Geosciences Modelling documentation Documentation

Release 0.1

Anne Fouilloux

Oct 30, 2020

Contents:

CHAPTER 1

Computing facilities

1.1 Geosciences' Servers

- wessel
- sverdrup
- moulin
- geo-hiperf

1.1.1 Getting Started with Linux at the Department of Geosciences

Check out the documentation at https://www.mn.uio.no/geo/english/services/it/help/using-linux/getting-started.html

1.2 Infrastructure as a Service

To get information on how to apply for projects and get virtual machines, etc:

- Main UH-IaaS cloud User Documentation is at http://docs.uh-iaas.no/en/latest/

Some examples on how to use IaaS for Geosciences:

- https://github.com/annefou/geofag-IaaS

1.3 National infrastructure for computational science

- abel
- fram, etc.

Documentation is available at https://www.sigma2.no/

You need to apply for computing and/or storage and may have to pay for getting these resources.

CHAPTER 2

Storage facilities

2.1 National Infrastructure for Research Data

- NIRD
- NIRD toolkit
- Data Management Plan
- Remote Visualization, etc.

To get a full overview of the available services go to https://www.sigma2.no/content/generic-sigma2-services

2.2 lagringshotell storage

- lagringshotell

CHAPTER 3

Models

3.1 WRF model

WRF and WRF-CHEM have been installed on Abel. To check which version is available:

```
module avail wrf

------------------------ /cluster/etc/modulefiles --------------------------
wrf/3.9.1.1
```

To set-up your environment for WRF (no chemistry):

```
export WRF_CHEM=0
module load wrf/3.9.1.1
```

To set-up your environment for WRF-CHEM (with chemistry):

```
module load wrf/3.9.1.1
```

When loading wrf, the following environment variables are defined:

- **WRF_HOME**: directory where WRF has been installed; this can be used to check WRF sources

- **WPS_HOME**: directory where WPS has been installed. It is unlikely you have to check or change WPS source code but you will find in this directory all the metadata required to run WPS (the GEOGRID.TBL, METGRID.TBL, and Vtable files).
- **WPS_GEOG_PATH**: contains theterrestrial static input data. Even if you use your own compiled version of WRF/WPS, it is NOT necessary to download these files again.
- **WRFIO_NCD_LARGE_FILE_SUPPORT**: it is set to 1 to allow you to store large netCDF files (more than 2GB).
- **WRF_EXAMPLES**: directory where all WRF and WRF-CHEM examples are stored. If you wish to run one of this example see our dedicated section on running tutorials.

WRF has been compiled with intel compilers and MPI:

```
 module list

 Currently Loaded Modulefiles:
1) use.own                   4) hdf5/1.8.14_intel        7) jasper/1.900.1         10)␣
↪wrf/3.9.1.1
2) intel/2015.0              5) netcdf.intel/4.3.3.1     8) ncl/6.2.0
3) openmpi.intel/1.8.3       6) intel-libs/2013.sp1.3    9) Anaconda3/5.1.0
```

We suggest you specify the version you wish to use to avoid any problems if we install a new default version (as it is important to stick to the very same version for your simulations).

3.1.1 Running WRF:

The following steps are necessary to run WRF on our systems:

1. WRF Preprocessing System (WPS)
2. WRF System

WRF Preprocessing System (WPS)

Most parameters for WPS must be given in namelist.wps

define_grid.py uses namelist.wps to plot your define

Most parameters for WPS must be given in namelist.wps

- define_grid.py uses namelist.wps to plot your defined grid
- run_geogrid.py runs geogrid.exe to create static data for your domain
- run_ungrib.py runs ungrib.exe to unpack your input GRIB data
- run_metgrid.py runs metgrid.exe to interpolate input data onto your model domain

WRF

Most parameters for WRF must be given in namelist.input

- run_init_wrf.py runs real.exe to initialize WRF and creates two files such as wrfinput_d<domain> and wrfbdy_d<domain> where <domain> is the domain number (01, 02, etc.)
- run_wrf.py runs the numerical integration program wrf.exe

