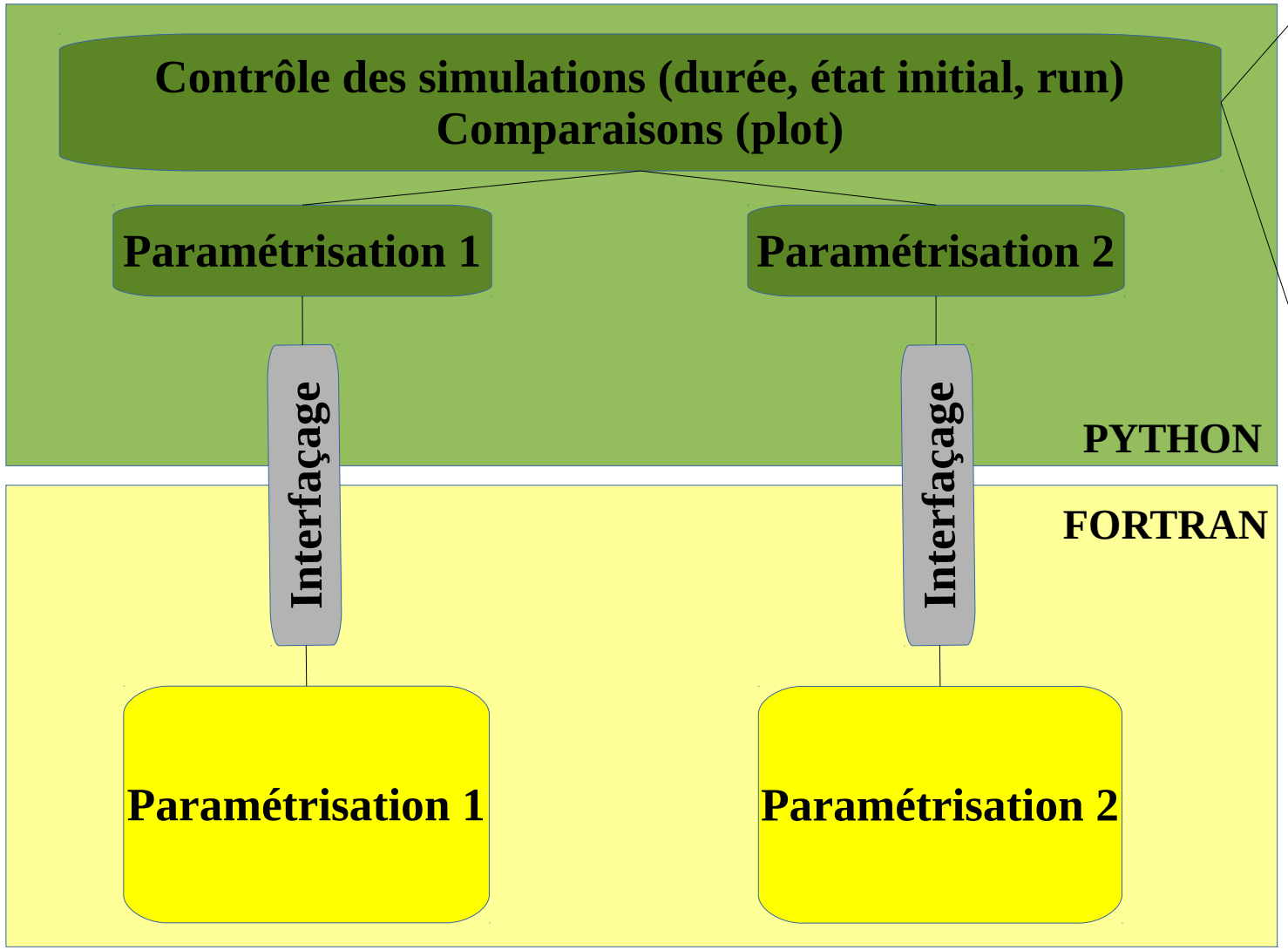
- Description de l’outil
- Application aux processus microphysiques
- Application à la sédimentation
- Perspectives

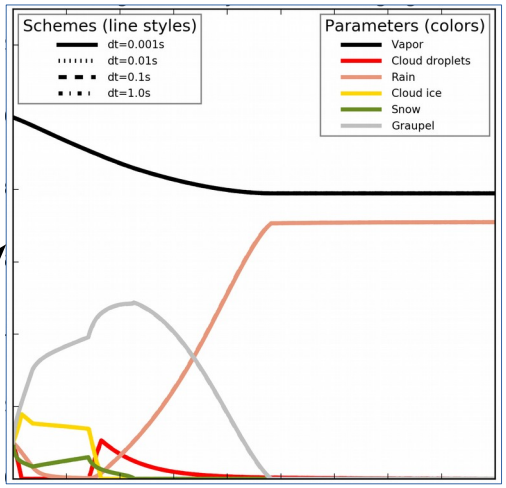
Choix techniques

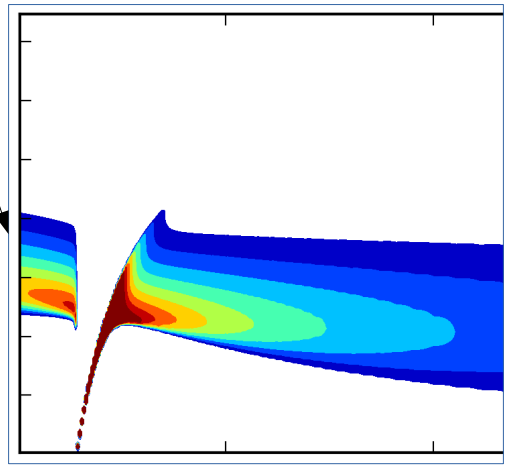
- L’outil doit
  - Isoler complètement une paramétrisation
  - Réutiliser les paramétrisations FORTRAN
  - Réaliser une avance temporelle
  - Effectuer des calculs de diagnostics et des tracés

