Documentation complémentaire sur les modifications réalisées pour la branche V2.2_clean

future version estampillée V3.3

Table des matières

Modifications réalisées

Ajout de la possibilité de sortir des données en niveaux hauteur au-dessus du sol

Besoins

Le besoin formulé était de sortir des champs interpolés en niveaux hauteur au-dessus du sol (via l'orographie).

Implémentation

Ajout d'une nouvelle spatial_shape ***XY-HG*** et gestion de la liste des niveaux demandés.

Utilisation

Via la Home Data Request pour les types dev, perso et extra et par les settings (sauf pour les variables de type extra)

Modifications à réaliser dans la Home Data Request par rapport à la syntaxe habituelle :

- label : ***label_var|label_ref*** avec ***label_var*** le label à utiliser pour la sortie et ***label_ref*** le label de la variable de référence
- spatial_shape : ***'XY-HG'***

Modifications à réaliser dans les settings (hors variables extra où l'axe doit être ajouté dans la table CNRM_coordinates.json) :

- Ajout des éléments concernant l'axe dans le simulation_settings, via la clé ***perso_sdims_description***

Exemple 1

Home Data Request :

```
perso;  zg50!zg;  atmos;  day;  CNRM_day;  time-mean;  XY-HG;  ANY;  ANY
```

Simulation_settings :

```
'perso_sdims_description':{
   'zg50':{
      'HG50':{
         'value':'50',
         'axis':"Z",
         'positive':'true',
         'out_name':'HG50'
      }
   }
}
```

Exemple 2

Home Data Request :

```
extra;  ua50m!ua;  atmos;  6hr;  CNRM_HOMZ6hr;  time-point;  XY-HG;  ANY;  ANY
```

Déclaration de ua50m dans la nouvelle table extra CNRM_HOMZ6hr.json :

```
"ua50m": {
            "frequency": "6hr",
```

```
            "modeling_realm": "atmos",
            "standard_name": "eastward_wind",
            "units": "m s-1",
            "cell_methods": "area: time: point",
            "cell_measures": "area: areacella",
            "long_name": "Eastward Wind at 50m",
            "comment": "Zonal wind at 50m height",
            "dimensions": "longitude latitude time height50m",
            "out_name": "ua50m",
            "type": "real",
            "positive": "",
            "valid_min": "",
            "valid_max": "",
            "ok_min_mean_abs": "",
            "ok_max_mean_abs": "",
            "cnrm_priority": "1"
        }
```


Déclaration la dimension height50m dans la table CNRM_coordinates.json :


```
"height50m": {
            "standard_name": "height",
            "units": "m",
            "axis": "Z",
            "long_name": "height",
            "climatology": "",
            "formula": "",
            "must_have_bounds": "no",
            "out_name": "height",
            "positive": "up",
            "requested": "",
            "requested_bounds": "",
            "stored_direction": "increasing",
            "tolerance": "",
            "type": "double",
            "valid_max": "120.0",
            "valid_min": "80.0",
            "value": "50.",
            "z_bounds_factors": "",
            "z_factors": "",
            "bounds_values": "",
            "generic_level_name": ""
        }
```


Perspectives

- Généralisation de la syntaxe sur les label à tout type d'entrée de la Home Data Request. L'idée étant de pouvoir définir simplement un nom de variable différent en sortie si besoin.
  - Normalement, la modification est déjà codée mais il faut vérifier que tout fonctionne.

Modification de la structure du package dr2xml pour la rendre compatible avec son packaging

Besoins

Simplification de la structure du package pour :

1. pouvoir mettre en place un packaging permettant de l'installer plus facilement
2. pouvoir appeler une version de dr2xml en particulier en changeant son chemin d'appel

Implémentation

Modification de la structure pour formaliser le package dr2xml (comme cela avait déjà été le cas pour xml_writer).

Perspectives

- Mise en place du packaging pour faciliter l'installation de dr2xml (actuellement, seule l'installation via git est disponible)

Ajout d'une interface pour l'adaptation aux projets autres que CMIP6 (définition des méta-données des fichiers de sortie)

Besoins

L'exercice CMIP6 est terminé et, pour pérenniser et continuer à exploiter pleinement l'outil dr2xml, il est nécessaire de l'adapter pour des simulations de développement et pour une utilisation dans le cadre d'autres projets (par exemple CORDEX).
Un premier jeu de modifications avait été réalisé pour produire des fichiers compatibles avec le format CORDEX (à quelques détails près) mais ce n'était pas généralisable à d'autres projets (conditions en dur dans le code).
L'objectif de ce nouveau jeu de modifications est de proposer une interface permettant d'adapter le format des données produites aux spécifications propres à des projets variés. Pour commencer, on s'intéresse aux utilisations pour le régional et la prévision saisonnière. Une fois les modifications validées, d'autres utilisations pourront être testées.

