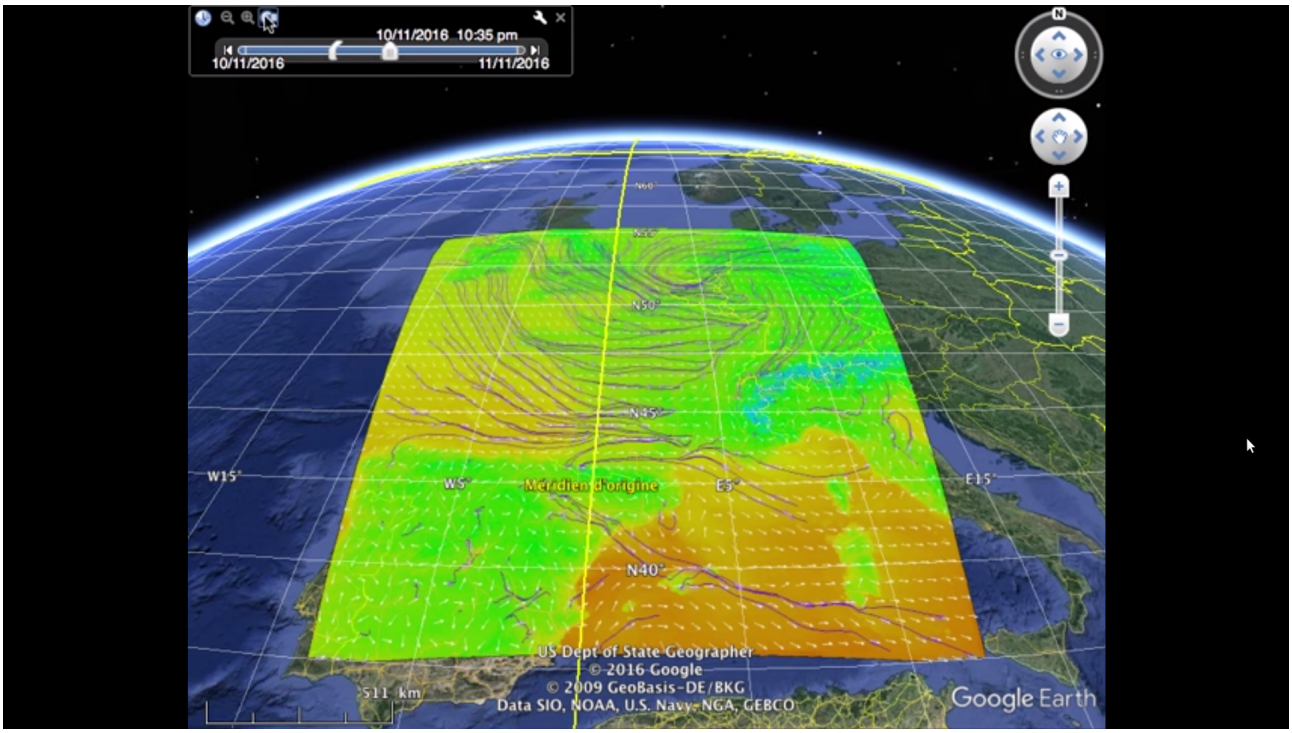
Université Grenoble Alpes **Polytech Grenoble**
RICM5 **Année 2017–2018**

Visualisation — Projet Météo



1 Introduction

Le but de ce projet est de réaliser une visualisation météo dans `Google Earth`. Cette visualisation est calculée dynamiquement à partir de données de simulation Météo France.

Météo France met à disposition publiquement des données issues d'un modèle de prévision. Ces données publiques contiennent un grand nombre de variables en chaque sommet de grilles géographiques. Les prévisions portent sur les prochaines 42 heures et sont mises à jour toutes les heures.

Le projet est à réaliser en binôme et devra être rendu avant le **Mardi 16 Janvier 2018, 23h00**.

2 Étapes à réaliser

2.1 Préparation de l'environnement de travail

Vous devrez installer `Paraview` et `Google Earth` (voir la Section 4 pour les notes sur l'installation).