  => Script python appelant des routines FORTRAN

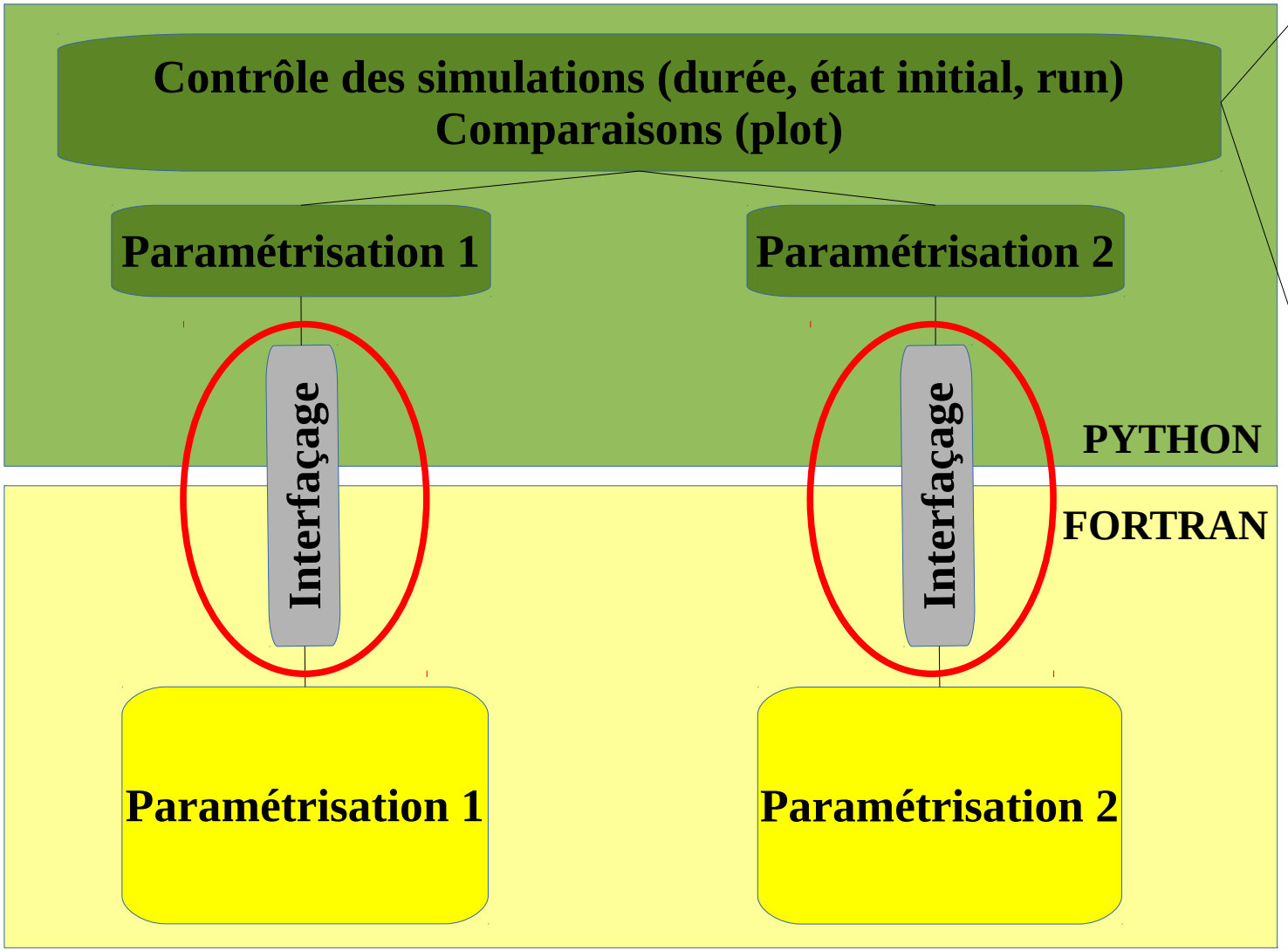
Organisation générale de l'outil

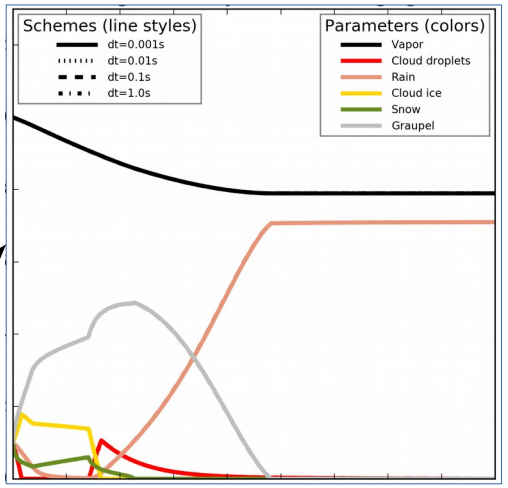


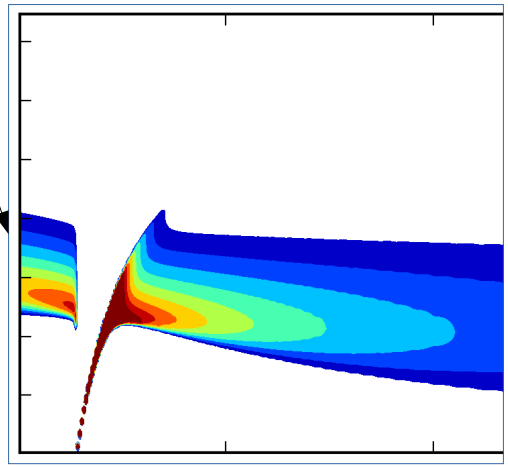




Organisation générale de l'outil







L'interfaçage est l'étape délicate et est assez technique (compilation, 32/64 bits, little/big endian...)

