==========

Cette doc vous donne les clés en main pour travailler avec LMDZ_Setup (avant 2023-07 : “tutorial_prod”)

Vous êtes les bienvenus à l’améliorer et l’enrichir, ainsi que les scripts et le README du LMDZ_Setup. Les futurs utilisateurs vous en remercieront !

Pour consulter les rubriques plus facilement, activez dans le menu “Affichage” l’option “Afficher le plan du document”

==========

Extraire LMDZ_Setup

au choix : rép. sous svn, ou archive *.tar :

a/ Recommandé : extraire directement le rép. LMDZ_Setup (ancien TEST_PROD)**:**

svn co https://svn.lmd.jussieu.fr/LMDZ/BOL/LMDZ_Setup

b/ Autre possibilité : **télécharger le tar :**

wget https://lmdz.lmd.jussieu.fr/pub/Training/LMDZ_Setup.tar

c/ Pour les habitués de longue date : un lien vers LMDZ_Setup.tar, avec l’ancien nom tutorial_prod.tar, reste disponible :

wget https://lmdz.lmd.jussieu.fr/pub/Training/tutorial_prod.tar

LMDZ_Setup se trouve sur le compte “lmdz” ici :
/u/lmdz/WWW/BOL/LMDZ_Setup et LMDZ_Setup.tar
/u/lmdz/pub/Training/LMDZ_Setup et .tar = liens vers /u/lmdz/WWW/BOL/

Cette doc, LMDZ_Setup_HowTo :
https://docs.google.com/document/d/1OLZG6e-86NiXuv5-aALxKIh-QPkp4BdCwWtiBFot-6c

==========

Rubriques

0/ La dernière version testée

1/ Principes, avertissements, choses à faire, ou à ne pas faire

2/ FAQ (avec les réponses, oui, c'est l'intérêt !)

3/ Questions en attente de réponse (à vos claviers…)

4/ Suggestions (par composante : ORCHIDEE, aérosols, XIOS, COSP etc)

5/ Problèmes connus (certains seront sûrement traités un jour :) **)**

6/ Problèmes inconnus

==========

0a / Commissions "critiques" depuis la mise sous svn

2024-06 : Explication et solution "finale" du plantage "srun: error: "Unable to create step for job nnnn : More processors requested than permitted"

il arrive seulement quand on passe par l'étape d'initialisation :

Dans tmp_SIM (script_SIMU) on a

```
#SBATCH --cpus-per-task=8 # nb de threads OpenMP = $omp du setup.sh
```

- quand init=0, tmp_SIM est soumis directement par setup.sh, et c'est son "cpus-per-task" qui est pris en compte ;
- mais quand init=1, tmp_SIM est soumis par "tmp", et "herite" du "--cpus-per-task=16" du header de "tmp" (en dur dans setup.sh).

Alors que dans "tmp" on a par la suite, logiquement pour ce0l : export OMP_NUM_THREADS=1

Explication (Idris) : un changement de version du gestionnaire de tâches "Slurm" début 2024 a des effets "non-souhaitées" (héritage de cpus-per-tasks" sur le lancement de "srun" dans "sbatch". Un retour au comportement "ancien" est prévu dans qques semaines.

Solution adoptée :

dans "tmp", pour le cas 'init=1', faire "oublier" le " --cpus-per-task" avant de soumettre tmp_SIMU :

```
bash -c 'unset \$(env | egrep "SLURM_|SBATCH_|SRUN_"| cut -d= -f1) ;
$submit tmp_$SIM'
```

==========

0b / Versions d'avant la mise sous svn

2023-07-07
DERNIERE VERSION AVANT CHANGEMENT DE NOM en LMDZ_Setup
et MISE SOUS SVN
est téléchargeable ICI :

https://lmdz.lmd.jussieu.fr/pub/Training/tutorial_prod.tar

ATTENTION pour main.sh : vous êtes fortement conseillés d'utiliser la version de LMDZ par défaut (dernière testing, et svn=""), ou de consulter l'équipe LMDZ(email @poihl, ou mattermost) si vous avez vraiment besoin d'une autre version/rev.
En principe, on peut utiliser un modipsl "unstable" avec
version="YYYYMMDD.trunk -unstable",
ou on peut changer uniquement la rev LMDZ après installation de modipsl*.tar (testing ou pas) avec svn="-r NNNN"

—----------

2023-07-07 : **Dernier tutorial_prod.tar** , avant passage au svn
https://lmdz.lmd.jussieu.fr/pub/Training/tutorial_prod.tar/tutorial_prod.tar_v20230707

- fichiers DEF/iso* : renommés en tant que DEF/iso*._iso pour qu'ils soient copiés dans SIM/DEF seulement si on tourne avec isotopes
- nettoyage dans les DEF/*def

2023-07-05:
tutorial_prod.tar -> Archive/tutorial_prod.tar_v20230705

- on impose dans main.sh que le lancement de boucles sur plus de 1 élément dans la liste $physics soit possible seulement une fois les start* et limit* disponibles dans le rép de travail, donc :
soit avec init=0 , après avoir fait tourner une fois la phase d'initialisation avec init=1 ;
soit avec init=SIM0 (une simu - spinup par ex - qui contient les restarts nécessaires)

A faire rapidement:

- Déplacer le changement automatique de flags : qu'il se fasse dans SIM/DEF non pas dans DEF d'origine

==========

VERSIONS ANCIENNES

stockées depuis dec 2022 dans :

https://web.lmd.jussieu.fr/~lmdz/pub/Training/Archive

2023-06-29

https://lmdz.lmd.jussieu.fr/pub/Training/Archive/tutorial_prod.tar_v20230629OK

avec : Nouvelle testing : rev LMDZ 4586 du 2023-06-26

physics="NPv6.3" par défaut en remplacement de la 6.1 (toujours disponible)

autres :

dans setup.sh, un “patch” est rajouté pour le cas “Jean-Zay” & “mysvn >= 4532” & “netcdf95 n’est pas déjà dans lmdz_env”

- on copie lmdz_env.sh original en tant que “lmdz_env.orig”
- on introduit dans lmdz_env.sh la ligne p” et
- on recopie lmdz_env.sh en tant que LMDZ/arch/arch-X64_Jean_Zay)
- plusieurs petites corrections **suite à un “set -e” malencontreux (ou pas…)**

2023-06-14

https://lmdz.lmd.jussieu.fr/pub/Training/Archive/tutorial_prod.tar_v20230614ORtrunk

Ajout de l'option "veget=7994", pour ORCHIDEE trunk correspondant à la config libIGCM LMDZOR_6.4work
Le routage (méthode = simple) est désactivé par défaut ; il peut être activé comme décrit dans la section "ORCHIDEE-routage". N’a pas été testé avec zoom.

2023-05-23

https://lmdz.lmd.jussieu.fr/pub/Training/Archive/tutorial_prod.tar_v20230523

Optimisation de l’option “isotopes” (avec S. Nguyen):

1/ ce seul choix dans main.sh fait tout le reste pour les isotopes : lmd_phys="lmdiso"

2/ dans setup.sh, une variable locale "isotopes=n" par défaut , passe à "y" si phylmd = "lmdiso" ("phylmd" étant "lmd_phys" reçu du main.sh). Elle est utilisée pour tous les changements automatiques "if isotopes=y"... et est transmise à script_SIMU (dans le "sed" général).

3/ dans script_SIMU, "isotopes" sert (seulement) à changer un flag dans iso.def à partir de la 2ème période de run, pour lire les isotopes dans les restarts.

4/ DEF/PHYS/physiq.def_NPiso est créé à base de NPv6.3, avec seule différence iflag_ice_thermo=0 Dans setup.sh, un "sed" assure que le flag est mis à zéro pour "isotopes=y" s'il ne l'est pas déjà.