2.2 Découverte des fichiers fournis

La première étape consiste à découvrir les fichiers fournis (voir Section 3). Ces fichiers sont contenus dans l'archive `Scripts Shell-Python-Paraview` située sur la page `Projet Visualisation 2017/2018`, elle-même accessible depuis la page du cours. Ils vous serviront de point de départ pour réaliser le projet.

2.3 Création du pipeline de visualisation dans `Paraview`

A l'aide des informations fournies en annexe, vous devrez exporter un script `pvpython` à l'aide de `Paraview` servant à créer votre visualisation à partir d'un fichier de résultats de simulation de type `nc`.

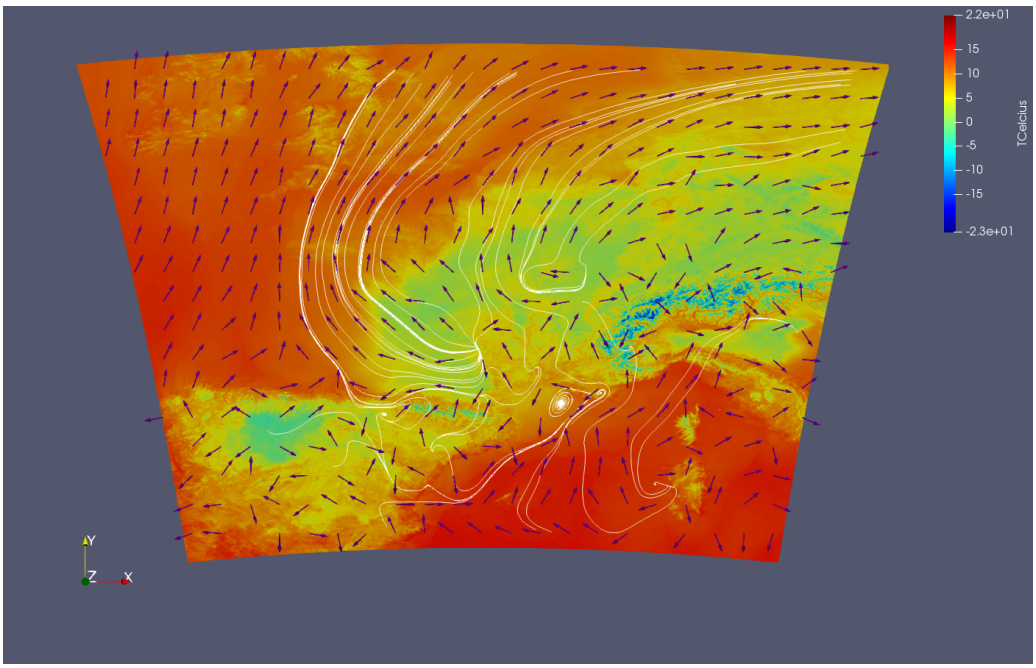


FIGURE 1 – Exemple de résultat obtenu dans Paraview

Ce script servira à générer des cartes de visualisation qui seront ensuite importées dans `Google Earth`. Cette visualisation doit comporter la température ainsi que des flèches indiquant les directions du vent et les lignes de courant du vent. Vous pouvez également ajouter d'autres variables présentes dans les fichiers de simulation (humidité, pluie, neige...).

2.4 Visualisation dans `Google Earth`

En utilisant votre script `pvpython` obtenu à l'étape précédente et en s'inspirant des scripts fournis, vous devrez obtenir une visualisation météo au format `kmz` pouvant être importée directement dans `Google Earth`. Vous devrez rajouter les courbes iso-valeur de température à votre visualisation générée à partir de `Paraview`. Vous devrez également personnaliser votre visualisation, par exemple changer la carte de couleurs de la température ou générer des courbes de contour de température pour des ensembles d'iso-valeurs différents. Vous devrez réaliser un pipeline de sorte que la visualisation au format `kmz` soit générée directement à partir d'un fichier de format `nc`. Vous pouvez également faire en sorte que votre script télécharge automatiquement ce fichier comme dans les scripts initiaux fournis.