TIPS: to get the usage of these python scripts, use -h or –help. For instance:

```
run_ungrib.py -h
Usage: run_ungrib.py --expid expid --vtable vtable --datadir datadir [--start_date␣
↪startdate --end_date enddate] [--interval_seconds val] [--prefix FILE]

Options:
  -h, --help            show this help message and exit
  -s startdate, --start_date=startdate
                         A list of MAX_DOM character strings of the form
                        'YYYY-MM-DD_HH:mm:ss' specifying the starting UTC date
                        of the simulation for each domain.
  -e enddate, --end_date=enddate
                        A list of MAX_DOM character strings of the form 'YYYY-
                        MM-DD_HH:mm:ss' specifying the ending UTC date of the
                        simulation for each domain
  -t interval_seconds, --interval_seconds=interval_seconds
                        The integer number of seconds between time-varying
                        meteorological input files. No default value.
  -p FILE, --prefix=FILE
                        prefix of the intermediate meteorological data files
  -v Vtable, --vtable=Vtable
                        vtable filename
  -i expid, --expid=expid
                        Experiment identifier
  -d datadir, --datadir=datadir
                        Directory where input fields can be found
```

For each of these python scripts, some arguments are optionals and indicated in squared brackets such as start_date and end_date. If you don't specify them on the command line, it will keep what has been defined in your namelist. This is usually how you will run your "real" cases.

3.1.2 WRF examples:

All WRF examples can be found in $WRF_EXAMPLES

A subdirectory can be found for each example and it contains all you need to run the corresponding examples (namelist.wps, namelist.input, Vtable, workflow.bash, and all the DATA required to run the simulation). You may found one or more files named workflow*.bash. These files are describing the sequence of programs to run for each example and can be divided in two groups:

January2000Case

This case is the East Coast Winter Storm of January 24-25, 2000 (http://cimss.ssec.wisc.edu/goes/misc/000125.html)

To run it, you need to execute each of the statement in workflow.bash:

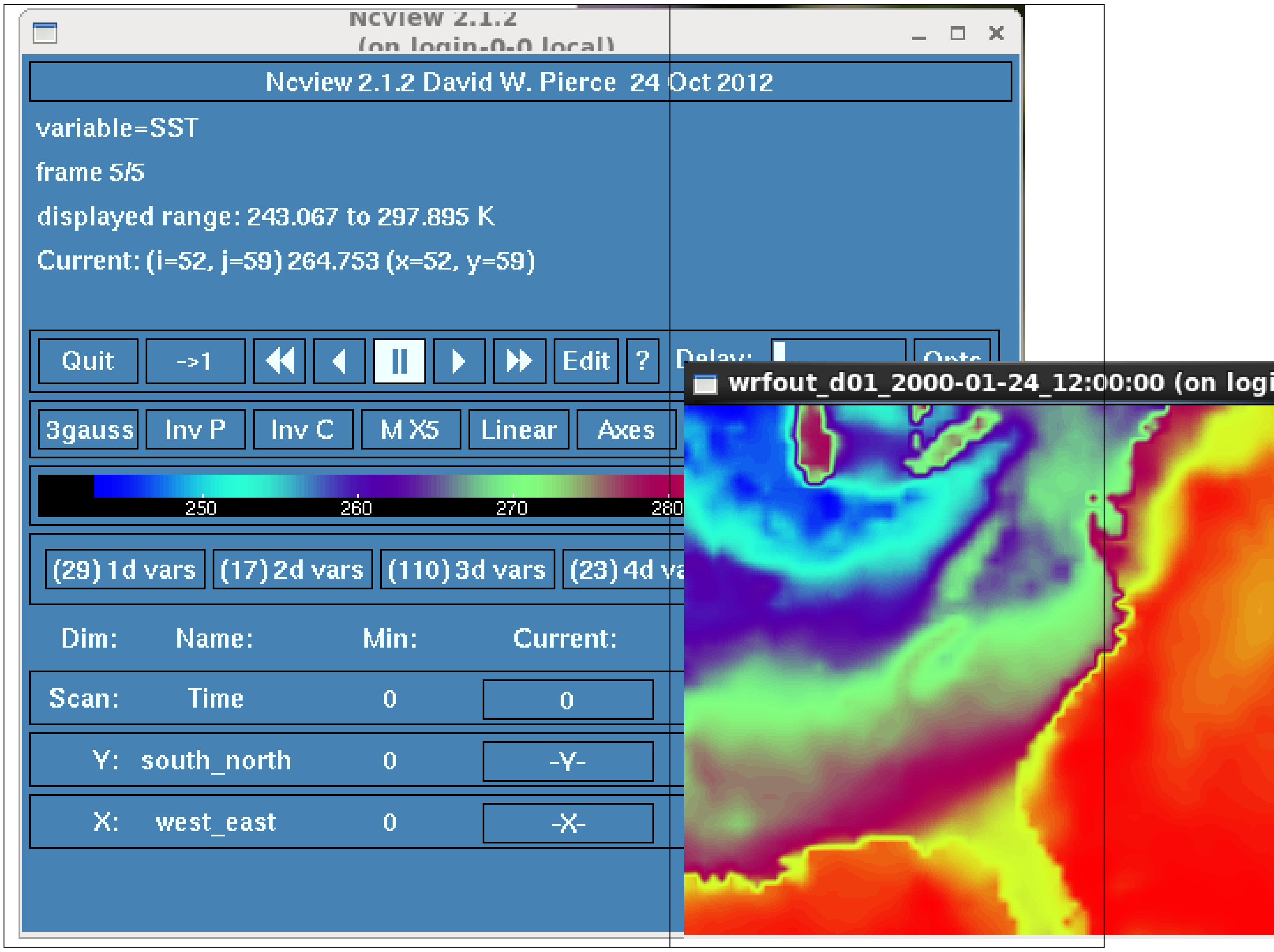
You don't have to change paths or namelists. it has been set-up to execute WPS/WRF in your workdir ($WORKDIR) and a subdirectory named January2000Case (named from your experiment identifier given with –expid option).