2023-05-05 : tutorial_prod.tar_v20230505iso
avec options par défaut :

- LMDZ nouvelle testing du 2023-04-12 (modipsl20230412.tar ; rev 4507)
- physiq.def_NPv6.3
- ORCHIDEE-CMIP6
- rrtm
- sans XIOS

Changement majeur : **introduction de l'option de tourner avec isotopes (S. Nguyen)**

Autres (mineur) : "rangement" dans setup.sh (notamment la section "Avoid combinations …"

2023-04-25 : https://lmdz.lmd.jussieu.fr/pub/Training/Archive/tutorial_prod.tar_v20230412

Changements principaux par rapport à la version précédente :

- MAJ de "version" LMDZ : nouvelle testing : rev 4507, modipsl.20230412.tar.gz
- "veget" : rajout de l'option rev **7983** sur la branche ORCHIDEE_2_2 (**XIOS exigé** !)
  Différences "mineures"de *def et *xml par rapport à la rev 7597 :
  - dans DEF : orchidee.def_6.2work est identique à orchidee.def_6.1, sauf DOWNREGULATION_CO2**_NEW**=y au lieu de DOWNREGULATION_CO2=y
  - dans DEF/XMLfilesOR*, field_orchidee et file_ ont des variables en plus

Changements mineurs:
- dans **main.sh** : les utilisateurs sont fortement conseillés d'utiliser la version par défaut (derniere testing, et svn=""), ou de consulter LMDZ Team (email à poihl, ou mattermost) s'ils ont vraiment besoin d'une autre version/rev. En principe, on peut utiliser un modipsl "unstable" avec version="YYYYMMDD.trunk -unstable", ou on peut changer uniquement la rev LMDZ après installation de modipsl*.tar (testing ou pas) avec svn="-r NNNN"

- dans **setup.sh** :
  - quand on lance main.sh avec installation du modèle, un fichier **install_lmdz_options.$$** est créé dans $STORE/MAINDIR avec les options de lancement de install_lmdz.sh (pour pouvoir lancer directement install_lmdz.sh sur $WORKDIR si besoin)
  - "gardefou" pour utiliser l'option XIOS avec Orchidee 2_2
    (Note : garder output_level="1" pour histmth si on l'active, sinon ça plante - voir section 5-Problèmes connus)

2023-03-29 :
https://lmdz.lmd.jussieu.fr/pub/Training/Archive/tutorial_prod.tar_v20230329
améliorations et tests avec LMDZ rev 4473 (ou 4476) pour testing 202303

NOTE : à partir de là, modipsl*.tar inclut IOIPSLv_2_2_5 au lieu de v2_2_2
install_lmdz.sh a été adapté au changement (rev 4508 ; asima) : compilation des outils "rebuild" dans rép. "IOIPSL/rebuild" au lieu de "IOIPSL/tools"

Changements:

-> physiq.def_NPv6.2, NPv6.2.0 et physiq.def_NPv6.3 rajoutés dans DEF/PHYS ; physiq.def_NPv6.3 copié en tant que DEF/physiq.def au lieu de NPv6.1

main.sh : option “physics=NPv6.1” par defaut, changée en physics="NPv6.3"

-> si veg=CMIP6 et XIOS, alors on force omp=2 dans setup.sh ; si omp=4 ou 8 (defaut), alors plantage à cause de Orchidee-CMIP6 ; AVEC DES BONNES EXPLICATIONS pour savoir pourquoi on a fait ça (*NB Laurent avait regardé le* lancement du gcm en MPIxOMP, et séparément de XIOS sur MPI seul, car OMP “pas fiable avec les versions de XIOS qu’on utilise” ; typiquement, en OMP=8 (defaut du tutorial_prod) le job restait “accroché” pour la combinaison XIOS-ORCMIP6 ; erreur probale dans xios_orchidee.f90, appelé par intersurf.f90, lui-même appelé par surf_land_orchidee_mod.F90

-> modif proposée par Sébastien Nguyen dans script_SIMU pour rendre le remplacement de ***$ndayh'day*** pour histmth dans config.def indépendant du nombre d’espaces entre celui-ci et ***“1day”*** pour histday:
sed -e 's/phys_out_filetimesteps=.*day* 1day/phys_out_filetimesteps= '$ndayh'day 1day/' DEF/config.def >| config.def*
est remplacé par:
*sed -e 's/phys_out_filetimesteps=**[[:space:]]*[0-9][0-9]day**/phys_out_filetimesteps= '$ndayh'day/' DEF/config.def >| config.def*

==========

2022-07-06 :

NOTE : qques corrections depuis, dans le même Archive/tutorial_prod_ecrad.tar vers lequel pointe tutorial_tar :

- màj de l’adresse de téléchargement : web* remplacé par lmdz.lmdz …
- XMLfilesLMDZ/iodef.xml : une ligne “context_orchidee” avait été introduite en juillet 2022, mais ce iodef est pour LMDZ seul (un iodef.xml pour LMDZ-OR est stocké dans XMLfilesOR***, et écrase celui pour LMDZ seul le cas échéant)

Une version tutorial prod qui tourne avec RRTM et ECrad et la version des sources du 8 juillet à télécharger ici :
wget --no-check-certificate -nv **https://lmdz.lmd.jussieu.fr/pub/Training/tutorial_prod.tar**

(qui est un lien vers :
https://lmdz.lmd.jussieu.fr/pub/Training/Archive/tutorial_prod_ecrad.tar)

NOTE : a/ (seul bémol): nbapp_rad à changer de 16 à 24 pour que ça marche avec ecrad. Pour l'instant il faut le faire à la main.

(b) Il faut aussi mettre aerosols=n mais il y a un garde fou qui vous préviendra.

Changements :

FH : **introduction de ECRAD** *(avec aide au debug et tests par AI et et MC)*

AS sur suggestions de FH :

- **options de compilation communes gcm et ce0l** ; concernant "-debug" cela implique une modif de flincom.f90 dans IOIPSL(tag2_2_2)/src distribué avec le modipsl.tar. Cette correction (co-production lguez- asima - jghattas) avait été commit-ée seulement dans la trunk et la 2_2_5 de IOIPSL.
  MAJ 2023-03 : Laurent a changé la version de IOIPSL dans les modipsl*.tar :
  IOIPSL tag2_2_2 → IOIPSL tag2_2_5
- **fichier unique compile.sh** (applicable à gcm et ce0l) créé **dans rép de soumission $STORE/$MAINDIR**, au lieu de $WORK/ …/modipsl/modeles/LMDZ
- **AUSSI : le 19 dec 2022 : changement des liens vers "pub"** pour avoir la main sur l'actualisation des certificats (en lien avec les soucis de "wget" qui nécessitent l'option "----no-check-certificate" le temps d'actualiser les certificats) :
  http://www.lmd.jussieu.fr/~lmdz/pub
  devient
  https://lmdz.lmd.jussieu.fr/pub

2022-07-06

- https://web.lmd.jussieu.fr/~lmdz/pub/Training/tutorial_prod.tar
  pointe maintenant vers tutorial_prod.tar_v20220705,
  rrtm=y/n a été remplacé par rad=oldrad/rrtm/ecrad
  ecrad ne fonctionne pas encore, il faut rajouter "namelist" et "data"

2022-05-23

- à telecharger ici : https://web.lmd.jussieu.fr/~lmdz/pub/Training/tutorial_prod.tar
- ou à récupérer ici :
  /gpfsstore/rech/gzi/rdzt896/TEST_PROD_LATEST/tutorial_prod.tar
  Les **changements** par rapport à la version précédente **"**20220106" sont **:**
- **par défaut : climato=1 et veget=CMIP6**
- **rev 4163 de LMDZ => on peut tourner de nouveau avec SPLA (-dust true) ;**
- ajout de la **rev 7597** de **Orchidee branche 2_2** (indiquée par Josefine le 19 mai 2022 ; en gardant aussi la rev 7330 pour l'instant) ; demande de Étienne et Khadija pour développements "irrigations" dans le module "routage simple" de Yann Meurdesoif
- avec Valentin Wiener : nouvelle **option "testmode" dans setup.sh** pour lancer automatiquement avec certaines options mentionnées dans README1_TestMode (ancien README1_DebugMode)
- era2gcm.sh : renommé en **era2gcm_tuto.sh**, info rajoutée sur scripts "officiels" dans svn/TOOLS de l'IPSL ; correction unpack pour fichiers "ta" de ERAI signalée par Maëlle

2022-05-23

La version améliorée cf vos suggestions, remarques, plantages etc., se trouve :
→ sur le compte "gzi":
/gpfsstore/rech/gzi/rdzt896/TEST_PROD_LATEST/tutorial_prod_v20220106.tar
→ sur le compte "lmd" :
/gpfsstore/rech/**lmd**/rdzt896/tutorial_prod_v20220106.tar

A noter les améliorations suivantes dans setup.sh :

→ Si on lance les scripts depuis **jean-zay-pp** à la place de jean-zay, soit par erreur, soit parce qu'on veut compiler plus vite : un **'exit'** est prévu dans setup.sh, avec l'explication qu'il faut se connecter sur jean-zay pour pouvoir lancer une simulation

→ Pour t**ests courts** (typiquement pour debug), dans setup.sh on peut decommenter cette ligne (et aussi la ligne "nday=1" dans script_SIMU):
#For 1-day test runs in debug mode
if ["$optim" = "-debug"] ; then **cput=00:10:00** ; fi

→ Pour compiler avec COSP (v1 ; les suivantes sont en dévéloppement par A. Idelkadi) :
Dans setup.sh il faut modifier opt_cosp dans le "if" comme ci-dessous (j'ai aussi rajouté 2 commentaires):
if [$cosp = y] ; then
  opt_cosp="-cosp true" ; ins_cosp="-cosp v1"
….