Implémentation

Ajout d'un répertoire projects contenant :

- ***projects_interface_definitions.py*** : définition des différents objets permettant de construire les fichiers de configuration relatifs aux différents projets.
- dr2xml.py : fichier contenant les éléments minimaux nécessaires pour faire tourner dr2xml
- un fichier par projet (au départ, basics.py, CMIP6.py, CORDEX.py, d'autres seront ajoutés ensuite)

Ajout d'une interface ***json_project_interface.py*** permettant de lire et d'interpréter le fichier de configuration du projet.

Modification des différents fichiers source de dr2xml pour prendre en compte les évolutions et, en particulier, supprimer du code les références en dur à des noms d'attributs projet-dépendants.

Utilisation

Ajout de clés supplémentaires dans le lab_and_model_settings :

- ***project_settings*** :
  Clé indiquant le projet de référence pour la configuration des fichiers de sortie (chaîne de caractère contenant le nom du projet s'il est connu (I.E. intégré à la base de projets de dr2xml) ou un chemin vers un fichier contenant le fichier python configurant le projet)
  Défaut : valeur de la clé ***project***
- ***project*** :
  Clé indiquant le projet auquel est rattaché le settings
  Défaut : ***'CMIP6'***
- ***save_project_settings*** :
  Une fois le dictionnaire python définissant entièrement le projet déterminé (et avant interprétation des paramètres venant des dictionnaires et settings), si cette clé ne vaut pas None, écriture à l'emplacement indiqué du contenu du dictionnaire.
  Défaut : ***None***
  NB : cette option n'est pour l'instant pas implémentée

Contenu des fichiers python spécifiques aux projets :

- Les références vers les classes de ***projects_interface_definitions*** utilisées
- Les clés suivantes :

- ***parent_project_settings*** :
  Nom du fichier parent, s'il y en a un.
  Défaut : ***None***
- ***internal_values*** :
  Dictionnaire contenant les valeurs des paramètres nécessaires pour faire tourner dr2xml. Si ces valeurs ont besoin d'être redéfinies, cela doit impérativement être fait dans ce dictionnaire.
  Défaut : ***dict()***
- ***common_values*** :
  Dictionnaire contenant les valeurs des paramètres nécessaires communes à plusieurs paramètres dans les projets. Cela permet de déterminer ces valeurs une fois pour toute au début du lancement de dr2xml.
  Défaut : ***dict()***
- ***project_settings*** :
  Dictionnaire contenant la définition des différents tags utilisés.
  Défaut : ***dict()***

- Les fonctions où les références aux fonctions appelées lors de la détermination des attributs et variables des fichiers.

Description des types d'objet disponibles pour construire les fichiers de configuration des projets :

- ***ValueSettings*** :
  - Objet représentant la provenance et la valeur d'un paramètre.
  - Attributs :
    - ***key_type*** (défaut: ***None***) : type de clé parmi ***combine***, ***common***, ***internal***, ***dict***, ***config***, ***laboratory***, ***simulation***, ***json***, ***DR_version***, ***scope***, ***variable***
    - ***keys*** (défaut : ***list()***) : liste des sous-clés (possibilité d'utiliser des ***ValueSettings***)
    - ***fmt*** (défaut : ***None***) : format à adopter pour la sortie (appel à la méthode format des chaînes de caractères avec la valeur obtenue)
    - ***src*** (défaut : ***None***) : source (cas de lecture d'un fichier json) sous la forme d'un ***ValueSettings*** ou du chemin vers un fichier
    - ***func*** (défaut : ***None***) : fonction à appeler (fonction python ou objet ***FunctionSettings***) pour créer la valeur si ***key_type*** ou ***keys*** sont vides ou pour la modifier
  - Utilisation :
    - si ***key_type*** vaut ***laboratory***, on cherche dans le ***lab_and_model_settings***
    - si ***key_type*** vaut ***simulation***, on cherche dans le ***simulation_settings***
    - si key_type vaut config, on cherche dans le fichier de configuration
    - si ***key_type*** vaut ***dict***, on cherche dans le dictionnaire passé lors de la création
    - si ***key_type*** vaut ***variable***, on cherche dans les attributs de l'entrée variable du dictionnaire passé lors de la création
    - si ***key_type*** vaut ***DR_version***, on récupère la version de la Data Request uutilisée
    - si ***key_type*** vaut ***scope***, on récupère l'objet ***scope*** associé à la Data Request
    - si ***key_type*** vaut ***internal***, on cherche dans ***internal_values***
    - si ***key_type*** vaut ***common***, on cherche dans ***common_values***
    - si ***key_type*** vaut ***combine***, on vérifie que toutes les valeurs de ***keys*** sont définies et on les combine via ***fmt***
    - si ***key_type*** vaut ***json***, on lit le fichier défini par ***src*** et on récupère son contenu
- ***ParameterSettings*** :
  - Objet représentant les caractéristiques d'un paramètre.
  - Attributs :
    - ***skip_values*** (défaut : l***ist()***) : liste des paramètres (possibilité de ***ValuesSettings***) non valides