To visualize your outputs (named wrfout*), you may use ncview:

```
module load ncview

ncview wrfout_d01_2000-01-24_12:00:00
```

and select the variables you wish to plot. For instance, to visualize SST:


Ncview 2.1.2 (on login-0-0.local)

Ncview 2.1.2 David W. Pierce 24 Oct 2012

variable=SST

frame 5/5

displayed range: 243.067 to 297.895 K

Current: (i=52, j=59) 264.753 (x=52, y=59)

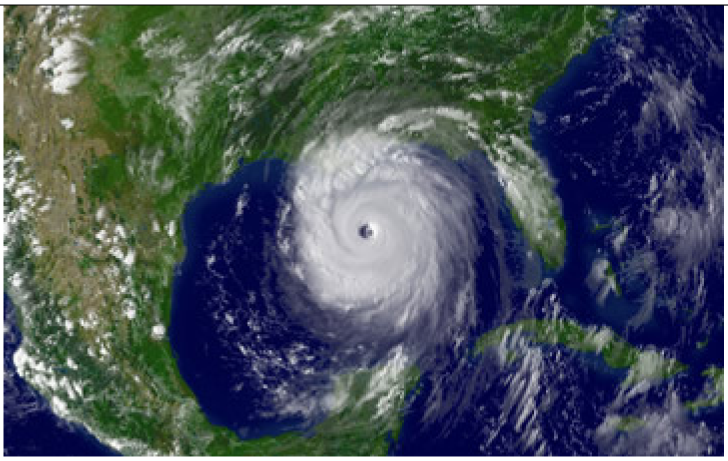
```
Dim:    Name:        Min:   Current:
Scan:   Time         0      0
Y:      south_north  0      -Y-
X:      west_east    0      -X-
```

TIPS: ncview is not recommended for quality graphical displays, but is a very handy tool for a quick first-look at the data.

HurricaneKatrina

As for previous examples, you just need to run workflow.bash and check the directory $WORKDIR/HurricaneKatrina

On August 28, 2005, Hurricane Katrina was in the Gulf of Mexico, where it strengthened to a Category 5 storm on the Saffir-Simpson hurricane scale, packing winds estimated at 175 mph. (http://www.katrina.noaa.gov/).



Hurricane Katrina on August 28, 2005 (image taken from http://www2.mmm.ucar.edu/wrf/OnLineTutorial/CASES/SingleDomain/)

HurricaneKatrinaSST

The goal here is to input SST into WRF model. For these runs we will use the Hurricane Katrina case data (*2005-08-28_00 to 2005-08-29_00*).