==========

1/ Principes, avertissements, choses à (ne pas) faire

==========

Lisez attentivement les README* et les commentaires dans les scripts ->
cela peut vous éviter des galères !

→ Principes de fonctionnement

---

installation et lancement :

NOTE juin 2024 : cf nouvelles règles de l'Idris pour bientôt : les scripts devront être installés impérativement sur $WORK au lieu de $STORE ;
le changement demande simplement de **redefinir dans lmdz_env.sh:**
STORED=$WORK (au lieu de STORED=$STORE)
Cela sera mis par défaut dans LMDZ_Setup une fois la nouvelle règle entrée en vigueur.

Par ailleurs, toujours dans lmdz_env.sh:
- pour **faire tourner, et ranger, vos simulations dans un sous-répertoire de $WORK,** par ex $WORK/Simulations_d, il suffit de mettre **$STORED=$WORK/Simulations_d**
- pour installer le modèle dans un sous-répertoire de $WORK, par ex $WORK/LMDZ_d, il suffit de changer LMDZD=$WORK en LMDZD=$WORK/LMDZ_d
Fin note juin 2024—----------

Les scripts doivent être installés sur $STORE (et sont donc sauvegardés avec les sorties) ; toute la mise en place des simulations se fait là-bas **comme suit :**

- **sur $STORE : on télécharge et désarchive tutorial_prod.tar**
  cd $STORE
  wget https://lmdz.lmd.jussieu.fr/pub/Training/tutorial_prod.tar
  tar -xvf tutorial_prod.tar
- **on copie ou renomme le rép résultant TEST_PROD -> par ex TEST_PROD_1**
- **dans TEST_PROD_1 : on choisit les options dans :**
  - **main.sh, et**
  - **la section "EXPERT-LEVEL CHOICES" de setup.sh**
- **dans TEST_PROD_1/DEF :**
  - **on peut changer le config.def** par une autre version disponible **;**
  - **on peut (même on doit!) régler les outputs :** dans config.def si l'on est avec IOIPSL, ou dans TEST_PROD_1/DEF/XIOSfiles* si on utilise XIOS
- **puis on lance main.sh :** le fonctionnement est un peu différent si on lance sur jean-zay ou jean-zay-pp, cf ci-dessous :

jean-zay vs jean-zay-pp :

→ **sur jean-zay** la compilation prend beaucoup de temps (plus de 1h ; si on choisit de tourner avec XIOS, lui seul prend 20 min).
→ **sur jean-zay-pp** la compilation est beaucoup plus rapide.

- **le modèle s'installe sur $WORK,** on va y regarder si on veut récompiler "à la main" (sans utiliser les scripts du tutorial_prod)

- **les simulations tournent sur $SCRATCH ;** à chaque lancement d'une simulation via main.sh, un nouveau répertoire est créé sur **$SCRATCH**, avec la même arborescence que sur $STORE, avec le nom de la simulation suivi d'un numéro (chaque fois différent) à la fin. **Conséquence : pour relancer une simulation qui a planté, il n'y a pas besoin de nettoyer dans $SCRATCH** (contrairement au libIGCM).

- **En cas de plantage, l'info se trouve sur $SCRATCH** (contrairement au libIGCM, où l'info est remontée dans le répertoire de simulation). Voir ci-dessous FAQ "Où rechercher la cause d'un plantage".

ATTENTION : si c'est la première fois que vous installez ORCHIDEE, dans une version/revision différente de "CMIP6" (autrement dit, si vous allez utiliser l'option **veget=7330 ou 7597 dans setup.sh) : demandez à orchidee-help@listes.*ipsl*.fr le mot de passe** pour accéder au dépôt svn de Orchidee sur "forge" (sinon, rien ne va se passer comme prévu… !)

→ I/O : IOIPSL vs XIOS

Une seule option gère l'utilisation de XIOS vs IOIPSL pour LMDZ et Orchidee.

XIOS est exigé pour tourner avec certaines versions/revisions de Orchidee post-CMIP, dont la seule disponible avec tutorial_prod c'est la rev 7266 sur la branche 2_2 sur laquelle se font les développements de la hydrologie, notamment par Agnès Ducharne et son équipe.

IOIPSL est utilisable pour tourner LMDZ "seul" (sans Orchidee, mais avec le modèle simple d' hydrologie "bucket"), ou avec Orchidee jusqu'à sa version CMIP6 inclue.

ATTENTION quand vous utilisez XIOS :

tous les fichiers file*.xml disponibles pour chaque modèle/version sont présents dans les répertoires DEF/XMLfiles*, mais seulement ceux utilisés habituellement (histhf, histday, histmth), ont été adaptés à tutorial_prod. Si vous voulez activer **d'autres sorties, VERIFIEZ / ADAPTEZ les file*.xml** correspondants (sinon -> plantage !) :

- remplacez les valeurs **"_AUTO_"** pour "output_level" et "enabled" par des vraies valeurs
- mettez **compression_level=0**, au lieu des valeurs par défaut 2 ou 4 (tutorial_prod utilise XIOS en mode "attaché", et non pas "serveur"; il faudrait faire tourner XIOS sur 1 MPI pour pouvoir utiliser compression_level non-nul)
- **DOUBLE ATTENTION si vous voulez XIOS avec veget=CMIP6 : dans ce cas, dans setup.sh on force "omp=2" au lieu de 8 car sinon plantage à cause de xios_orchidee.f90 MAIS ca ralentit le run beaucoup évidemment !** On devrait même utiliser "omp=1" car XIOS n'est pas stable avec OMP ; libIGCM utilise le plus souvent 1MPI pour XIOS. Laurent F va proposer un script avec XIOS en mode "detaché"…

NOTE : après l'installation du modèle, 2 fichiers sont **actualisés automatiquement** par setup.sh dans DEF/XMLfilesLMDZ : **context_lmdz.sh et field_def_lmdz.xml**, qui :

- n'ont pas besoin de réglage utilisateur, et
- doivent être parfaitement cohérents avec la rev LMDZ

→ Installation en mode debug

Si on lance l'installation (install=-install dans main.sh) **avec l'option optim="-debug"** dans setup.sh, alors le modèle s'installe avec **"debug" dans le nom** (voir definition de la variable *LMDZname* dans setup.sh ; la changer si on préfère autrement). L'option **optim="-debug" dans setup.sh ne fonctionne pas si le modèle a été déjà installé !!** (install="" dans main.sh). Voir précisions sur la recompilation ci-dessous.

→ Précisions sur la (re)compilation du modèle

Quand on lance main.sh avec "install=-install" :

main.setup transmet l'option à setup.sh , et

setup.sh lance install_lmdz.sh avec option “bench=0”.

Alors install_lmdz.sh effectue :

- l’extraction du modèle,
- son installation en mode parallèle (option “-parallel mpi_omp” en “dur” dans setup.sh),
- la compilation de tous les éléments **sauf LMDZ** :
    - **XIOS** est compilé si on choisit “xios=-xios” dans main.sh : c’est exigé pour tourner avec des revs sur la branche Orchidee_2_2, optionnel dans les autres cas
    - **ORCHIDEE** est compilé une première fois en séquentiel, puis une 2ème fois en parallèle. Cela permet de recompiler LMDZ dans l’un ou l’autre mode sans recompiler ORCHIDEE.

      NOTE : Une fois l’installation faite, on relancera main.sh avec option install=””, et les scripts du tutorial_prod ne recompileront pas ces elements. Si toutefois on veut recompiler ORCHIDEE, il faut utiliser le script **compile_orc.sh** créé dans modipsl/modeles/ORCHIDEE - après quoi, il faut recompiler le gcm (cf ci-dessous).

      CORRECTION à faire dans install_lmdz.sh : dans compile_orc.sh il manquent probablement des lignes concernant arch…?

Ensuite, setup.sh compile l'exécutable **gcm**, et crée le script **compile.sh** (avec les options choisies) dans modipsl/modeles/LMDZ. Ceci pourra être utilisé pour recompiler le modèle.

Si on lance main.sh avec option init=1 pour créer les fichiers initiaux et aux limites, alors l'exécutable **ce0l** sera compilé aussi.