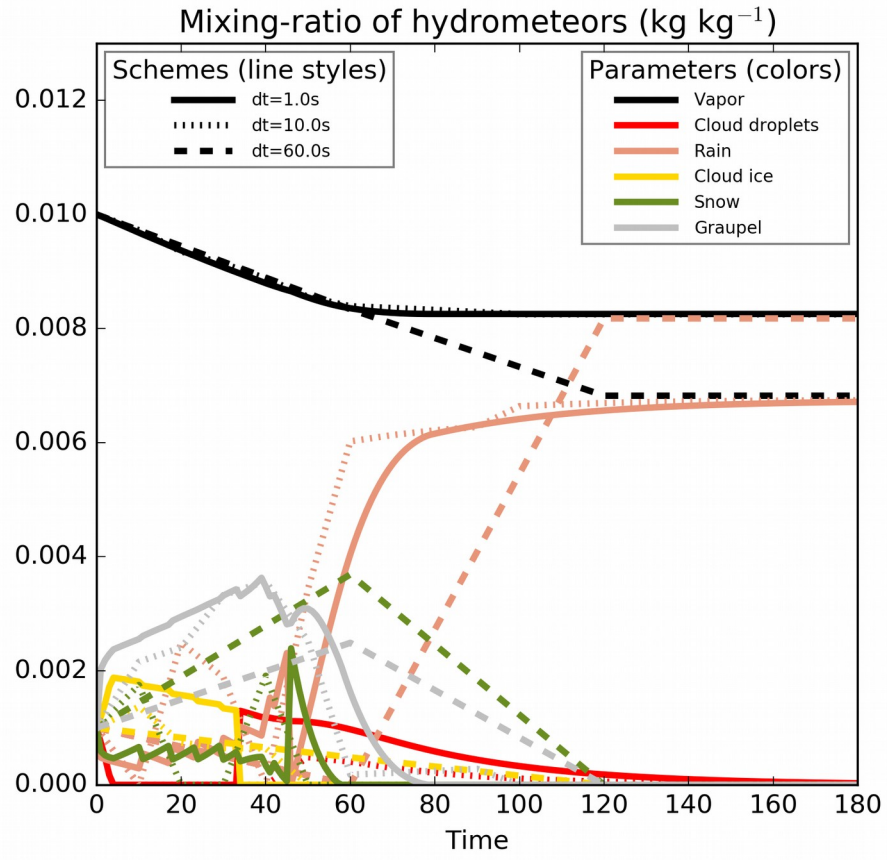
Première application

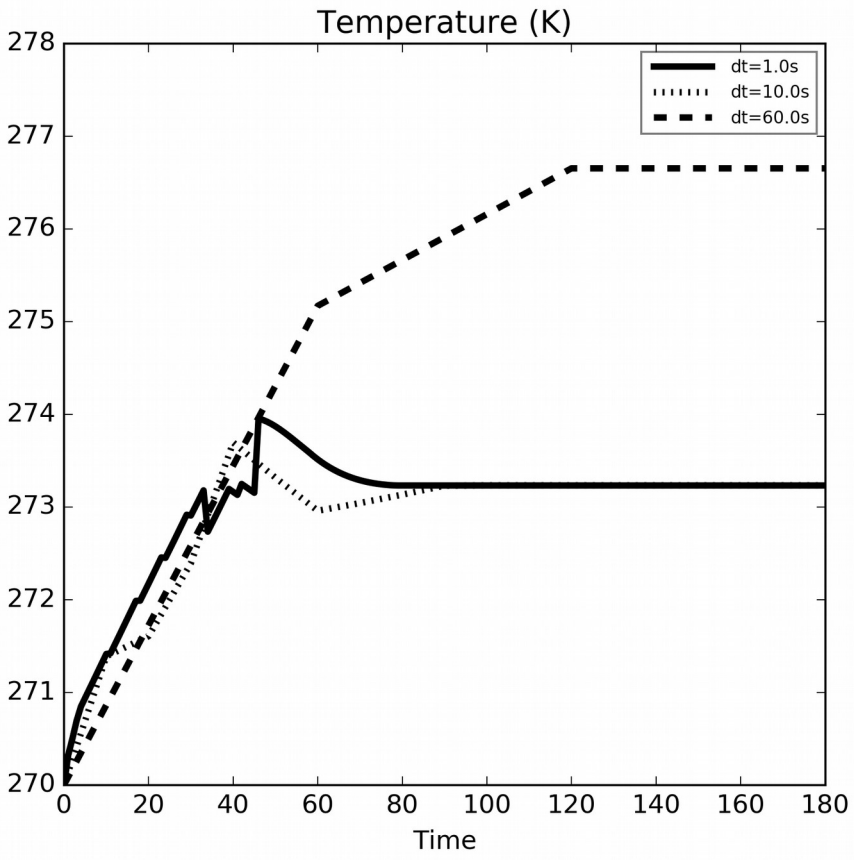
Application de l’outil aux processus microphysiques

Description du test

- Approche 0D
  - on définit un état initial : T=270K, rv=10g/kg, rx=1g/kg
  - on choisit plusieurs pas de temps
  - on réalise des intégrations indépendantes pour chaque pas de temps avec une sortie tous les pas de temps (seule la tendance de la microphysique intervient dans l'avance temporelle)

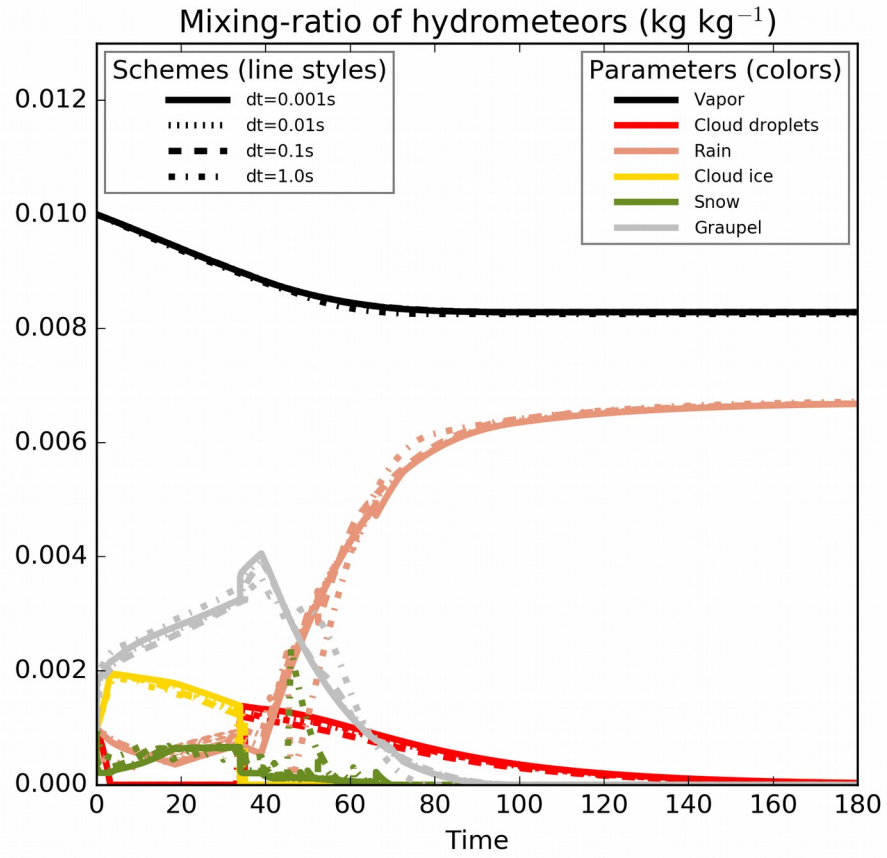
Situation de départ : grands pas de temps

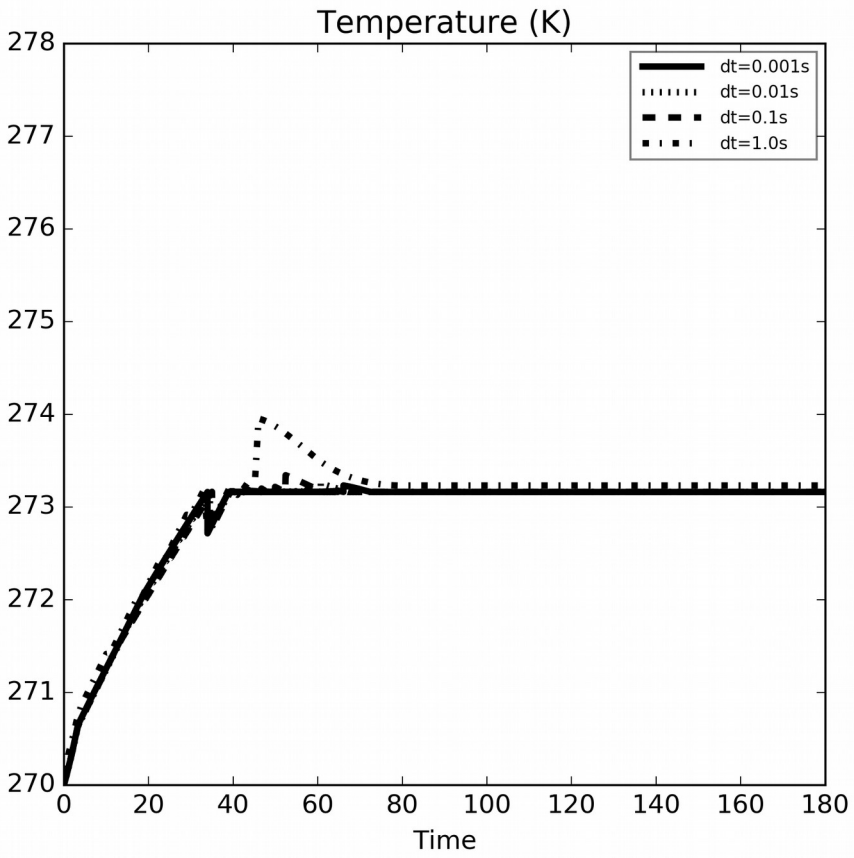




Grande dépendance au pas de temps
=> Recherche d'une convergence en diminuant le pas de temps