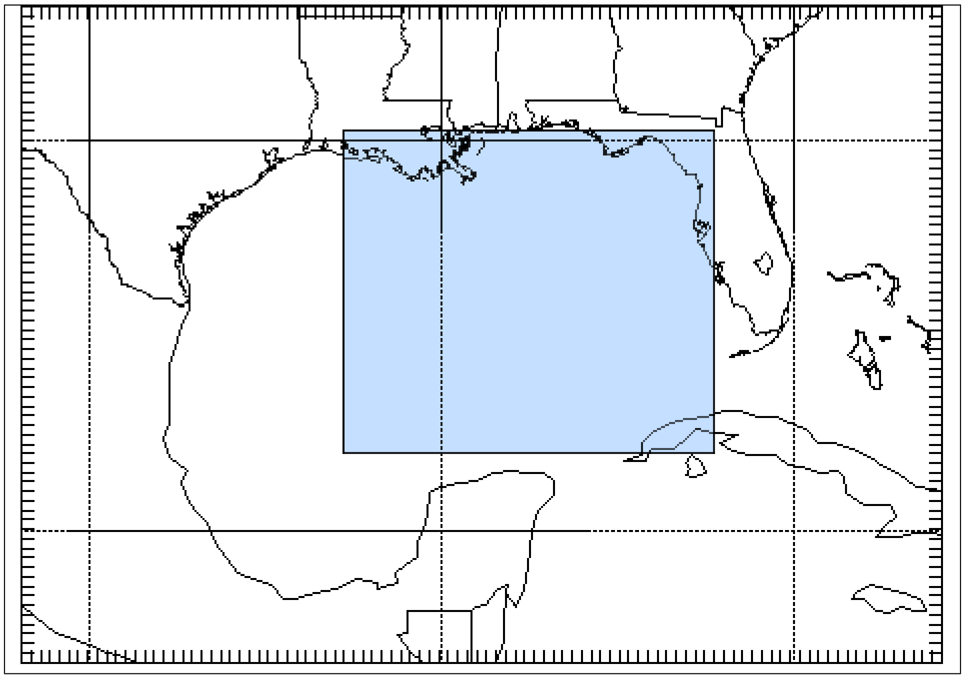
SST are typically added to the model:

a. Use the SST at the initial time as a constant field for all time periods (*this is good for short runs, like real-time runs, where SST is not updated during the WRF model run*) b. As an extra input at each model input time (*this is good for long -months- model runs*)

NestedModelRuns

For these runs we will use the Katrina Hurricane case data (*2005-08-28_00 to 2005-08-29_00*).

The domain we are going to set up is show below (image taken from http://www2.mmm.ucar.edu/wrf/OnLineTutorial/CASES/NestRuns/index.html).



There are a number of different ways to set up nested model runs (*in this tutorial we are only going to set-up 2-way interactive nested runs*).

a. Two-way nested run, with one input file The preprocessing steps for this case will be similar to a single domain setup. The only difference, is that during the wrf.exe execution, a second (*or more*) nest(s) is initiated. The corresponding workflow can be found in *workflow_a.bash*

b. Two-way nested run, with two input files

For this case the pre-processing programs need to be run to create extra input data for the wrf model run. At the WRF model step, one has the choice to:

i. Use all the meteorological and static data for nested domains as input, (see workflow_b.bash)
or ii. Use only the static data for nested domains as input (see workflow_bb.bash).

c. One-way nesting using ndown

ndown is used to run one-way nested runs AFTER wrf has already been run for the mother domain.

One-way nesting can also be done similar to two-way nested runs (*both a and b above*), by simply setting feedback in the WRF namelist.input file equal to 0. The corresponding workflow can be found in *workflow_c.bash*

RestartRun

This case study we will use the same setup as for the Single Domain run, we will just restart it from the previous run. As for all other examples, run workflow.bash

April2005Case

This example is showing how to run from ERA-Interim data instead of GFS data. As you can see in workflow.bash, another Vtable needs to be used (Vtable.EI).

WRF has been compiled with WRF_CHEM=1 and WRF_KPP=1 and is therefore suitable for WRF-CHEM simulations. The three next examples shows how to use WRF-CHEM at UIO.

BiogenicEmissions

This example uses WRF-CHEMand its goal is to get familiar with the methodology by which the MEGAN biogenic emissions are introduced into the WRF-Chem simulation.

This exercise is intended to be completed by students that have knowledge about setting-up and running the WRF numerical model.