Après compilation, les exécutables (ce0l et gcm) sont stockés dans modipsl/modeles/LMDZ/bin, sous un nom contenant **certaines options de compilation** :
gcm_${resol}_phy${phylmd}_para_mem${suforch}${sufaer}.e
Exemple typique : *gcm_32x32x39_phylmd_para_mem_orch.e*
pour LMDZ à la résolution 32x32x39, avec physique phylmd, en parallèle, couplé avec Orchidee, et aérosols absents ou prescrits (en tout cas pas en couplage avec le modèle d’aerosols SPLA).

ATTENTION : Quand on lance main.sh, **setup.sh vérifie si l'exécutable avec les options choisies existe** dans LMDZ/bin, et lance la compilation s’il ne le trouve pas. **Il n’est pas capable de détecter le changement des options autres que celles prévues dans le nom du gcm (***${resol}, ${phylmd},${suforch}, ${sufaer}*) , par exemple : **xios, cosp, debug, rrtm** (mai $rad a été rajouté dans le nom du gcm en 2022, suite à l’introduction de ECRad)**, inlandsis** ... **--->** possibles **plantages à l'exécution**.
Alors, **quand on veut recompiler le gcm en changeant des options non prévues dans le nom du gcm**, il faut soit **utiliser compile.sh**, soit - plus sûre - **réinstaller le modèle à partir d’un nouveau répertoire** TPROD.

→ SSTs et SIC

Par défaut, on utilise des **moyennes mensuelles** de SSTs et SIC **AMIP**.
Les fichiers amip* sont téléchargés
depuis https://lmdz.lmd.jussieu.fr/pub/3DInputData/Limit/
dans **LMDZ_Init**, après vérification qu'ils n'y sont pas déjà.

Nouveau (2023) : Possibilité d'utilisation de **ERA5** , fichiers) :

- testée avec libIGCM, pas avec LMDZ_Setup -
- Les fichiers originaux ERA5 de SSTs (sstk*.nc) et SIC (ci*.nc), (mensuels, à fréquence horaire, sont mis à disposition à l'Idris par Sophie.Bouffies-Cloche (Sophie.Bouffies-Cloche@ipsl.fr).
- Anne Cozic les transforme en fichiers annuels de moyennes mensuelles (comme les "amip" utilisés habituellement), et les met à disposition sur les espaces communs ($WORK) au TGCC (subipsl/subipsl/ERA5_SST_as1e5), et à l'Idris (groupe "psl", IGCM/ATM/LIMIT/ERA5.as1e5/original/1440x721) : années **1979 - 2022**

Pour l'année en cours, Anne à mis au point une procédure pour compléter l'année avec les mois manquants pris de l'année précédente (ex : 2022 pour 2023), de façon à avoir aussi une année complète.

Les scripts à utiliser pour la création des limit.nc sont :
–> **get_ci_sst_mixteyear.x** pour preparer une **année mixte**.

→ **get_ci_sst_era5.x** pour une **année complète**,

Avec ces scripts, on met sur les continents les valeurs des la variable "skt", ce qui évite de faire planter create_etat0_limit (qui n'est pas adapté à des valeurs de températures nulles sur les terres).

Dans LMDZ_Setup, il faut adapter les scripts pour faire tourner ce0l avec ces fichiers SST et SIC à 0.25 au lieu des AMIP habituels :
Exemple (F. Cheruy):
/gpfsstore/rech/gzi/rgzi019/LMDZ_Init et
/gpfsstore/rech/gzi/rgzi019/P2OASetup/setup.sh (copié aussi ici :
/gpfsstore/rech/gzi/rdzt896/setup.sh_ERA5_FCheruy)

→ Guidage

Les scripts de pre/post-traitement comme era2gcm_tuto.sh : c'est une bonne pratique de les lancer sur "jean-zay-pp" (plutôt que sur "jean-zay")

Les fichiers de réanalyses

L'emplacement des fichiers des réanalyses sur les différentes machines, et l'accès individuel aux fichiers, sont gérés par Sophie Bouffiès-Cloché, IPSL : **Sophie.Bouffies-Cloche@ipsl.fr** .

Le script **test_ERAfiles_JeanZay.sh** aide à repérer les fichiers absents ou interdits d'accès. **Merci de nous signaler si vous constatez un changement de chemin d'accès** vers les réanalyses, pour qu'on puisse mettre à jour les scripts !

Le répertoire de guidage

Le job (créé à partir de script_SIMU) cherche les fichiers de guidage dans un dossier nommé **GUIDE**.

Quand on crée des fichiers de guidage avec **era2gcm_tuto.sh**, le script crée un dossier nommé **GUIDE_${rea}**, avec rea= la réanalyse choisie (**ERAI, ERA5 ou OPERA**). Ensuite, il crée un **lien** "GUIDE" qui pointe vers GUIDE_${rea} .

Ce mode de fonctionnement a été choisi pour pouvoir changer facilement le répertoire cible en passant d'une réanalyse à une autre.

Par contre si un **répertoire physique** (et non pas un lien) GUIDE existe, il **ne sera pas effacé** - pour éviter du chagrin à ceux qui ont déjà leurs forçages favoris créés.

Le script era2gcm_tuto.nc (ancien era2gcm.sh, renommé en "_tuto" pour le différencier de celui utiliseé avec libIGCM, cf LMDZPedia_nudging)

Les versions depuis nov. 2021 écrivent les fichiers en simple au lieu de double précision (taillé 50% plus petite, très intéressant à haute résolution!).

NOTE : la lecture des variables et des dimensions des netcdfs dans guide_loc_mod.F90 est "case sensitive" (majuscules/minuscules comptent!).

Guidage en humidité : dans les versions de script_SIMU d'avant nov 2021, il manquait le rapatriement sur $SCRATCH du fichier **hur.nc** (s'il existe).

→ Aerosols

Options possibles :

défaut : aerosols=**clim**

autres : **n** (sans aérosols) et **spla** (Simple Aerosol Model, développé par O. Boucher, avec poussières et sels de mer interactifs)

ATTENTION : Si vous voulez passer à "aerosols=clim" dans un répertoire où vous avez tourné une première fois avec "aerosols=n" :
Le téléchargement et l'interpolation des fichiers d'aérosols se fait seulement si "aerosols=clim" dans setup.sh, par le job d'initialisation (lancé si "init=1" dans main.sh), comme la création des fichiers initiaux et limit. Le plus facile (automatique) de passer de "n" à "clim" c'est alors de refaire l'étape d'initialisation :

- effacer les répertoires INIT et LIMIT déjà créés,
- modifier dans config.def : flag_aerosols=6, ok=ade=y, ok_aie=y, ok_cdnc=y, ok_alw=y
- relancer main.sh avec init=1 (mais sans install).

Les fichiers d'aérosols téléchargés seront placés dans le répertoire **LIMIT**, ainsi que leur version après interpolation **horizontale** sur la grille du modèle. Ils ont tous 79 niveaux verticaux. Pas de changement de grille verticale ; si besoin, cela est fait par le gcm.e.

Plus en détail :
Les fichiers de forçage d'aérosols utilisés dans les cas "clim" et "n" sont ceux créés pour CMIP6 avec la configuration LMDZORINCA (avec aérosols et chimie interactive).
Ils sont téléchargés d'ici :

https://www.lmd.jussieu.fr/~lmdz/pub/3DInputData/AerChem/

- Dans le cas "**n**", on utilise seulement le fichier aerosols**1850**_from_inca.nc, correspondant aux aérosols naturels ;
  Le fichier interpolé horizontalement est copié en tant que **aerosols.nat.nc**

- Dans le cas "**clim**", on utilise en plus **aerosols9999_from_inca.nc**, avec les aérosols **moyennés sur 1995-2014**.
  Le fichier interpolé horizontalement est copié en tant que **aerosols.clim.nc**
  (le fichier pour l'année 2000 est aussi disponible ;
  pour l'utiliser à la place de la moyenne "9999", il faut modifier dans setup.sh)

Le job d'initialisation est lancé sur $SCRATCH, dans le sous-répertoire INIT, sous le nom "tmp". Il fait l'interpolation horizontale des fichiers d'aérosols, via le script interp_aerosols.sh (avec "cdo rempacon"). L'interpolation verticale se fait dans le code LMDZ, par la subroutine pres2lev (dans libf/misc/pres2lev_mod.F90) appelée par phylmd/readaerosol_interp.F90 .