Situation de départ : petits pas de temps





On est proche de la convergence avec de très petits pas de temps
Mais encore des oscillations

Bilans de chaleur

- Oscillations dues à la non prise en compte de la rétroaction sur la température lors des changements de phase
- Exemple de la fonte :
  - Fonte activée par température positive
  - La fonte induit une diminution de la température
    - Sans bilan de chaleur, la fonte est totale, la température devient négative et un regel est activé au pas de temps suivant (=> oscillations)
    - Avec le bilan, la fonte s'arrête lorsque la rétroaction induit un changement de signe de la température
- Tous les processus de changement de phase sont concernés

- Avec ces bilans, les simulations réalisées avec de petits pas de temps sont convergentes

Autres modifications

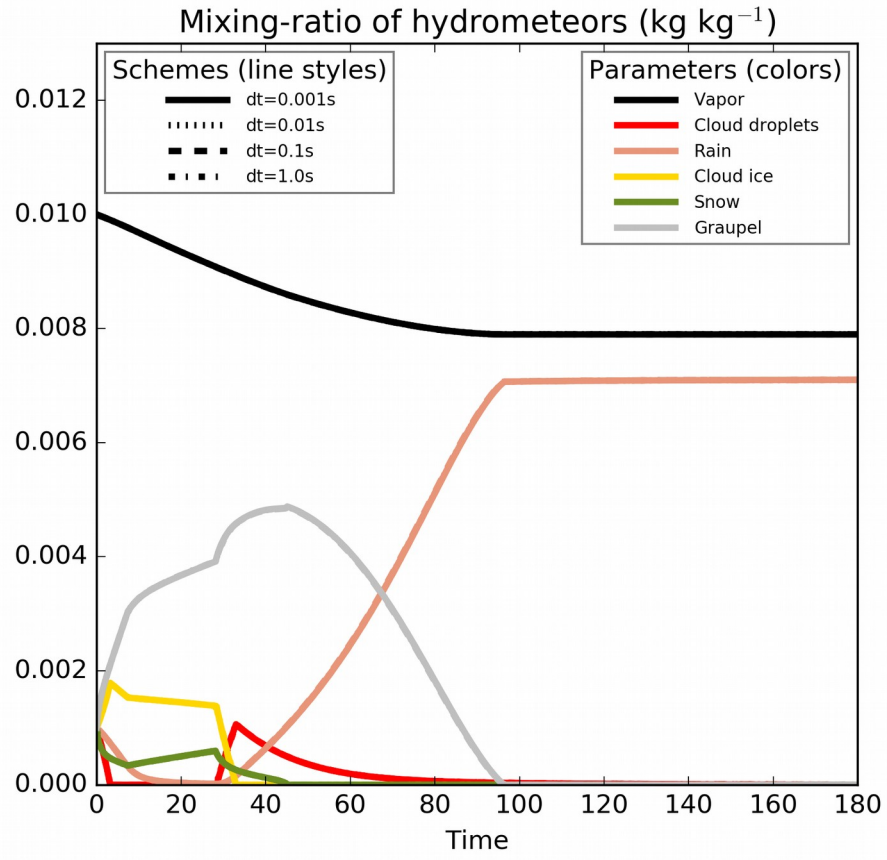
- Givrage de la neige par les gouttelettes de nuage
  - n'était pas fonction du contenu en eau nuageuse
  - remplacé par la formulation de Murakami (1990)
- Revisite de la croissance du graupel
  - choix du mode de croissance plus continu (améliore la continuité par températures légèrement négatives)
  - activation du « water-shedding » par température positive (améliore la continuité par températures légèrement positives)
- Modifications concernant la grêle
  - taux de conversion du graupel en grêle changé (mais pas de validation scientifique pour l'instant)
  - processus calqués sur ceux du graupel

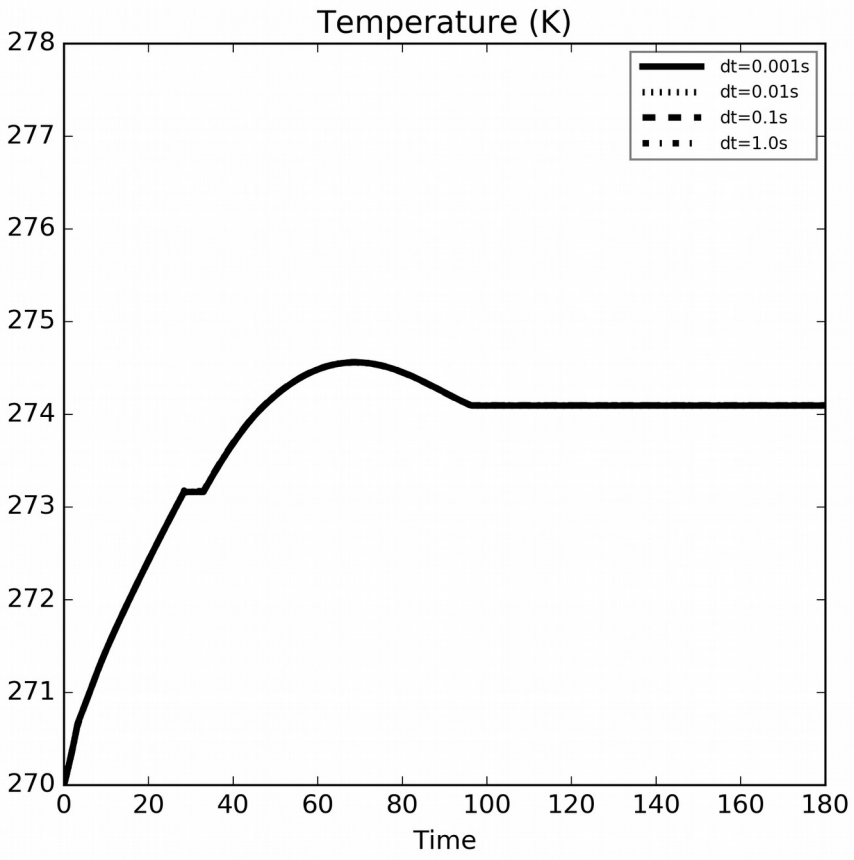
Bilan / Limitations

- Bilan des modifications
  - forte réduction de la dépendance au pas de temps de chacun des processus
  - toutes les modifications (sauf conversion en grêle) ont été faites en essayant d'améliorer la représentation physique
  - la grêle pronostique est maintenant utilisable (mais demande à être réglée)
- Limitations
  - ces modifications seules ne permettent pas de traiter le problème de dépendance liées à la compétition entre les processus => convergence jusqu'à 10s de pas de temps
  - il a été nécessaire de mettre en place un découpage en sous pas de temps au-delà de 10s