There are two workflows:

- option-1 (*workflow-opt1.bash*) uses GOCART-RACM_KPP aerosol option (chem_opt=301), Geunther biogenic emissions (bio_emiss_opt=1). dust, sea salt, DMS, and biomass burning will still be included so keep those options turned on.
- option-2 (*workflow-opt2.bash*)

DustErosion2010

This example uses WRF-CHEM. The purpose of this example is to get familiar with the methodology by which the dust erosion fields are introduced through the WRF Preprocessing System (WPS). The corresponding workflow is called*workflow.bash*

GOCARTaerosols

This example uses WRF-CHEM. A global emissions data set was prepared by a program called "prep_chem_sources" and with this program anthropogenic emissions, GOCART background fields and biomass burning (wild fire) emissions was previously mapped to the user domain. In this exercise you will use the emissions data and follow the methodology for making a WRF-Chem forecast shown here. The corresponding workflow is called *workflow.bash*

3.1.3 Running "long" simulations on abel

SLURM batch system

For most of your WRF runs (wrf.exe), you will need to use SLURM batch system (you can not use more than 30mn CPU on the interactive node and cannot run WRF efficiently in parallel). This requires to create what we call a batch job script: it's a script with additional SLURM directives. SLURM is the current batch system used on abel.

The most important when running wrf is to:

- choose the number of tasks (#SBATCH –ntasks)
- set the amount of memory per task (#SBATCH –mem-per-cpu)
- set the wall clock time limit (#SBATCH –time)

and add the following SLURM directives in your script:

```
# Number of tasks (cores):
#SBATCH --ntasks=8
```

it sets the number of tasks to 8 (i.e. WRF will be using 8 processors for running wrf.exe)

```
  # Max memory usage per task:
  #SBATCH --mem-per-cpu=4000M
  #

it sets the maximum amount of memory you will be using per tasks. In the
```

above example, 4000 Mb per task may be used.

```
# Wall clock limit:
#SBATCH --time=100:0:0
```

it sets a limit of 100 hours for your run. It is advised to split long simulations in chunks and create restart files regularly instead of submitting a huge job.

For more information on the queue system, look here.

Save your outputs on your local machine

When running WRF, your outputs will be stored in $WORKDIR and deleted after about 45 days. It is therefore important to save your outputs on your local machine. This is usually done with an rsync command (see example below) and to avoid having to enter your password everytime you invoke this command, you must set your SSH-keys:

Step 1 on your local machine:

% cd ~/.ssh % ssh-keygen -t rsa

Generating public/private rsa key pair. Enter file in which to save the key (~/.ssh/id_rsa): (just type return or something like ~/.ssh/id_rsa_name of my machine) Enter passphrase (empty for no passphrase): (just type return)

Enter same passphrase again: (just type return) Your identification has been saved in ~/.ssh/id_rsa Your public key has been saved in ~/.ssh/id_rsa.pub The key fingerprint is: Some really long string

Step 2: Then, paste content of the local ~/.ssh/id_rsa.pub file into the file ~/.ssh/authorized_keys on the remote host.

Step 2: Then, paste content of the local ~/.ssh/id_rsa.pub file into the file ~/.ssh/authorized_keys on the remote host. Please make sure the file ~/.ssh/authorized_keys is readbale by you only:

```
chmod og-rwx ~/.ssh/authorized_keys
```

You can do Step 1 on your local machine and Step 2 on abel and then Step 1 on abel and Step 2 on your local machine to have access from and to abel.

Once this is done, you can use rsync command or scp to copy your data back to your local directory. An example in given in the complete batch script given below.

Example of a complete batch script for running wrf on abel

Here we run wrf.exe for the January200Case (see WRF examples). We assume that all the previous steps have been run and run wrf.exe in batch mode.

```
#!/bin/bash
# Job name:
#SBATCH --job-name=wrf
#
# Project:
#SBATCH --account=geofag
#
# Wall clock limit:
#SBATCH --time=100:0:0
#
# Max memory usage per task:
#SBATCH --mem-per-cpu=4000M
#
# Number of tasks (cores):
#SBATCH --ntasks=8
#


# Set up job environment (do not remove this line...)
source /cluster/bin/jobsetup
ulimit -s unlimited


module load wrf


# Run the model (wrf.exe)
run_wrf.py --expid January2000Case --npes 8 \
           --namelist /cluster/software/VERSIONS/wrf/examples/January2000Case/
↪namelist.input

# Copy your data on your local machine
```

(continues on next page)

(continued from previous page)

```
rsync -avz $WORKDIR/January2000Case/wrfout* $USER@sverdrup.uio.no:January2000Case/.
rsync -avz $WORKDIR/January2000Case/wrfrst* $USER@sverdrup.uio.no:January2000Case/.

#
# End of jobscript
#
```

To submit your job script:

```
sbatch job_wrf.sh
```

For more information on Abel see The Abel compute Cluster.