→ ORCHIDEE

Les versions de Orchidee gérées sont :
-> la version CMIP6 (qui tourne encore avec la hydrologie Choinel à 2 couches),
-> rev 7983 sur la branche 2_2 (avec les développements de la hydrologie par Agnès Ducharne et collègues),
-> rev 7994 sur la trunk, correspondant à la config libIGCM **LMDZOR_6.4work**

ATTENTION :
1/ les revs post-CMIP6, sur la **branche 2_2 comme sur la trunk, exigent XIOS** !!
La compilation en // sans XIOS plante dans src_parallel/xios_orchidee.f90 :
error #6632: Keyword arguments are invalid without an explicit interface. [N_GLO]
CALL xios_set_axis_attr(axname, n_glo=axlen, VALUE=axval)

2/ Plusieurs fichiers **orchidee.def_6.*** sont à disposition dans DEF :

orchidee.def_6.1 pour veget=CMIP6, ou veget=**7330 ou 7597** sur la branche 2_2
orchidee.def_6.2work pour veget=7983 sur la branche 2_2

orchidee.def_6.4work pour veget=7994 sur la trunk

(dans ce cas, orchidee_pft.def_6.4work sera utilisé automatiquement aussi ; rien ne doit être modifié là-dedans)

Le fichier orchidee.def_6.* cohérent avec l'option "veget=" choisie, sera automatiquement copié en tant que "orchidee.def" lorsqu'on lance les scripts. Avant lancement, il faut donc vérifier (et modifier le cas échéant) les options dans le orchidee.def_6.* concerné.

3/ les revs post-CMIP6, sur la **branche 2_2 comme sur la trunk, sont compilées** pour l'instant par **makelmdz_fcm** avec **"-v orchidee2.1"**

Explications de Josefine :

Question :

D'après notre makelmdz_fcm, les options de compilation pour Orchidee sont :
-v false/orchideetrunk/orchidee2.1/orchidee2.0/orchidee1.9

Que se passe-t-il si on extrait et compile ORCHIDEE r7994 sur la trunk, mais ensuite on compile LMDZ utilisant "-v orchidee2.1" ?

Si on lit makelmdz_fcm :

```
if [[ "$veget" == "orchidee2.0" ]] ; then
     orch_libs="-lsechiba -lparameters -lstomate -lparallel -lorglob -lorchidee"
     CPP_KEY="$CPP_KEY ORCHIDEE_NOUNSTRUCT"
elif [[ "$veget" == "orchidee2.1" ]] ; then
     CPP_KEY="$CPP_KEY ORCHIDEE_NOLIC"
     orch_libs="-lsechiba -lparameters -lstomate -lparallel -lorglob -lorchidee"
elif [[ "$veget" == "orchideetrunk" ]] ; then
     orch_libs="-lorchidee"
else
     orch_libs="-lsechiba -lparameters -lstomate -lparallel -lorglob"
fi
```

Avec orchidee2.1, on va compiler LMDZ avec le cle cpp ORCHIDEE_NOLIC. On ne va donc pas prendre en compte les avancements fait dans le trunk qui permet que les fractions lic soitent traité par ORCHIDEE. Ca marche très bien sans cela puisque dans tout façon c'est en cours de développement et ce n'est pas activé par default. En fait on devrait compiler le trunk avec -v orchideetrunk mais avec landice_opt>2, ce qui ne change rien par rapport à *orchidee2.1*
Si on voudrait activer landice_opt=2 , il faut compiler avec -v orchideetrunk.

Versions de Orchidee qui ne sont plus gerées :

2023-oct : Après mise sur svn de LMDZ_Setup, seulement la r7983 a été retenue. Utiliser un Archive/tutorial_prod.tar_v*** si besoin des revs précédentes 7330/7597.

2021-oct-13 : on ne gère plus des versions de Orchidée antérieures à CMIP6, après ménage" dans tutorial_prod, install_lmdz.sh et modipsl*.tar)

Activer les sorties sechiba

Les sorties sechiba sont produites sur scratch, mais script_SIMU (tmp_SIM job) les efface ligne 363 , avant qu'ils soient rebuild-es ; il faut enlever “sec* “ de la liste des fichiers à effacer :

set +e ; \rm out* sec* sta* list* rest* gcm.e aer* ; set -e

Activer les sorties stomate

Dans orchidee.def

Writefrequency in seconds in sechiba_history.nc : *c’est faux ? ce sont les stomate qui sont activées comme ça ? vérifier avec Josefine*

default = 0

WRITE_STEP = 0 : mettre 86400 pour sorties journalières

Retrouver facilement les options de compilation de Orchidee *(demande de Maëlle)*

tutorial_prod est très orienté vers le travail avec LMDZ. Les scripts du tutorial, ainsi que les scripts compile.sh et compile_ce0l.sh crées par install_lmdz.sh dans modipsl/modeles/LMDZ, permettent de recompiler facilement les exécutables ce0l et gcm. Par contre, la **compilation de Orchidee** se fait normalement **une seule fois, par install_lmdz.sh**, après l’installation du modèle.

Si l’on souhaite recompiler Orchidee - ce qui demande de recompiler ensuite LMDZ aussi :

-> on peut utiliser **compile.sh**, en rajoutant **“-full”** dans les options de compilation de LMDZ, visibles dans $LMDZD/$LMDZname/modipsl/modeles/LMDZ/compile.sh

Avantage: c’est sûr ; Inconvénient, c’est long !

-> PAS TESTé : on peut utiliser **compile_orc.sh** créé par install_lmdz.sh (versions après nov. 2021) dans modeles/ORCHIDEE, similaire au compile.sh du LMDZ. Il affiche ce message à la fin :

“`Maintenant il faut **recompiler LMDZ aussi**, avec ./compile.sh dans modeles/LMDZ`”

→ ORCHIDEE-routage

A partir de Orchidee rev 7597 sur la branche 2_2 (option “veget=7597” dans setup.sh) , et aussi de Orchidee trunk rev 7994 (veget=7994) on peut activer le routage “simple” de Yann Meurdesoif, y compris pour une grille zommée (cf test par Josefine; zoom pas encore testé dans tutorial_prod).

Par défaut, le routage n’est pas activé. Pour l’activer, il faut :

- dans DEF/orchidee.def_6.* , mettre :

RIVER_ROUTING=y

ROUTING_METHOD = simple

RIVER_DESC=y *(NOTE : il passe automatiquement à “n” après l'exécution de la première période de simulation)*

- dans DEF/XMLfilesOR7***/iodef.xml , rajouter après le context_orchidee.xml cette ligne (elle peut déjà être présente à la fin du fichier, mais commentée) :
<context id="orchidee" src="./context_routing_orchidee.xml"/>

Voir aussi : User Guide/Routage Simple

→ Isotopes (since 2023-05-05)

Pour activer les isotopes il faut utiliser cette **seule option** :

lmd_phys="lmdiso" dans **main.sh**
⇒ “phylmdiso” sera utilisé au lieu de “phylmd”
⇒ DEF/PHYS/physiq.def_NPiso sera copié en tant que DEF/physiq.def

Ce fichier **DEF/PHYS/physiq.def_NPiso** est comme le **_NPv6.3**, avec seule différence **iflag_ice_thermo=0** (au lieu de 1). Dans setup.sh, un "sed" assure que le flag est mis à zéro pour "isotopes=y" s'il ne l'est pas déjà.

Dans **setup.sh**, une variable locale "**isotopes**=n" par défaut , passe à "y" si phylmd = "lmdiso" ("phylmd" étant "lmd_phys" reçu du main.sh). Elle est utilisée pour tous les changements automatiques "if isotopes=y"... et est transmise à script_SIMU (dans le "sed" général).

Dans **script_SIMU**, "isotopes" sert (seulement) à changer un flag dans iso.def à partir de la 2ème période de run, pour lire les isotopes dans les restarts.

La **compilation** se fait avec isotopes pour le gcm, alors que pour ce0l, on continue à compiler sans isotopes (S Nguyen changera cela bientôt)

==========

2/ FAQ

==========

→ Vérifier que l’installation et la compilation se sont bien passées

après un premier lancement de main.sh avec l’option “install=-install” :

- Si install_lmdz.sh (lancé par setup.sh, lui-même lancé par main.sh) s’est bien terminé, un fichier *surface_env* , contenant les options pour le modèle de surface (Orchidee ou bucket), doit être présent dans modipsl/modeles ;

- Les options de compilation dans *modeles/LMDZ/compile.sh* sont conformes aux choix précisés dans main.sh et setup.sh ;

- L'executable *gcm* (et *ce0l* si on a lancé main.sh avec "init=1") est présent dans *modeles/LMDZ/bin*, avec les suffixes corrects pour résolution, parallelisme, Orchidee (ou pas), spla (ou pas)

ATTENTION : Si vous avez des plantages à la compilation sans raison apparente, vérifiez l'état de la machine (http://www.idris.fr/status.html), et aussi le degré d'occupation de votre $WORK (commande "idrquota" ; voir http://www.idris.fr/faqs/quotas.html), car si vous êtes proches de la limite, cela peut faire planter.