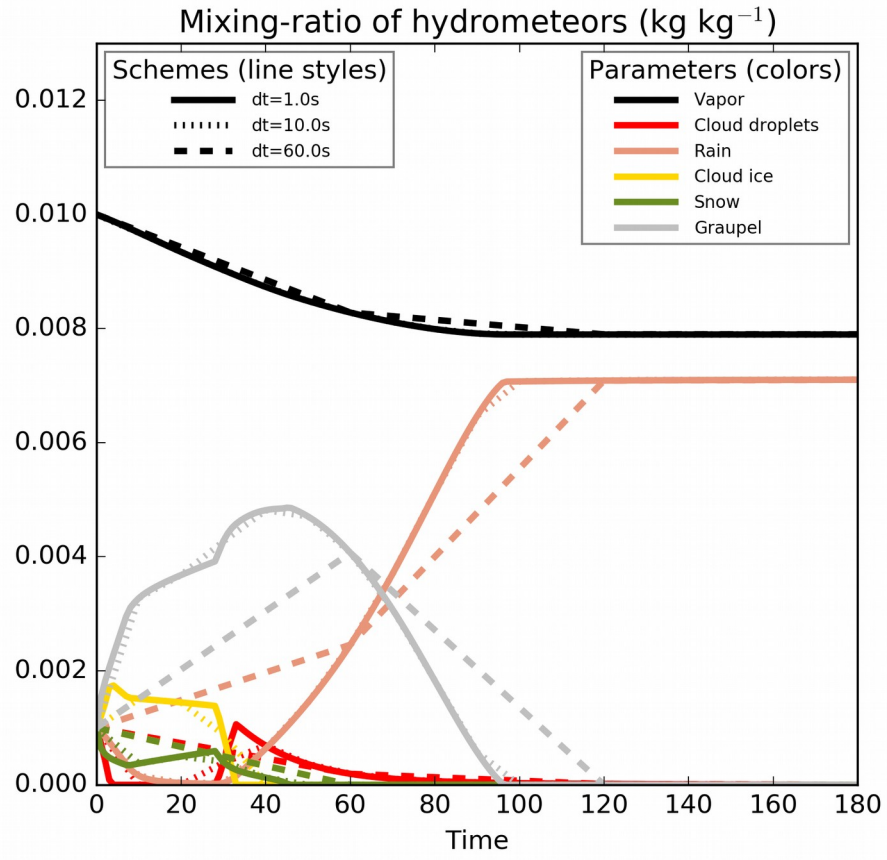
METEO FRANCE

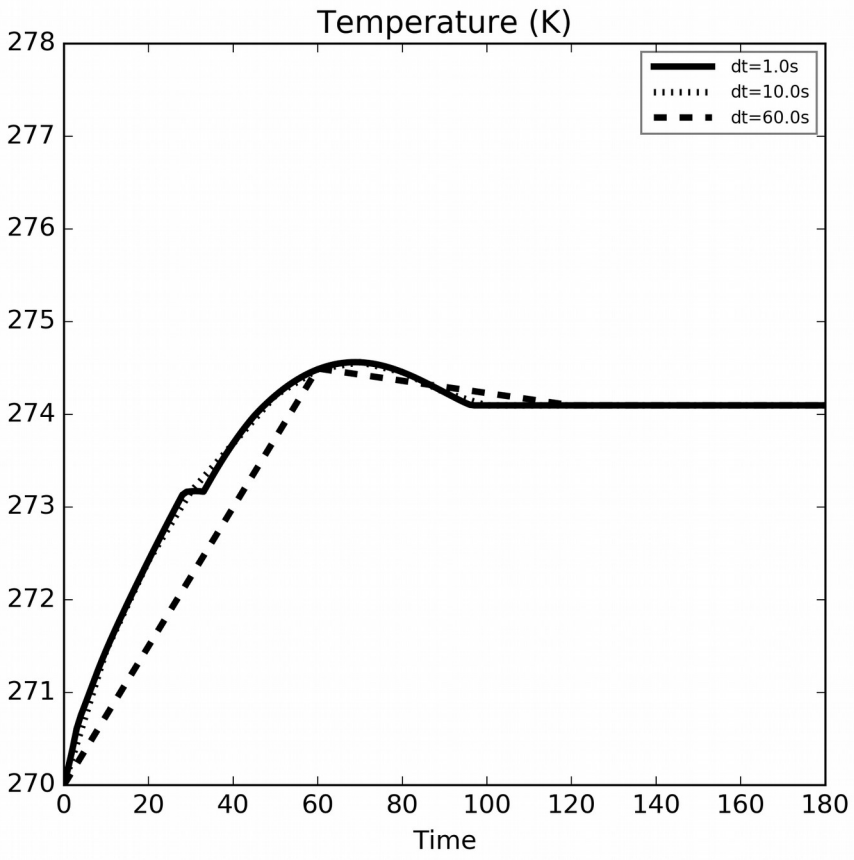
Résultat : petits pas de temps





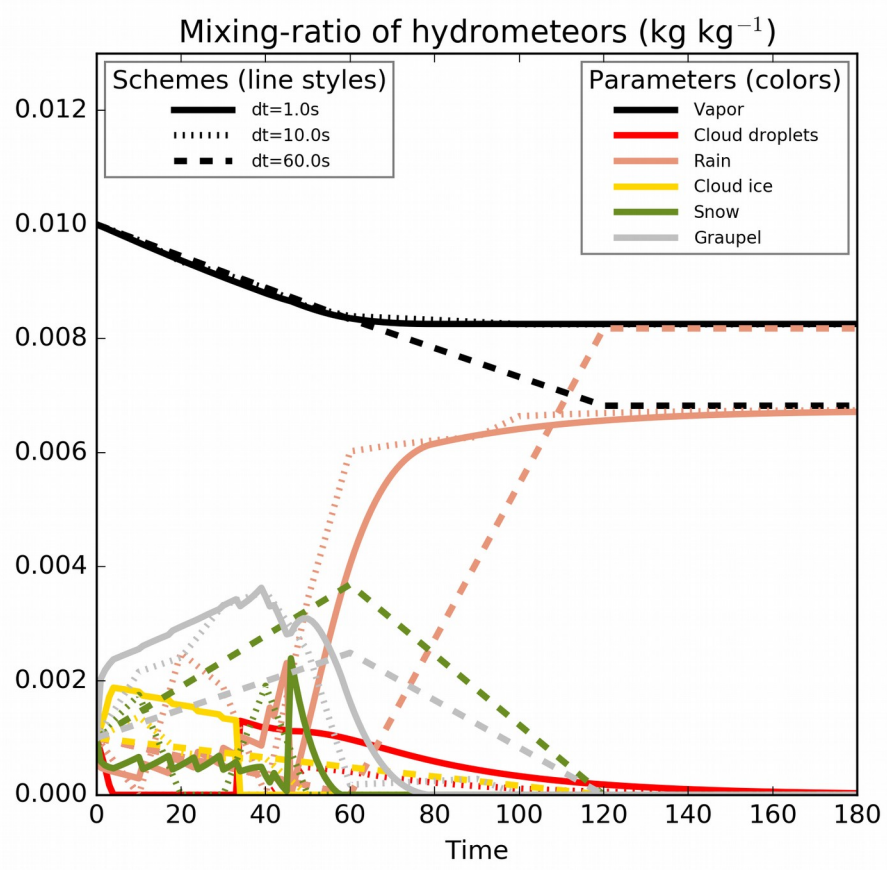
Résultat : grands pas de temps



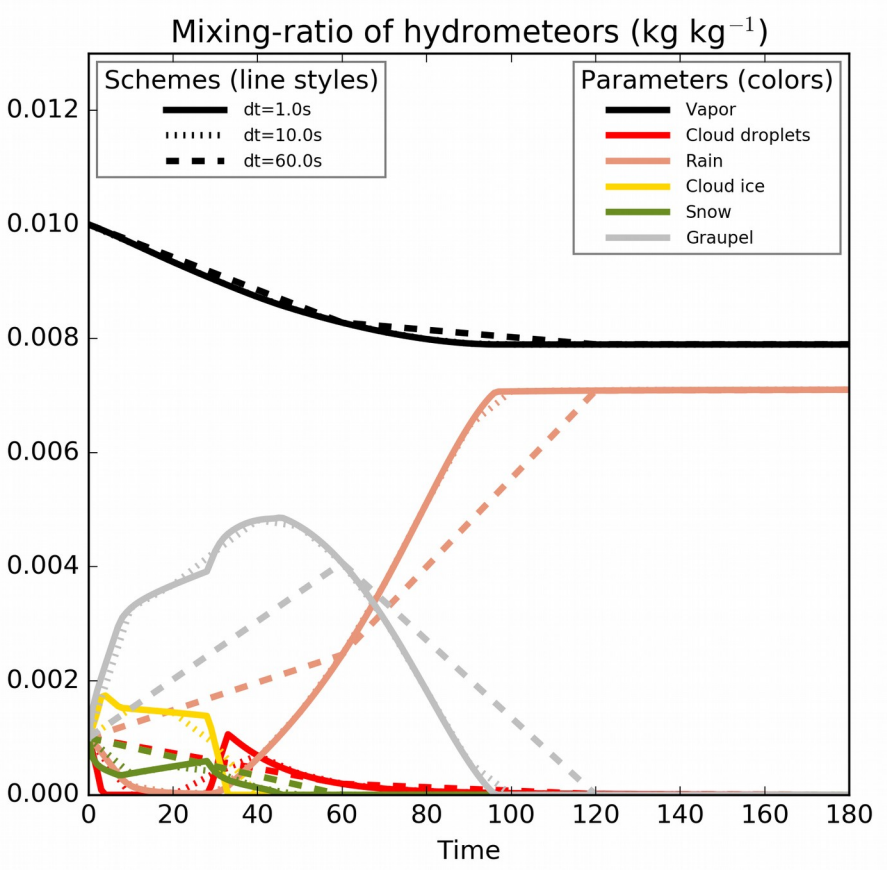


Comparaison avant / après

Avant



Après