2.5 Animation de la prévision météo sur une plage de temps

Réalisez une animation montrant l'évolution de la prévision météo au cours du temps, par exemple sur une journée. Idéalement, vous devriez pouvoir fournir un script prenant en paramètre une plage horaire sur laquelle l'animation doit s'étendre, et celle-ci est ensuite générée automatiquement et exportée au format `kmz`.

2.6 Bonus : Animation de la prévision météo pour une même heure

Il est intéressant de visualiser l'évolution de la prévision calculée par Météo France pour un instant donné. Par exemple visualiser l'évolution de la prévision pour le lendemain à 14h au cours du temps, les simulations étant calculées toutes les heures. Si vous avez terminé les étapes précédentes, réalisez une animation permettant de visualiser l'évolution des simulations au cours du temps pour un instant donné dans le futur.

3 Fichiers fournis

3.1 Description du répertoire

- `DATA` : données téléchargées en format `grib2`, et converties en format `nc` (initialement vide).
- `KML` : fichiers `kml` et `kmz` générés
- `PYTHON` : scripts `python` permettant de télécharger et convertir les données et scripts `pvpython` permettant de créer des visualisations à partir de `Paraview`.

3.2 Scripts `python`

Les scripts suivants sont fournis dans le dossier `python`. Certains scripts sont basés sur `pvpython` qui est une extension de `python` permettant de réaliser des scripts utilisant les fonctionnalités de `Paraview`. Les scripts `python` et `pvpython` doivent être lancés avec leurs commandes respectives.

- `DateDeLaPrevisionAromeHD` : script `python` prenant en paramètre un nombre d'heures à partir de l'heure actuelle et renvoie l'heure de la prévision précédente la plus proche
- `DateDuRunAromeHD` : script `python` renvoyant l'heure de la simulation la plus récente
- `RequeteAromeHD` : script `python` prenant en paramètre un nombre d'heures à partir de l'heure actuelle et le nom d'un package. Ce nom de package indique quelles données de simulation doivent être récupérées. Dans ce projet nous utiliserons le package SP1 correspondant au vent et à la température au sol. Le script télécharge les données et les stocke dans le répertoire DATA au format `grib2` si elles n'y sont pas déjà présentes.
- `VisuAvecTemperature` : script `pvpython` prenant en paramètre un fichier de simulation au format `nc` et générant une image dans le même dossier représentant la carte des couleurs de la température prévue. Ce script a été généré avec `Paraview` puis modifié afin de générer une image sans lancer `Paraview` (rendu *offscreen*).
- `MoteurContourConnecte` : fonctions permettant de calculer le contour connecté d'un champ de valeurs, par exemple la température.
- `ContourConnecteEnKML` : script `pvpython` utilisant `MoteurContourConnecte` et permettant de générer un fichier `KMZ` affichant les contours connectés d'une carte de température.

3.3 Scripts Bash

Deux scripts fonctionnels utilisant les scripts `python` précédemment décrits vous sont fournis :

- RUNME_IMAGE.sh génère un fichier `kmz` affichant une carte de couleur selon la température
- RUNME_CONTOUR.sh génère un fichier `kmz` affichant les courbes iso-valeur de la température

Les fichiers `kmz` générés sont des fichiers compressés contenant un fichier `kml` accompagné des images. Ces fichiers peuvent être ouverts directement dans `Google Earth` comme les fichiers `kml`.

Les deux scripts fournis prennent en paramètre le nombre d'heures de décalage dans le temps afin de calculer l'heure à laquelle la prévision doit être récupérée. Par exemple si le script est lancé à 13h30 avec la commande `./RUNME_IMAGE.sh 2`, la simulation prévoyant la météo à 15h sera téléchargée. Attention, les heures sont données en UTC, ce qui équivaut à une heure de moins par rapport à l'heure d'hiver française.