3.1.4 How to change or add new code?

The easiest for changing WRF code is to copy the pre-installed version,

make your changes and then recompile WRF. Here is an example:

```
module load wrf

mkdir $HOME/WORK

cd $HOME/WORK

cp -R $WRF_HOME .

module purge

export WRF_HOME=$HOME/WORK/WRFV3

module load wrf
```

Then you can edit any files and recompile WRF:

./clean -a

./configure

(make sure you choose 15 i.e. dmpar)

```
./compile em_real >& compile.log
```

It is unlikely you have to change WPS but if you need to, you should do the same (define WPS_HOME, etc.).

3.1.5 TIPS and known problems

Often-seen runtime problems

- Segmentation fault (core dumped): if it appears immediately after starting wrf.exe, it may be due to insufficient stack memory. Try:

```
ulimit -s unlimited
```

and rerun wrf.exe

Problems for creating all the WRF inputs when having more than one domain (and get wrfinput_d01 and wrfbdy_d01 only)

- if your namelist.wps has the correct list of dates for each domain, make sure you do not use start_date and end_date option when running run_ungrib.py

For instance:

```
run_ungrib.py --expid  NestedModelRunsB           \
              --vtable  /cluster/software/VERSIONS/wrf/examples/HurricaneKatrina/
↪Vtable.GFS \
              --datadir /cluster/software/VERSIONS/wrf/examples/HurricaneKatrina/DATA/
↪Katrina
```

- If you wish to pass the list of dates, make sure you give the dates for each domain:

```
run_ungrib.py --expid  NestedModelRunsB           \
              --start_date "2005-08-28_00:00:00, 2005-08-28_00:00:00"    \
              --end_date "2005-08-29_00:00:00, 2005-08-28_00:00:00"     \
              --interval_seconds 21600            \
              --prefix FILE                       \
              --vtable  /cluster/software/VERSIONS/wrf/examples/HurricaneKatrina/
↪Vtable.GFS \
              --datadir /cluster/software/VERSIONS/wrf/examples/HurricaneKatrina/DATA/
↪Katrina
```

```
::
```

Possible error messages if the data download is incorrect

Landsea mask

When running metgrid, the following error may result if you download constant fields for the surface:

```
/usit/abel/u1/irenebn/Scandinavia/April2005Case/METGRID.TBL
Processing domain 1 of 1
   LSM:2014-04-15_12
   Z:2014-04-15_12
Processing 2014-04-15_00
   FILE
ERROR: Cannot combine time-independent data with time-dependent data for field␣
↪LANDSEA.mask
--------------------------------------------------------------------------
MPI_ABORT was invoked on rank 0 in communicator MPI_COMM_WORLD
with errorcode 0.
NOTE: invoking MPI_ABORT causes Open MPI to kill all MPI processes.
You may or may not see output from other processes, depending on
exactly when Open MPI kills them.
--------------------------------------------------------------------------
```

This error resulted from having downloaded lsm (variable 129.128), and is solved by avoid downloading constant fields.

3.2 OpenIFS model

- OpenIFS

3.3 Flexpart model

FLEXPART (FLEXible PARTicle dispersion model) is an Lagrangian transport and dispersion model and can be used to study a large range of atmospheric transport processes. The current version is FLEXPART V 10.4 and you can find more information and resources on FLEXPART in the reference paper reference paper or at flexpart.eu.