→ Relancer une simulation qui a planté, sans réinstaller le modèle

Dans main.sh on commente (avec #) la ligne "install=-install", pour laisser : install=""

install=""
#install=-install

-> Si la partie "init" a tourné correctement, et on veut garder les répertoires INIT et LIMIT avec leur contenu déjà créé, alors on met aussi :
init=0 (au lieu de 1)
-> Dans le cas contraire, on efface ces répertoires, et on relance avec init=1.
Rappel : pour une simulation longue, il faut mettre dans main.sh dès le debut toute la période souhaitée (mthini/mthend), pour que les fichiers limit.nc soient tous créés.*

Ensuite, pour la simulation : si le répertoire avec le "name" donné dans main.sh existe déjà, alors quand on relance main.sh, il va poser la question si vous êtes sûrs de vouloir prolonger la simulation.

-> Si le plantage s'est produit dès le début (pas encore de de période terminée - yr ou mo selon le cas - avec sorties complètes et restarts crées), il faut aussi effacer le répertoire de simulation.

-> Si la simulation a planté après avoir tourné correctement au moins une période, et qu'on veut la continuer, on répond "yes/oui" à la question de main.sh.

→ Connaître l'état d'avancement d'une simulation :

Un fichier texte nommé "etat" est automatiquement créé dans $STORE/MAINDIR/SIM , et mis à jour après chaque période (yr ou mo) effectuée (marquée par "OK").
Exemple de contenu :

200001 a faire
200001 OK
200002 a faire

Pour savoir si la simulation tourne encore :

squeue -u $USER
(d'autres commandes très utiles ici :
http://www.idris.fr/jean-zay/cpu/jean-zay-cpu-exec_controle_batch.html)

Si la simulation n'est pas terminée, et votre job n'est plus visible sur la machine (ni en attente, ni en cours d'exécution), alors il faut voir pourquoi : problème de la machine, ou (plus probablement) un joli plantage à aller investiguer sur $SCRATCH ...

→ Lancer une nouvelle simulation dans le même répertoire TPROD

Pour lancer une simulation *Newsimu* à comparer à une autre appelée *Simu* qui a déjà tourné dans le même répertoire TPROD (par exemple après avoir fait des modifs dans le code, et recompilé en lançant le script *compile.sh* dans modipsl/modeles/LMDZ) :

Dans main.sh on change :
name=Simu
par
name=Newsimu

Dans l'idéal la nouvelle simulation utilise des start* et limit* déjà créés dans INIT (ou disponibles dans Simu) et LIMIT, et donc on peut utiliser dans main.sh : *init=0*, ou *init=simu*

On relance main.sh : ça mettra le répertoire *newsimu* à côté de l'ancien *simu*.

→ Recompiler gcm.e et (re)lancer une simulation

(par exemple après des modifications dans le code, éventuellement en mode debug)

Pour recompiler le modèle :
(ici en bref ; voir plus de détails dans la section "Précisions sur la (re)compilation du modèle"

- on se place dans *$LMDZD/$LMDZname/modipsl/modeles/LMDZ*
- on lance le script *compile.sh*, après modification si besoin - par exemple ajout de l'option "-debug"
  NOTE : pour recompiler d'abord Orchidee, utiliser compile_orc.sh dans *modipsl/modeles/ORCHIDEE*

Pour (re)lancer la simulation :
on se place dans $STORE/TPROD et on lance main.sh. Celui-ci détecte si une simulation avec le nom $SIM correspondant à la variable "name" existe déjà dans TPROD.

- Si on recommence la simulation depuis le début, il faut effacer le répertoire $SIM - préférablement même avant de relancer main.sh.

- S'il s'agit de prolonger une simulation qui a déjà tourné pour au moins une periode (yr/mo), alors on lance main.sh et on répond "yes/oui" à la question *"La simulation $SIM existe deja. Il est preferable d arreter et de verifier. / Si vous etes sur de vous, vous pouvez la prolonger / Voulez vous la prolonger '? (y/o)' "*

→ Travailler dans un nouveau répertoire TPRODnew en utilisant un modèle déjà installé

(par soi-même, ou par quelqu'un d'autre si on a les droits d'accès)

C'est très intéressant de le faire si on a exactement les mêmes exécutables, et on change juste des choses qui affectent les fichiers initiaux et aux limites : le calendrier (clim=0 vs 1), mthini&mthend, les paramètres du grid (zoom) ...
Ça reste assez intéressant si on change la résolution : il faut recompiler quand-même, mais cela évite l'extraction du modèle, et les 20 min de compilation de XIOS).
Ce n'est pas une méthode sûre si on change d'autres options de compilation (debug, cosp, aérosols...) - sauf si on est un esprit aventureux, prêt à affronter les plantages éventuels pour cause d'incohérence de flags dans les fichiers *def , ou de xml… vous voilà prévenus !

> **ATTENTION : Quand on crée TPRODnew en copiant un TPROD existant** :
> Parfois on peut trouver cela plus pratique que de repartir de l'archive tutorial_prod.tar, avec toutes ses options par défaut. Les **DEF/*.def** du TPROD qu'on copie correspondent à la dernière simulation qui y a tourné. **Ajustez si besoin les *def pour que les différents flags soient cohérents avec vos nouvelles options**.

En pratique :
Disons qu'on a déjà installé le modèle en faisant tourner main.sh avec "install=-install" dans un répertoire TPROD.
On veut commencer à travailler dans un nouveau répertoire TPRODnew (obtenu soit en copiant TPROD, soit en repartant du tutorial_prod.tar), en utilisant le modèle installé via TPROD.
Pour cela, dans TPRODnew :

- dans lmdz_env, la variable LMDZD doit être la même que dans TPROD (par défaut c'est le cas, car c'est notre $WORK)
- la variable LMDZname definie dans setup.sh doit être la même que dans TPROD elle aussi (par défaut c'est le cas si on utilise la même option de compilation *$optim*, définie dans setup.sh, qui peut être ="" ou "-debug")

On peut aussi utiliser le modèle installé par quelqu'un d'autre, si on y a accès. Même principe : on doit redéfinir les variables LMDZD et LMDZname pour qu'elles correspondent au modèle respectif. Cela peut se faire en dur, par exemple :

```
LMDZD=/gpfswork/rech/gzi/rdzt896
LMDZname=LMDZ20210927.trunkOR7266
```

→ Chercher la cause d'un plantage à l'exécution

Si la simulation n'est pas terminée (cf fichier "etat" sur $STORE/$MAINDIR/$SIM), et le job n'est plus en machine (ce qu'on vérifie par la commande **"**squeue -u $USER"), alors la cause du plantage est identifiable dans l'un ou l'autre de ces 2 fichiers :

$SCRATCH/$MAINDIR/out*

$SCRATCH/$MAINDIR/$SIMnnnnn**/listing** (“nnnnn” étant un nombre qui suit le nom $SIM, et qui est différent à chaque fois qu’on lance la simulation)

La commande **“ls -ltr”** dans $SCRATCH/$MAINDIR ordonne le contenu de ce répertoire avec les derniers fichiers/répertoires créés/modifiés à la fin, et leurs propriétés (pratique à plus d’un titre!).

==========

3/ QUESTIONS EN ATTENTE de réponse

==========

“Y en a eu, y en a plus !” (sic!)
Vous en avez une ?

==========

4/ SUGGESTIONS/SOUHAITS

==========

→ **General**

seasonal.sh : à tester (crée fichier annuel à partir de fichiers mensuels, mais pas le répertoire TS_DA ; quid de TS_MO... ?)

→ **ce0l**

2022 mars : Etienne souhaite des outils pour simulations type “météo”, **Ionela les a**
- création de ECDYN “précis” pour un jour donnée, utilisant réanalyses ERA(5 de préférence).
- création de limit.nc utilisant fichiers réanalyses 6h au lieu des moyennes mensuelles disponibles pour téléchargement sur le site LMDZ
- Adriana : voir echanges par mail avec Ionela et Etienne sur la manière de partager les scripts de Ionela : ex : “toolbox” séparé ?)