Les scripts appellent les fichiers `python` fournis, décrits dans la section précédente. Ceux-ci téléchargent les données de simulation depuis le site Meteo France via une requête HTTP. Les données récupérées au format `grib2` sont ensuite converties au format `nc` afin de pouvoir être importées dans `Paraview` grâce à la commande :

```
wgrib2 fichierGrib2 -match ":TMP:" -netcdf fichierNC
```

Toutes ces données sont stockées dans le répertoire DATA. Pour réduire le temps de conversion, par défaut uniquement les données de température sont conservées (option `-match ":TMP:"`, `TMP` étant l'indicatif pour la température). Pour obtenir toutes les données, vous devrez supprimer cette option ou la modifier. Un script `pvpython` est ensuite utilisé afin de créer la carte des couleurs à l'aide de `Paraview`. Un fichier `kml` est ensuite créé afin d'afficher cette carte de couleurs, et une archive `kmz` contenant tous les fichiers nécessaires est créée dans le dossier `kml`.

4 Rendu

Votre rendu sera composé d'un compte-rendu succinct décrivant les solutions choisies pour réaliser le projet, quelques captures d'écran de résultats ainsi que la description des difficultés rencontrées. Vous devrez également rendre les scripts réalisés ainsi que certains fichiers `KMZ` produits. Le pipeline rendu doit être fonctionnel.

5 Annexe

5.1 Environnement de travail

Pour ce projet, il est fortement recommandé d'utiliser un système Linux ou Mac OS pour pouvoir lancer les scripts bash. Vous devriez utiliser une version de `Paraview` récente, par exemple la version 5.4. `Paraview` doit être installé de sorte que `pypython`, la version `python` permettant d'exécuter des scripts `Paraview`, puisse être utilisée directement en ligne de commande. Par exemple pour Linux, il est possible de modifier le fichier .bashrc afin de paramétrer la variable d'environnement PATH.

L'utilitaire `wgrib2` doit également être installé et accessible depuis la ligne de commande. Les instructions d'installation sont disponibles à l'adresse suivante :

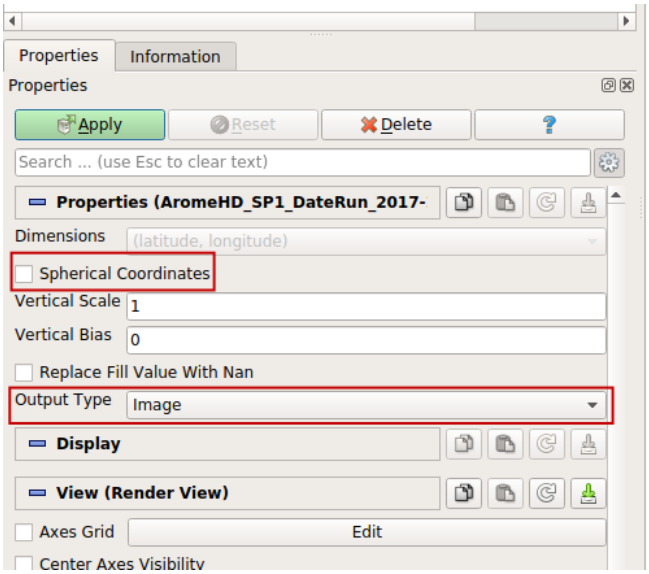
http://www.ftp.cpc.ncep.noaa.gov/wd51we/wgrib2/INSTALLING

L'utilisation de `Google Earth` en ligne pour `Google Chrome` est déconseillée, celle-ci ne gérant pas correctement les fichiers `kml`. Il est donc fortement recommandé d'installer `Google Earth` sur votre machine.

5.2 Filtres `Paraview` à utiliser

Les filtres suivant doivent être utilisés afin de générer une visualisation de température et de vent dans Paraview.

NetCDFReader permet d'importer un fichier au format `nc`. Accessible dans l'interface via le menu `File` puis `Open`, puis choisir `NetCDF files generic`. Décocher `SphericalCoordinates`, et choisir `image` dans le champ `Output Type`. Dans un script `pvpython`, prend en paramètre le nom du fichier à charger.