3.3.1 FLEXPART installation

FLEXPART is currently not available as a preloaded module on SAGA which means that you have to compile the source code yourself. First you need to download the FLEXPART source code by cloning the git repository.

```
$git clone https://github.com/flexpart/flexpart.git
```

After you have cloned the repository you need to edit the paths in the Makefile in flexpart/src/makefile. To compile on saga you would need change the following paths:

```
#Changes to be made in flexpart/src/makefile
F90       = ifort
MPIF90    = mpif90

INCPATH1 = /usr/include
INCPATH2 = /cluster/software/ecCodes/2.9.2-intel-2018b/include
LIBPATH1 = /cluster/software/ecCodes/2.9.2-intel-2018b/lib
```

Before you compile FLEXPART you need to load the necessary fortran modules:

```
module load ecCodes/2.9.2-intel-2018b
module load netCDF-Fortran/4.4.4-intel-2018b
```

When compiling FLEXPART, you can add ncf=yes to enable netCDF output.

```
$ cd flexpart/src
$ make ncf=yes
```

3.3.2 FLEXPART Setup

Setting up a FLEXPART is primarily done by editing the COMMAND, RELEASES, OUTGRID files in the flexpart/options folder. For a in depth explanation of the different settings in flexpart take a look at the reference paper . FLEXPART expect following files and folders to be present:

```
$ ls
options output pathnames
```

- *options* folder containing the flexpart setting files COMMAND, RELEASES, OUTGRID etc.
- *output* folder which should be empty
- *pathnames* a files containing the necessary paths, particularly the path to the atmospheric forcing

```
$ less pathnames
./options/
./output/
/
/path/to/AVAILABLE_WINDFIELDS
============================================
pathnames (END)
```

AVAILABLE_WINDFIELDS is a text file contains the paths to where the atmospheric forcing is stored and has to be formatted in a certain way. (A python script which makes the AVAILABLE_WINDFIELDS). The formatted AVAILABLE_WINDFIELDS file should look something like this.

```
DATE     TIME         FILENAME             SPECIFICATIONS
    YYYYMMDD HHMISS

    ________ ______     __________________      __________________
19990301 000000      /cluster/shared/flexpart/databases/flexpart/WIND_FIELDS/ECMWF/
↪GLOBAL/ERA5/1999/03/EA99030100      ON DISC
19990301 030000      /cluster/shared/flexpart/databases/flexpart/WIND_FIELDS/ECMWF/
↪GLOBAL/ERA5/1999/03/EA99030103      ON DISC
19990301 060000      /cluster/shared/flexpart/databases/flexpart/WIND_FIELDS/ECMWF/
↪GLOBAL/ERA5/1999/03/EA99030106      ON DISC
19990301 090000      /cluster/shared/flexpart/databases/flexpart/WIND_FIELDS/ECMWF/
↪GLOBAL/ERA5/1999/03/EA99030109      ON DISC
```

(continues on next page)

(continued from previous page)

```
19990301 120000      /cluster/shared/flexpart/databases/flexpart/WIND_FIELDS/ECMWF/
↪GLOBAL/ERA5/1999/03/EA99030112      ON DISC
                                             ...........................
```

On Saga atmospheric forcing for FLEXPART is stored in shared folder */cluster/shared/databases/flexpart* and currently only ERA-INTERIM is globally available on Saga, from 1986 until 2016.

3.3.3 Running FLEXPART

To run FLEXPART Saga you would first need to make a job script:

```
#! /bin/bash
#SBATCH --account=nnXXXXk
#SBATCH --time=00:40:00
#SBATCH --ntasks=1
#SBATCH --mem-per-cpu=6G
#SBATCH --mail-user=example@student.geo.uio.no
#SBATCH --job-name=FLEXPART_TEST
#SBATCH --mail-type=FAIL
set -o errexit
set -o nounset
module --quiet purge
module load ecCodes/2.9.2-intel-2018b
module load netCDF-Fortran/4.4.4-intel-2018b
export PATH=/path/to/flexpart/src:$PATH
time FLEXPART
```

Make sure you have the jobscript in the folder where the FLEXPART simulation has been setup and then submit the job to the jobqueue.

```
$ sbatch flexpart_job.sh
```

FLEXPART simulations does not depend on each other, which means that FLEXPART can easily be parallelized by running many FLEXPART instances at the same time, through submitting multiple jobs to the job queue. Though be careful as there is a limit of 10000 jobs that can be in the queue at the same time.

3.4 MITgcm model

The MITgcm model can be used to study both atmospheric and oceanic phenomena. It is available on github at https://github.com/MITgcm/MITgcm and the corresponding documentation at https://mitgcm.readthedocs.io/en/latest/index.html.