→ **Orchidee**

Frédérique (avec Artem) pour quand c est possible, idealement nous avons une reunion MOSAI du 20 au 22 Juin, j'aurais aimé pouvoir faire une simul avant.

la config est la: /ccc/work/cont003/gencmip6/cheruyfr/TESTCM7/modipsl/

elle a été installée avec libIGCM avec LMDZOR_v6.4_work mais en prenant la trunk 4515 de LMDZ suite au poihl de lundi.

Il n y a pas de regriddage supplémentaire.

Pour les fichiers d'Orchidee, voir avec Josefine.

→ **Aérosols**

Discussion Frédérique - Olivier B :
- en réflexion : introduire des aérosols strato : il faudrait qu'ils soient interannuels, non ? par rapport aux années particulières avec des éruptions volcaniques (ex 1991 Pina Tubo). Il faut aussi le script d'interpolation des aerosols strato (moyennes latitudinales)

→ **Autres forçages** climoz à introduire

→ **XIOS**

- pouvoir faire tourner LMDZ avec IOIPSL (pratique pour définir la liste de sorties dans config.def pour le tuning), et Orchidee avec XIOS (pour les revs récentes qui exigent XIOS)

  Idées :
  - *Pas testé, mais d'après Laurent ça ne peut pas marcher : compiler Orchidee avec XIOS et LMDZ avec IOIPSL*
  - *Testé, avec plantage : OR et LMDZ compilés avec XIOS, all_xml_ok=false dans config.def (2 tests, l'un avec iodef pour LMDZOR, l'autre pour Orchidee seul)*
  - *Solution retenue, à travailler : script bash (à base de sed et awk) pour MODIFIER les XML à base des instructions habituelles dans config.def*

- Question de Étienne : Sorties par station pour comparaison avec obs.
  Idee : utiliser le fichier existant histstn ? file_def_histstn_lmdz.xml

==========

→ **COSP**

Nettoyage des fichiers XMLs **:** garder seulement les fichiers nécessaires et suffisantes pour tourner sans cosp (cosp=n dans main.sh) et avec COSP v1 (cosp=y dans main.sh)

En discussion avec Abderrahmane, cf son mail :

Dans les 2 cas il faut utiliser :

field_def_lmdz.xml (rien à changer dans les 2 cas)
field_def_cosp1.xml (rien à changer dans les 2 cas)
*file_def_hist*COSP_lmdz.xml (à activer les sorties si option v1 pour cosp : enabled="TRUE. sinon enabled="FALSE.)*
Pour les niveaux de sorties Cosp, c'était fait en fonction de CMIP6.
Je pense que peut-être le mieux (par défaut) est de se limiter aux sorties Calipso et aussi désactiver les sorties histhfCOSP.nc
En plus de ça, si l'option "v1" est activée, ok_cosp devrait être à y dans config.def

Par ailleurs, il y a un ménage à faire dans DefLists concernant ces fichiers. Il y en a utilisés temporairement mais que je dois supprimer dans la suite.

→ **INLANDSIS**

- **FAIT** dans setup.sh **:** *pouvoir compiler le modèle avec le nouveau schéma de neige inlandsis : rajout de l'option "inlandsis=y/n" dans setup.sh, et la clé "-inlandsis true" à la compilation ; le reste se contrôle dans physiq.def*
  A faire (Étienne, Valentin):
  - restartsis.nc : Après une exécution, inlandsis crée un restart (restartsis.nc) qu'il faut renommer en startsis.nc pour l'exécution suivante.
  Comme Orchidée, pour la première exécution (premier mois ou première année ou premiers 10 ans ..), si on n'as pas fait de spin-up, il n'y a pas besoin de startsis.nc et le modèle part d'une initialisation standard.
  - inlandsis.def (une fois mis à jour par Étienne) dans LMDZ/DefLists

==========

5/ PROBLÈMES CONNUS

==========

(parfois ce ne sont pas des problèmes du tutorial_prod, il est juste le révélateur de bugs dans le code, ou de problèmes particuliers d'une config)

→ **voir README0_HowTo**

→ **Autres :**

Avec XIOS, l'activation des sorties histmth avec output_level > 1 cause un plantage . Cela concerne surtout l'utilisation de tutorial_prod avec ORCHIDEE_2_2, qui exige XIOS.

Le problème est decrit ici :
https://lmdz-forge.lmd.jussieu.fr/LMDZPedia/XIOS:_Error_due_to_too_many_variables_in_an_output_file
et
https://trac.lmd.jussieu.fr/LMDZ/ticket/100

Un moyen de contourner ce problème :

dans file_def_histmth_lmdz.xml : mettre output_level=”1”, et aussi level=”1” pour les variables qu’on veut absolument sortir - en augmentant en même temps le niveau d’autres variables dont on n’a pas besoin, pour ne pas dépasser le nombre max de 232 variables en sortie (défaut pour output_level=1), pour lequel on sait que ça tourne

Quelques détails sur le plantage :

- Quand on augmente output_level pour histmth (déjà à partir de output_level=2, qui implique de rajouter 24 vars en sortie), le job plante très vite comme décrit ici (NetCDF/HDF library issue) : https://trac.lmd.jussieu.fr/LMDZ/ticket/100 ;
- Quand on garde output_level=1, mais on rajoute des variables en sortie en baissant leur level à 1, le job "tourne à vide" (j'ai testé pour les 24 variables qui ont normalement level="2")

Vu que le pb est absent avec libIGCM (XIOS en mode detaché, MPI seul), on peut se demander si c'est lié à l'utilisation de XIOS en mode attaché ? dans tutorial_prod (et par défaut sur OMP=8 mais probablement sans incidence, car ça plante aussi pour OMP=1 ou 2).

→ **Plantage avec XIOS et ORCMIP6 en omp=8 (defaut dans tutorial):**

Ca plante dans moins de 1 minute avec “error” dans fichier sur $scratch “outNNNN” (il faut faire regulierement “tail” de ce fichier pendant que ça tourne pour voir quand ça plante), mais le job reste accroché , il faut le “tuer” .

Par contre **ça passe en forçant dans setup.sh : omp=1** (comme la plupart de la prod CMIP6) **ou omp=2** (comme les tests du Bench) !!

→ **mai 2022 : plantages en debug avec ORCHIDEE et XIOS**

pour ORCMIP6, OR7266, OR7330 : vérifier si avec les revs à jour sur la branche 2_2 ça tourne ?

Pourtant Josefine dit que OR7330 devrait tourner. On attend la rev suivante, testée en couplage avec LMDZ, en fev 2022.

Le plantage pour LMDZOR7266:
/gpfsscratch/rech/gzi/rdzt896/TEST_PROD_v20211019modelOct12/outNPv6.11656920
/gpfsstore/rech/gzi/rdzt896/TEST_PROD_v20211019modelOct12

Le plantage avec veget CMIP6 :
$SCRATCH/TEST_PROD_v20211026cmip/outNPv6.11719473

MAJ Adriana:

2022-02 j’ai rajouté la rev 7597 de Orchidee 2_2, mais je n’ai pas testé en debug ; Khadija a bien tourné avec ceci les simus pour sa thèse.

2023-04 J’ai rajouté la rev 7983 sur la branche Orchidee 2_2, pour Frédérique avec Artem (et Pierre Tiengou ?). Pas (encore) testé en debug.

-> avec IOIPSL à jour 2.2.5 , les revs recentes de OR tournent (trunk ou la branche 2_2), mais OR-CMIP6 plante

Tests dans répertoires : "TEST_PROD_v20220509" avec LMDZ rev 4145 du 9 mai 2022 et un install_lmdz modifié temporairement
- passage à IOIPSL à jour, conjugué avec changement de modipsl.tar
(tests&plantages avec **Or 7526** dans /gpfsstore/rech/**gzi/**rdzt896/**TEST_PROD_v20220331)**

MAJ 2023-03 : Laurent à remplace IOIPSLv2_2_2 par 2_2_5 dans modipsl*tar

→ **plantage de LMDZ seul dans une config. zoomée de Valentin (Antarctique)**

Résolu, si c'était bien ça le problème

le job continue de tourner dans le vide alors que la simu s'est arrêtée (le listing du $SCRATCH ne se remplit plus, mais sans message d'erreur). Un problème de communication entre les processeurs pourrait en être la cause, et a été résolu par Laurent en passant **ntasks_per_node** de 5 à 2 dans le header du fichier script_SIMU sur $STORE. Le modèle semble donc fonctionner avec mpi=32, omp=8 et ntasks_per_node=2. :

→ **résolu : guidage**

Maëlle : dans era2gcm.sh pour utiliser ERA interim il faut aussi faire la conversion de ta en float (c'était fait pour u et v seulement ; r (humidité) n'en a pas besoin

→ **résolu : ce0l (ne) plante (plus) en mode debug : commission faite dans IOIPSL**/src/flincom.f90 ;

Adriana :

J'ai signalé à Anne et Josefine la cause identifiée par Lionel dans IOIPSL et la solution "brutale" qu'il a proposée en attendant mieux :

"Dans flincom.f90, flinopen essaie de lire les attributs globaux r_day, r_month et r_year dans ECDYN.nc, alors que ces attributs n'existent pas. flinopen ne vérifie pas si la lecture des attributs a réussi et utilise ensuite les valeurs dans un appel à ymds2ju. Avec les options de débogage, notamment init=snan, cela fait planter."