        - ***forbidden_patterns*** (défaut : ***list()***) : liste des patterns au sens python non valides
        - ***conditions*** (défaut : ***list()***) : liste des conditions à vérifier pour que le paramètre soit ajouté (booléen ou ***ConditionSettings***)
        - ***default_values*** (défaut : ***list()***) : liste des valeurs par défaut (possibilité de ***ValueSettings***)
        - ***cases*** (défaut : ***list()***) : liste des cas à vérifier successivement (objets de type ***CaseSettings***)
        - ***authorized_values*** (défaut : ***list()***) : liste des valeurs autorisées (possibilité de ***ValueSettings*** renvoyant une unique valeur) ou objet ***ValueSettings*** renvoyant une liste de valeurs
        - ***authorized_types*** (défaut : ***list()***) : liste des types (au sens python) autorisés
        - ***corrections*** (défaut : ***dict()***) : dictionnaire donnant pour chaque valeur la nouvelles valeur à utiliser
        - ***output_key*** (défaut : ***None***) : dans le cas où le paramètre ne doit pas être ajouté avec la valeur de l'entrée, le nom à utiliser en sortie
        - ***num_type*** (défaut : ***string***) : type d'objet au sens netcdf (***int***, ***string***)
        - ***is_default*** (défaut : ***False***) : dans le cas où une valeur par défaut existe, cette clé vaut ***True*** (mis à jour automatiquement si non initialisée)
        - ***fatal*** (défaut : ***False***) : indique si la clé doit absolument être présente ou si elle est facultative
- ***TagSettings*** :
    - Objet représentant les caractéristiques d'un tag.
    - Attributs :
        - ***attrs_list*** (défaut : ***list()***) : liste des noms des attributs du tag (chaînes de caractères)
        - ***attrs_constraints*** (défaut : ***dict()***) : dictionnaire contenant pour chaque attribut du tag un objet de type ***ParameterSettings*** contenant ses caractéristiques
        - ***vars_list*** (défaut : ***list()***) : liste des noms des variables du tag (chaînes de caractères)
        - ***vars_constraints*** (défaut : ***dict()***) : dictionnaire contenant pour chaque variable du tag un objet de type ***ParameterSettings*** contenant ses caractéristiques
        - ***comments_list*** (défaut : ***list()***) : liste des noms des commentaires du tag (chaînes de caractères)
        - ***comments_constraints*** (défaut : ***dict()***) : dictionnaire contenant pour chaque commentaire du tag un objet de type ***ParameterSettings*** contenant ses caractéristiques
- ***FunctionSettings*** :
    - Objet représentant les caractéristiques d'une fonction.
    - Attributs :
        - ***func*** : fonction devant être appliquée
        - ***options*** (défaut : ***dict()***) : dictionnaire avec les options à passer à la fonction (en plus de la valeur éventuelle)
- ***ConditionSettings*** :
    - Objet représentant les caractéristiques d'une condition.
    - Attributs :
        - ***check_value*** : valeur à tester (peut être un objet de type ***ValueSettings***)
        - ***check_to_do*** : opération à réaliser parmi :
            - ***eq*** : vérifie que ***check_value*** se trouve dans ***reference_values***
            - ***neq*** : vérifie que ***check_value*** ne se trouve pas dans ***reference_values***
            - ***match*** : vérifie que ***check_value*** match au moins une des valeurs de ***reference_values***
            - ***nmatch*** : vérifie que ***check_value*** ne match aucune des valeurs de ***reference_values***

        - ***reference_values*** : valeur ou liste de valeurs (peuvent être du type ***ValueSettings***) de référence pour le test.
    - Cas particulier :
        - Dans le cas où ***reference_values*** est une liste vide et où ***check_to_do*** vaut ***eq***, on vérifie que ***check_value*** n'est pas définie.
- ***CaseSettings*** :
    - Objet représentant les caractéristiques d'un cas.
    - Attributs :
        - ***conditions***: liste ou valeur isolée indiquant la condition à vérifier parmi des :
            - booléens
            - ***ConditionSettings***
        - ***value*** : la valeur à prendre, si elle existe, si la condition est valide (peut être un objet de type ***ValueSettings***)

Format des dictionnaires :

- ***internal_values*** et ***common_values*** : chaque clé du dictionnaire est le nom d'un paramètre et sa valeur un objet ***ParameterSettings***.
- ***project_settings*** : chaque clé du dictionnaire est le nom d'un tag et sa valeur un objet ***TagSettings***.