Deuxième application

Application de l’outil à la sédimentation

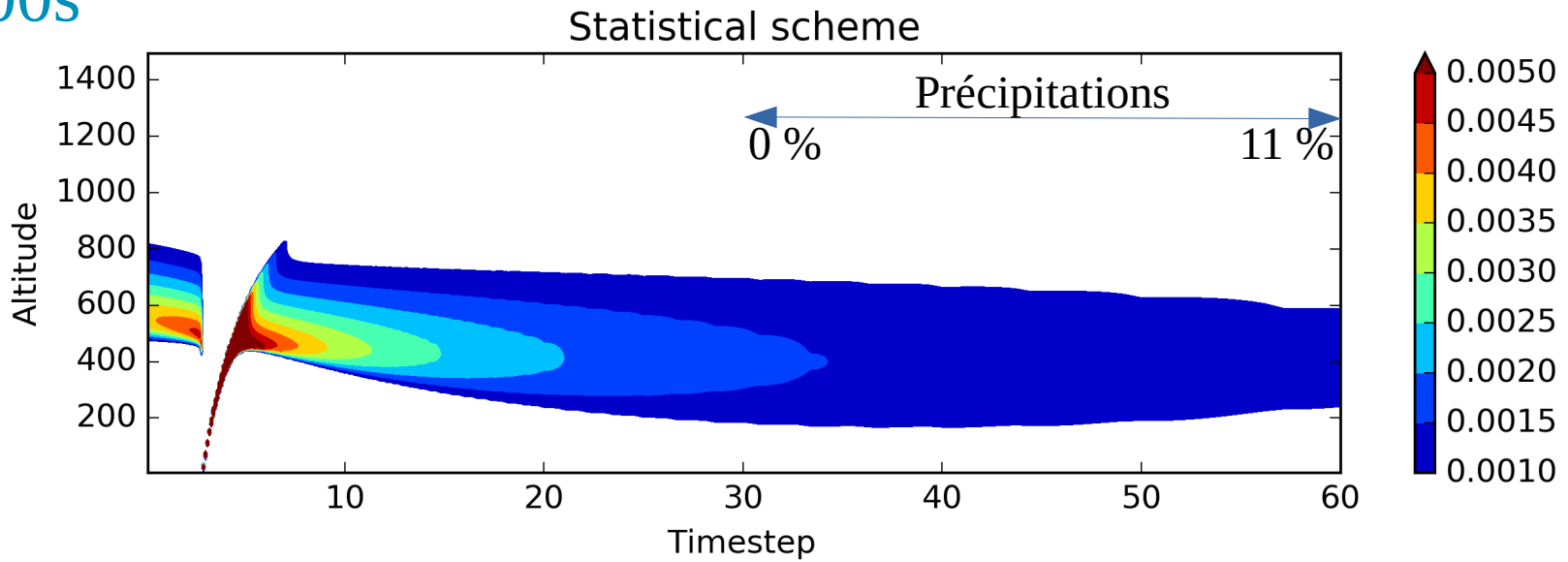
Description du test

- Approche 1D
  - grille verticale régulière (Δz=10m)
  - 0.1g/kg de pluie placée à 1400m de hauteur
  - sédimentation pendant 400s avec différents pas de temps