J'ai corrigé et commit-é une solution plus fiable convenue avec Josefine.

MAJ 2023-03 : le plantage état dans IOIPSL tag 2_2_2; Laurent a remplacé cette version par la 2_2_5 (déjà corrigée de ce bug).

==========

6/ Problèmes inconnus

==========

(normalement toujours vide, car dès qu'un problème devient connu, il passe au point (4) …)

7/ A faire bientôt

==========

IOIPSL

- 2022-11-24, discuté avec Laurent : il mettra à jour IOIPSL dans les modipsl.tar : 2.2.5 au lieu de 2.2.2 → **FAIT à partir de la testing 20230321** !!
  Cela inclut la correction de flincom.f90 qui permet de faire tourner ce0l compilé en debug (sinon, plantage, job init "hanging")
- "erreur netcdf" -> fichier grilles_gcm.nc pas crée : cause modifs recentes de install_lmdz : **resolu** dans rev 4356 , mieux à prendre : 4357 avec grille_gm.nc plus "propre"

Quelques ajustements dans setup.sh, déjà commencés dans TEST_PROD_v20220713_noaer/setup.sh +

- ecraser compile.sh créé par install_lmdz.sh dans modipsl/modeles/LMDZ par celui créé dans tutorial_prod (qui peut avoir en plus l'option "-dust true") : adapter "You'll need to go in $MODEL/modipsl/LMDZ/, modify and run the script compile.sh" dans setup.sh ;; ALTERNATIVEMENT : est-ce qu'on fait install créér compile.sh seulement si bench=1 ??

- enlever "mem" des options de compil : "option -mem is obsolete (now always on in parallel)" - Attention "mem" rentre dans le nom du gcm ! - discussion sur **mattermost** avec Lionel, canal Installation et tutoriels, titre "**Nom de l'executable "**

- automatiser changement de nbapp_rad : 16 pour rrtm, 24 pour ecrad ? NON ; Reponse Frédéric sur mattermost , canal Ecrad dans Lmdz

"*garder nbapp_rad=16 par défaut"*

Prendre nbapp_rad=16 par défaut, c'est bien et indépendant du choix du code radiatif (hormis la question du bruit de MCica mais on a dit qu'on instruirait cette question plus tard).

C'est un compromis. On sait qu'on fait une erreur en n'appelant le rayonnement que toutes les 1h30. Mais c'est aussi vrai à 1h.

On peut vouloir prendre une fréquence d'appel plus éelvée sur des comparaisons sur site, pour instruire la question de la sensibilité à la fréquence d'appel, ou des questions de compensation d'erreur.

SPLA: NB pas besoin du traceur spla pour l'étape init, le code sait initialiser à zéro les traceurs qu'il ne trouve pas dans start*nc

2023 -mars-31 : reprise des tests avec SPLA dans nouvelle testing 4476 : a tourné
/gpfsstore/rech/gzi/rdzt896/TEST_PROD_20230329spla
modele dans /gpfswork/rech/gzi/rdzt896/LMDZ20230321.trunkrORNONE/modipsl
Pas besoin de t

a faire SPLA :

config.def_spla copié comme config.def - mais à faire automatiquement dans setup.sh !

- reinstaller et test avec xios (JE a tourne avec ; Binta avec ou sans ??)

- à la fin reinstaller avec orchidee

- faire 1 an - 2006 de Bodex - quelles données AERONET utilisait Binta ??
attention plus lent avec spla = MPI seul !! 44 ntasks

- FAIT mais à améliorer : avec aerosols=spla, il faut une pause entre l'étape "init" et la simulation, pour interpoler les vents à 10m sur le grid LMDZ - ce qui necessite grilles_gcm.nc produit par ce0l ; comme pour le guidage d'ailleurs ; il faudrait aussi que le rep de la simu ne soit pas créé ..? Mettre l'arret au même endroit que si l'on est sur jean-zay-pp ; pareil pour le guidage !

il faut dire : "après creation de guidage et uv10m, il faut relancer avec init=0 - car iit aurait deja tourné
et quand on relance, le pb est que le rép de la simu NPv** a été déjà crée - est-ce que ça marche si je dis 'oui' pour la prolonger ? a tester….

DOCUMENTER les flags dans physiq.def : iflag_albedo
(changer de nom en iflag_albedo_oce ?)
https://trac.lmd.jussieu.fr/LMDZ/browser/LMDZ5/trunk/libf/phylmd/ocean_albedo.F90?rev=2227

8/ Un peu de philo …

==========

(depuis échanges mattermost)
pourquoi j'ai choisi de faire les modifs automatiques sur les DEF d'origine :

Pour une 1ere simu, EXP0, dans un TEST_PROD "neuf" :

- on modifie - si on veut - les def dans TEST_PROD/DEF (les seuls def disponibles à ce stade)
- on met nos choix dans main.sh et setup.sh

Au lancement, les scripts vont modifier automatiquement les def selon nos choix dans main et setup. L'intérêt que j'ai vu de faire ces modifs automatiques directement dans TEST_PROD/DEF, c'est qu'elles peuvent même corriger des flags qu'on mettrait nous à la main dans TEST_PROD/DEF, de manière eronnée, en desaccord avec nos choix dans les scripts. De maniere que dans un TEST_PROD donné, les scripts et les TEST_PROD/DEF seront toujours coherentes - en fait la coherence est encore plus sure après avoir fait tourner une fois main.sh.

A mon avis, pour la 1ere simulation EXP0, si on se retrouve à faire autre chose qu'on pense, c'est dans les options de main et setup qu'on le voit beaucoup plus facilement qu'en comparant les defs avant et après les modifs automatiques. Pour les modifs dans un physiq,

pour sensibilité, c'est de toute façon dans TEST_PROD/DEF/PHYS qu'on doit les faire, et de là elles vont arriver dans EXP_N, via $phys indiqué dans main.sh. Comme elles ne sont pas impactees par les modifs automatiques (sauf incohérence avec optionsdes scripts), à nous de garder l'original, non?

Et pour comparer les simulations de sensibilité EXP0...N, n'importe ce qu'on change (des options dans main, setup, ou des params dans TEST_PROD/DEF), on saura ce qu'on a fait par rapport à la reference EXP0 en comparant EXP_N/DEF avec EXP0/DEF ... non ?

Mais il y a peut-être des cas de figure auxquels je n'ai pas pensé…

“**tutorial_prod et configs : défaut, spla, iso, orchidee_v??**
Pour les autres configs que "defaut", il (me) faudrait reflechir à introduire des instructions claires pour le choix manuel - à minima - du config.def au debut. Peut-être pour d'autres fichiers spécifiques aussi.

Juste 2 exemples pratiques : ==> pour SPLA

- j'ai un config.def_spla que c'est moi qui copie en tant que config.def avant tout autre chose, et je peux y regler les sorties ;
- mais le traceur.def_spla, que je n'ai pas besoin de modifier à la main, est copié automatiquement en tant que traceur.def.

==> Pour Orchidee :

- le orchidee.def_$version est copié automatiquement en tant que orchidee.def en fonction de l'option "veget" mise dans setup.sh. C'est OK si l'on utlise les flags par défaut, mais le jour où qq'un veut tourner avec des flags differents, il faut savoir quel orchidee.def* modifier…

Du coup, on pourrait séparer :

- les choix des versions de def, qui doit être fait par chacun en "pleine conscience" selon sa config, dans TEST_PROD/DEF :

  cp -p file.def_MonChoix file.def ; et modifier file.def si besoin

- les changements automatiques faits par les scripts sur le jeu de def choisis TEST_PROD/DEF/*.def ; Ces modifs pourraient se repercuter seulement dans TEST_PROD/EXP/DEF (en repondant à tes habitudes)

Le tri entre ce qu'il faut automatiser pour éviter aux utilisateurs de se perdre, et ce qu'on doit leur confier comme "choix responsable" serait à decider avec chaque "responsable de branche" ...