Exemple

Voir les exemples dans la branche V2.2_clean_test_example.

Limitations

Modifications compatibles uniquement avec python 3 (perte de la compatibilité entre python 2 et python 3).

Perspectives

Terminer les modifications associées afin d'éliminer complètement du code les spécificités CMIP6.

Ajout d'une interface permettant de se passer de la Data Request CMIP6

Besoins

L'exercice CMIP6 est terminé et dr2xml commence à être utilisé pour des simulations de développement et dans le cadre d'autres projets (par exemple CORDEX).
L'objectif de ce nouveau jeu de modifications est de proposer une interface permettant de s'affranchir complètement de la Data Request CMIP6 (pas de Data Request ou une Data Request alternative) tout en gardant le mécanisme de la Home Data Request.

Implémentation

Ajout d'une nouvelle clé, ***data_request_used***, dans le ***lab_and_model_settings*** pour indiquer la data request à utiliser (défaut : ***CMIP6***).

Ajout d'un module ***dr_interface*** contenant :

- ***__init__.py*** : interface générale vers les data request, renvoie l'une de celles disponibles
- ***definitions.py*** : définitions d'objets, dont ***Scope***, pour avoir une interface générique
- ***CMIP6.py*** : interface version la data request CMIP6, corrections éventuelles à appliquer aux valeurs de la data request renvoyées.
- ***no.py*** : interface vers une data request vide

Modification des fichiers source de dr2xml pour s'adapter à l'interface et éviter les corrections spécifiques en-dehors de cette interface.

Utilisation

Choix de la Data Request à utiliser via la variable ***data_request_used*** du ***lab_and_model_settings***.

Attention, dans le cas où on n'utilise pas la Data Request CMIP6, il faut définir les coordonnées inconnues dans le fichier ***coordinates.json.***

Exemple

Exemple ***AAD50_641_nodr*** dans la branche ***V2.2_clean_test_example***.

Perspectives

- Déplacement des dernières scories liées à la Data Request CMIP6 encore présent dans le code source vers l'interface pour finaliser la mise en place de l'interface.

Modification du comportement lié au paramètre max_split_freq

Besoins

Certains utilisateurs ont besoin de pouvoir donner une borne maximale à la fréquence de coupure, y compris lorsque celle-ci n'est pas exprimée en année.

Implémentation

Modification du comportement lié à la variable ***max_split_freq*** qui esst utilisé dès qu'elle est définie.

Utilisation

Définition de la variable ***max_split_freq***.

Ajout de la possibilité de sortir plusieurs variables dans le même fichier

Besoins

Sur certains systèmes de fichiers, la multiplication des petits fichiers pose problème.
Pour CMIP6, il fallait produire des fichiers par variable et cela augmente donc le nombre de petits fichiers.
Afin de résoudre ce problème, la possibilité de sortir plusieurs variables dans le même fichier a été ajoutée.

Implémentation

Modification du fichier ***Xwrite.py*** et ajout du paramètre ***grouped_vars_per_file*** (simulation_settings et lab_settings).
Toutes les variables ne peuvent pas être regroupées, il faut que les paramètres suivants soient identiques pour chacun d'entre elles (dans l'ordre) :

- ***frequency***
- ***table***
- ***grid_label***
- ***split_freq***
- ***split_freq_format***
- ***split_last_date***
- ***split_start_offset***
- ***split_end_offset***
- ***grid_description***
- ***grid_resolution***
- ***target_hgrid_id***
- ***zgrid_id***
- ***source_grid***

Si des variables ne sont pas dans cette liste, elles sont laissées dans des fichiers séparés.

Utilisation

Ajout de la variable grouped_vars_per_file dans le simulation_settings ou le lab_settings. Cette variable doit contenir une liste de liste de noms de variables (label).

Exemple

Exemple de valeur pour le paramètre :
[[« ta », « ua », « va », « zg »], [« tas », « uas », « vas », « pr », « pc », « psl »]]

Perspectives

- Si besoin, assouplir les conditions pour pouvoir avoir des variables dans les mêmes fichiers.