- Avec les paramètres utilisées dans le schéma :
  - cette quantité de pluie tombe à 3.3m/s
  - la zone avec de la pluie devrait tomber de 1320m en 400s
  - avec l'épaisseur de maille utilisée, CFL=1 pour Δt=3s

Situation actuelle : schéma opérationnel (Bouteloup, 2010)

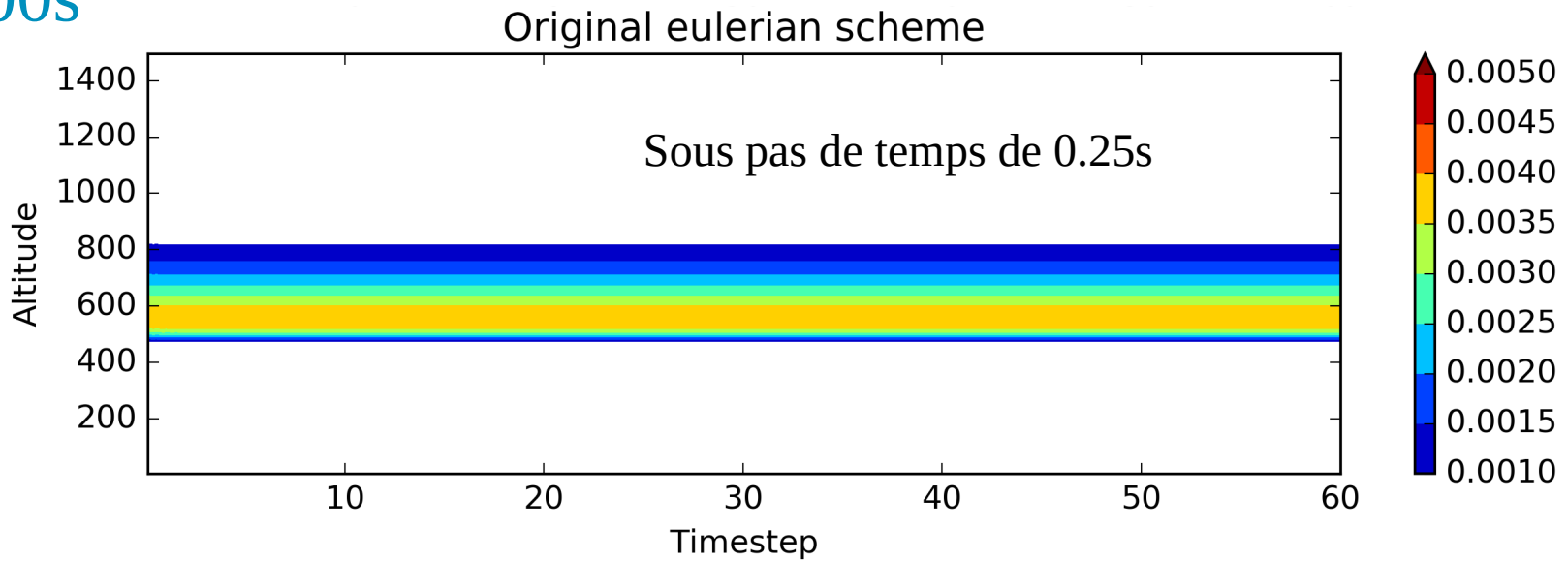
Après 400s



- Schéma statistique
  - forte dépendance au pas de temps
  - diffusion sur la verticale
  - retard dans l'arrivée au sol
  - bonne performance numérique (temps de calcul)

Situation actuelle : schéma eulérien

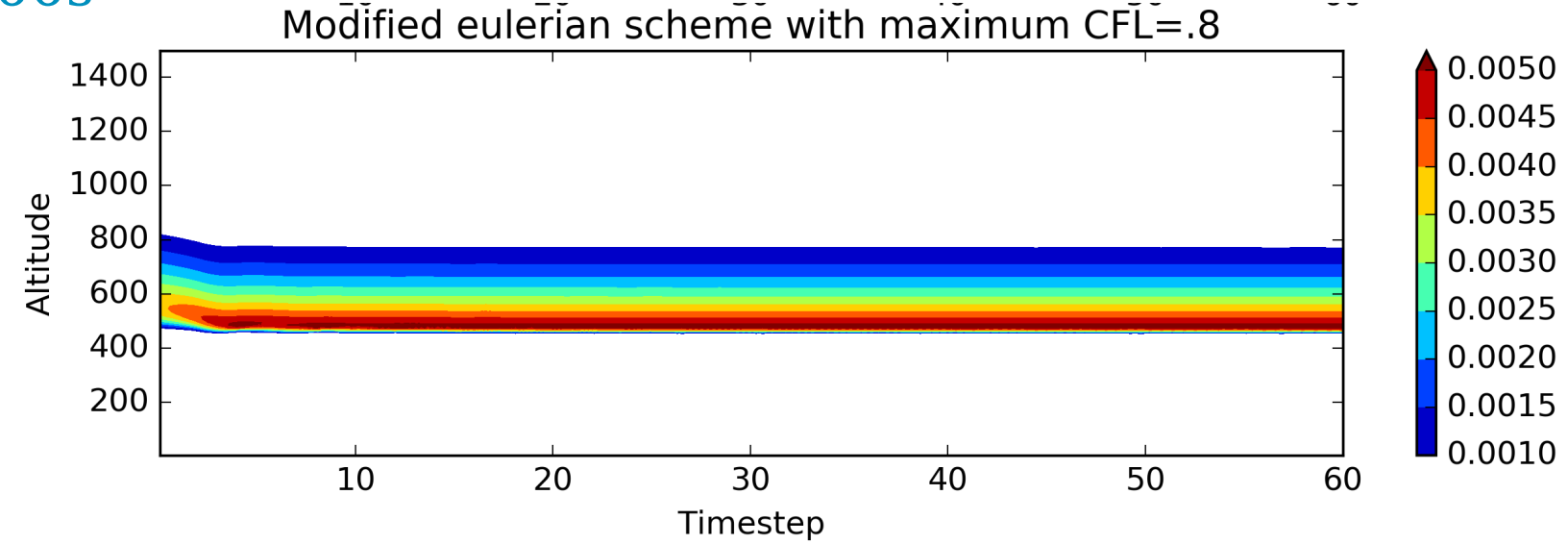
Après 400s



- Schéma eulérien
  - pas de dépendance au pas de temps
  - diffusion sur la verticale
  - retard dans l'arrivée au sol
  - mauvaise performance numérique

Modification proposée

Après 400s



- Schéma eulérien avec sous pas de temps adaptatif
  - très peu de dépendance au pas de temps
  - diffusion sur la verticale toujours présente mais moins forte
  - retard dans l'arrivée au sol
  - performance numérique assez proche du schéma statistique
- Problème ne semble pas pouvoir être complètement résolu tant qu'on ne considère qu'un seul moment