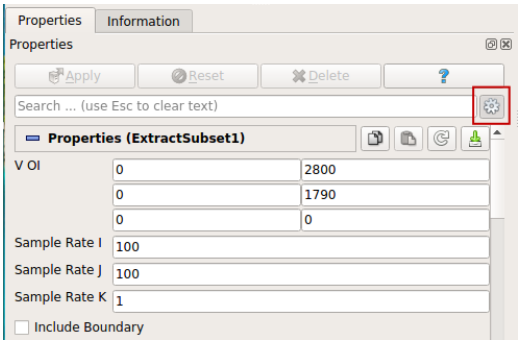
Calculator permet de calculer un nouvel ensemble de valeurs à partir d'un ensemble existant. La température étant exprimée en degrés Kelvin dans les données de simulation, ce filtre doit être utilisé pour la convertir en degrés Celcius. Il permet également de calculer les vecteurs du vent à partir de leurs composantes, en utilisant la formule suivante :

```
Vent = UGRD_10maboveground * iHat+VGRD_10maboveground * jHat
```

`UGRD_10maboveground` et `VGRD_10maboveground` étant des scalaires correspondant aux composantes des vecteurs vent.

Threshold permet de restreindre un ensemble de valeurs selon des seuils minimum et maximum. Utilisé pour conserver uniquement les données pertinentes dans les données de simulation. Les données non pertinentes étant représentées par des températures très grandes, celles-ci doivent être filtrées en utilisant une température maximum, par exemple de 100°C.

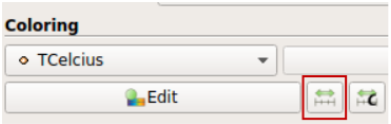
ExtractSubset permet d'extraire un échantillonnage régulier des données. Les paramètres `Sample Rate i` et `Sample Rate j` permettent de définir la fréquence d'échantillonnage des données. Le paramètre `Sample Rate k` n'est pas utilisé puisque nous utilisons des données en 2 dimensions. Ces paramètres sont accessibles en affichant les paramètres avancées (bouton en haut à droite du filtre).



Glyph permet d'afficher des flèches représentant un champ de vecteurs. Afin d'obtenir des flèches affichées sur une grille régulière, ce filtre doit être placé à la suite du filtre `ExtractSubset`. Les paramètres importants sont `Coloring` qui définit la couleur des flèches et `Scaling` leur taille.

Streamline permet de générer des lignes de courant à partir d'un champ de vecteurs. Le paramètre `Seed type` permet de choisir l'origine des lignes de courant pour l'algorithme, entre ligne et point.

Notes

- Les filtres peuvent être renommés avec un double clic sur leur nom.
- Utilisez `Save State` du menu `file` régulièrement pour sauvegarder votre pipeline `Paraview` et ainsi éviter de le perdre.
- Le bouton `RescaleToDataRange` de l'onglet `Display` des filtres (à côté du bouton `edit`) permet de mettre à jour l'échelle de couleurs lorsqu'un filtre est modifié. 

5.3 Export d'un état `Paraview` en `pvpython`

La fonction `Save state` de `Paraview` permet de sauvegarder le pipeline actuel sous forme de script `pvpython`. Pour cela, choisissez `Save state` puis modifiez `Paraview state file` en `Python state file`. Ce script doit ensuite être modifié afin de pouvoir générer une image directement, sans ouvrir `Paraview`. Vous pouvez vous inspirer du script fourni `VisuAvecTemperature` pour obtenir un script exécutable.

Afin de générer des images `Offscreen`, les lignes suivantes doivent être ajoutées :

```
renderView1.UseOffscreenRendering = True
renderView1.UseOffscreenRenderingForScreenshots = True
```

Vous pouvez changer les valeurs de `ViewSize` afin d'obtenir une image haute résolution :

```
MonEchelle = 0.5
renderView1.ViewSize = [(int) (2801*MonEchelle),(int) (1791*MonEchelle)]
```

Vous pouvez modifier le fichier initial de type `nc` par un autre fichier, ou prendre le nom de fichier en argument :

```
lecteurNC = NetCDFReader(FileName=[sys.argv[1]])
```

Vous devez également sauvegarder l'image à la fin du script :

```
WriteImage("image.png")
```