3.5 CESM model

3.5.1 Brief description of model

The Community Earth System Model (CESM) is a state-of-the-art, fully coupled, global climate model that developed by the Climate and Global Dynamics Laboratory (CGD) at the National Center for Atmospheric Research (NCAR).

3.5.2 The latest CESM version

The current CESM release is CESM 2.1.0 which is also used for CMIP6 experiments. The recent developments in CESM2 are explained here: http://www.cesm.ucar.edu/models/cesm2/whatsnew.html

3.5.3 How to download?

To download the current release :

```
git clone https://github.com/NordicESMHub/cesm.git my_cesm    cd my_cesm    ./
manage_externals/checkout_externals
```

CESM2 documentation is here : https://escomp.github.io/cesm/release-cesm2/

3.5.4 Simpler Models

The CESM team also released several simpler configurations of CESM2. Currently available simpler atmosphere (CAM) models are listed here : http://www.cesm.ucar.edu/models/simpler-models

3.5.5 The CESM Working Groups

The CESM Working Groups are responsible for development and improvement of the CESM. Scientists in each group meet regularly at least once a year. Information on working groups for each component model and several specific application topics can be found here : http://www.cesm.ucar.edu/working_groups/

3.5.6 How to acknowledge CESM?

CESM is sponsored by the National Science Foundation (NSF) and the U.S. Department of Energy (DOE). If you use CESM model or simulations, it is important to acknowledge the sponsors. Acknowledgment examples are provided here : http://www.cesm.ucar.edu/publications/

3.6 NorESM model

- NorESM

3.7 ESys-Particle model

- ESys-Particle

3.8 SURFEX/Crocus model

(in progress. . .)

3.8.1 Brief description of model

3.8.2 How to run it?

On Wessel, load the module Surfex `module load surfex`. From the terminal then run Surfex/Crocus with the command `s2m`. Type s2m for help via the terminal. An offline simulation command for running CROCUS can look like:

```
s2m -b 20181001 -e 20181009 -f FORCING_xarray11.nc -g -o outputTest/
```

3.8.3 How to create forcing data for offline simulation?

The file format must be Netcdf, in format NETCDF3_CLASSIC, with all variables in float64 type. The dimension time must be set as an unlimited dimension. All variable should have an attribute _fillValue=-9999999.0

An example file can found: https://opensource.umr-cnrm.fr/projects/snowtools_git/repository/revisions/master/raw/DATA/FORCING_test_base.nc

https://opensource.umr-cnrm.fr/projects/snowtools_git/wiki/Generate_your_own_forcing_files

Required variables:

- LAT (Number_of_points) float64 . . .
- LON (Number_of_points) float64 . . .
- LWdown (time, Number_of_points) float64 . . .
- PSurf (time, Number_of_points) float64 . . .
- Qair (time, Number_of_points) float64 . . .
- Rainf (time, Number_of_points) float64 . . .
- SCA_SWdown (time, Number_of_points) float64 . . .
- Snowf (time, Number_of_points) float64 . . .
- Tair (time, Number_of_points) float64 . . .
- UREF (Number_of_points) float64 . . .
- Wind (time, Number_of_points) float64 . . .
- Wind_DIR (time, Number_of_points) float64 . . .
- ZREF (Number_of_points) float64 . . .
- ZS (Number_of_points) float64 . . .
- aspect (Number_of_points) float64 . . .
- slope (Number_of_points) float64 . . .

The forcing data can be previewed with netcdf programs such as ncview on linux (http://meteora.ucsd.edu/%7Epierce/ncview_home_page.html).

3.8.4 Where to find further information?

Vionnet, V., et al. “The detailed snowpack scheme Crocus and its implementation in SURFEX v7. 2.” Geoscientific Model Development 5 (2012): 773-791. doi:10.5194/gmd-5-773-2012

https://opensource.umr-cnrm.fr/projects/snowtools_git/wiki

CHAPTER 4

Data

4.1 ECMWF data

All public datasets can now be accessed and retrieved from the Climate Data Store at https://cds.climate.copernicus.eu

Information on how to migrate from ECMWF API to CDS API is available at https://confluence.ecmwf.int/display/CKB/C3S+ERA5%3A+Web+API+to+CDS+API

The link above is also useful to get information on datasets that are not yet officialy available from the Climate Data Store (for instance ERA5 monthly means of daily means).

CHAPTER 5

Indices and tables

- genindex
- modindex
- search