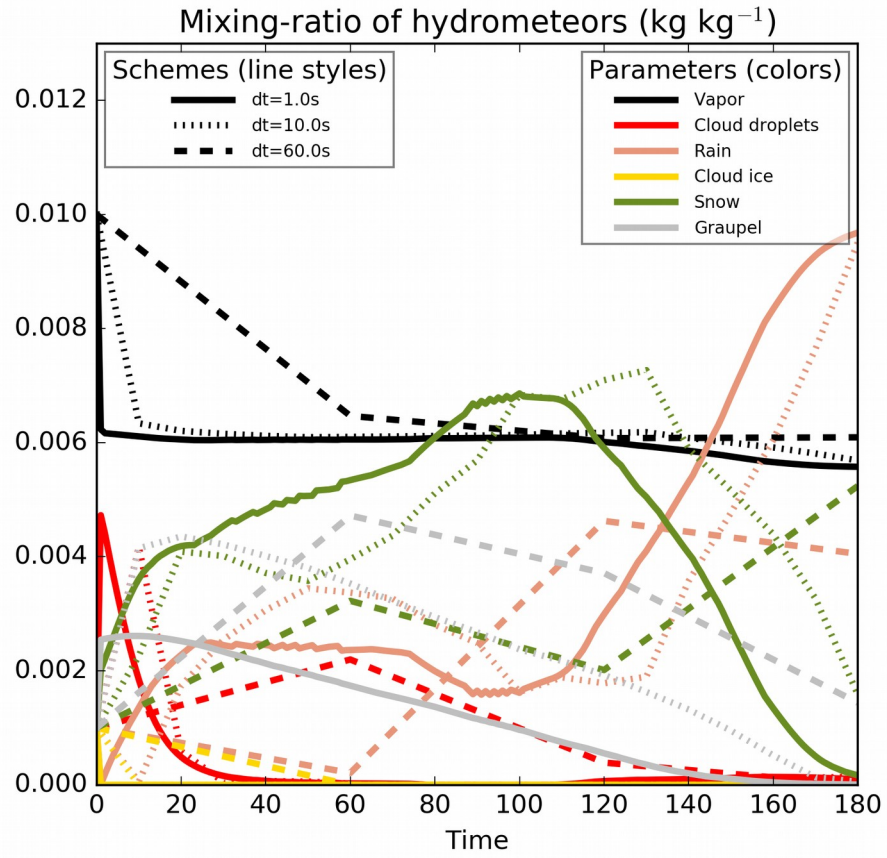
METEO FRANCE

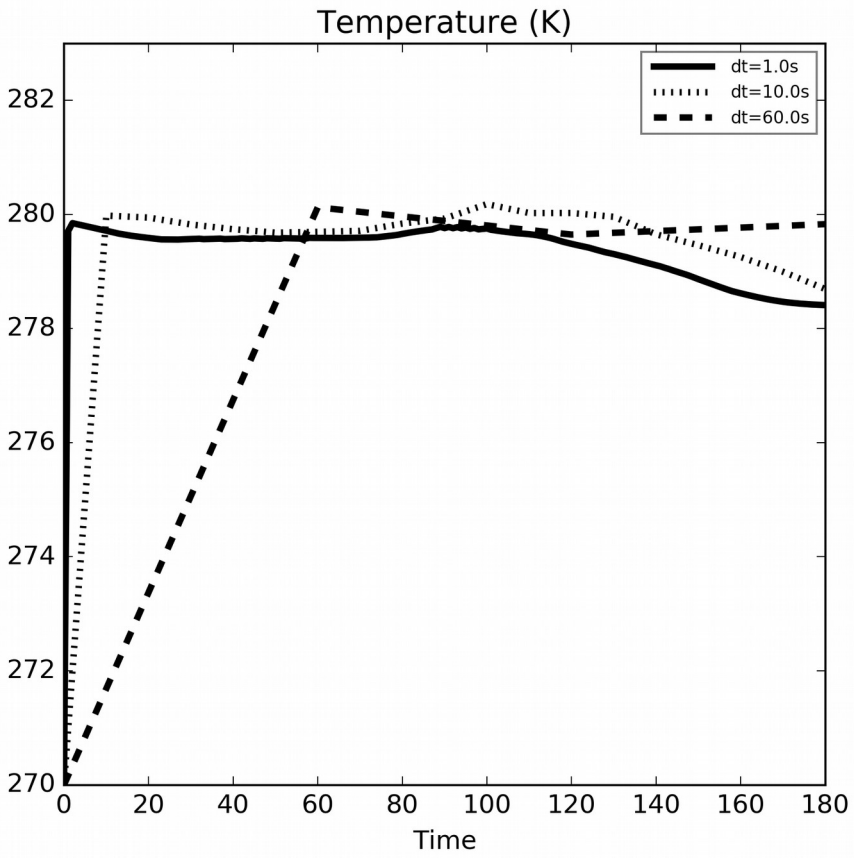
Perspectives

- Aide au développement de paramétrisations
  - microphysique
  - CVPP, turbulence, …

- Outil de comparaison de paramétrisations hébergées par différents modèles :
  - AROME
  - Méso-NH
  - WRF

Premier test avec une microphysique de WRF (Thompson, 2008)





Questions ?

Interfaçage : FORTRAN

- Identification des routines (initialisation, appel)
- Écriture éventuelle d'un enrobage (« assumed shape », 32/64 bits)
- Compilation du modèle et de l'enrobage (avec option -fPIC)
- Édition de liens pour créer une librairie partagée (.so)

Modèle

```
SUBROUTINE INIT(KK)
  INTEGER, INTENT(IN) :: KK
  USE …
  …
END SUBROUTINE INIT

SUBROUTINE PARAM(PI, PO)
  REAL, DIMENSION(:), INTENT(IN) :: PI
  REAL, DIMENSION(:), INTENT(OUT) :: PO
  USE …
  …
END SUBROUTINE PARAM
```

Enrobage

```
SUBROUTINE PARAM_PY(KI, PI, PO)
  INTEGER, INTENT(IN) : KI
  REAL, DIMENSION(KI), INTENT(IN) :: PI
  REAL, DIMENSION(KI), INTENT(IN) :: PO
  CALL PARAM(PI, PO)
END SUBROUTINE PARAM_PY
```

Interfaçage : PYTHON

```
import numpy
from ctypesForFortran import IN, OUT, ctypesForFortranFactory

ctypesFF, _ = ctypesForFortranFactory("./lib.so")

@ctypesFF()
def init(KK):
    return ([KK],          [(numpy.int64, None, IN)],      None)

@ctypesFF()
def param_py(PI):
    KI = len(PI)
    return ([KI, PI),
              [(numpy.int64, None, IN),
               (numpy.float64, (KI, ), IN),
               (numpy.float64, (KI, ), OUT)],
               None)

init(5)
zo = param_py(zi)
```

Ouverture de la librairie partagée

Signature des routines

Utilisation