CAPT Procedures

Hsi-Yen Ma, Stephen A. Klein, Jim Boyle (retired) and John Tannahill (retired)

Lawrence Livermore National Laboratory

06/17/2020 v1.0

The work is funded by the Regional and Global Model Analysis Program Area and Atmospheric System Research Program of the U.S. Department of Energy as part of the Cloud Associated Parameterizations Testbed(CAPT). This work is supported under the auspices of the U.S. Department of Energy by LLNL under contract DE-AC52-07NA27344. LLNL-TM-811726

CAPT Procedures

1. Introduction

This CAPT Procedures document is primarily focused on what is required to run DOE E3SM EAM/ELM hindcasts on US DOE NERSC supercomputing facility (https://www.nersc.gov). However, some scripts which were previously used for CESM CAM5 could be used for CESM CAM6 with small or no modification.

1.1. Group membership

There are two NERSC Computing "groups" that are relevant to accomplishing the various tasks described in this document =>

```
mp193     # For access to CAPT data & runs    (PI: Hsi-Yen Ma).
e3sm      # For access to E3SM files          (PI: Ruby Leung).
```

1.2. Local repository directory

To find any CAPT-related files, one needs to find out where the local repository is currently located. **$CAPTUTILS** will be used for this location.

1.3. env_nersc_uvcdat_e3sm.csh file

The **env_nersc_uvcdat_e3sm.csh** file defines a number of useful environment variables, aliases, etc. These definitions will be used throughout this document. The contents of the file can be viewed at:

$CAPTUTILS/**env_nersc_uvcdat_e3sm.csh**

Now source this file with:

source $CAPTUTILS/**env_nersc_uvcdat_e3sm.csh**

This will also now permanently define $CAPTUTILS.

Note that users may need to re-source this file if the settings are lost. It may also be beneficial to create a personal copy of this file, so that it can enhanced/modified as desired.

1.4. Required software for non-NERSC machines

Community Data Analysis Tools (CDAT):

```
https://github.com/CDAT/cdat/wiki/install/
https://cdat.llnl.gov/
```

Climate Data Operators (CDO):

```
https://code.mpimet.mpg.de/projects/cdo/
```

Fortran compiler: Some filtering and interpolation codes require a Fortran compiler

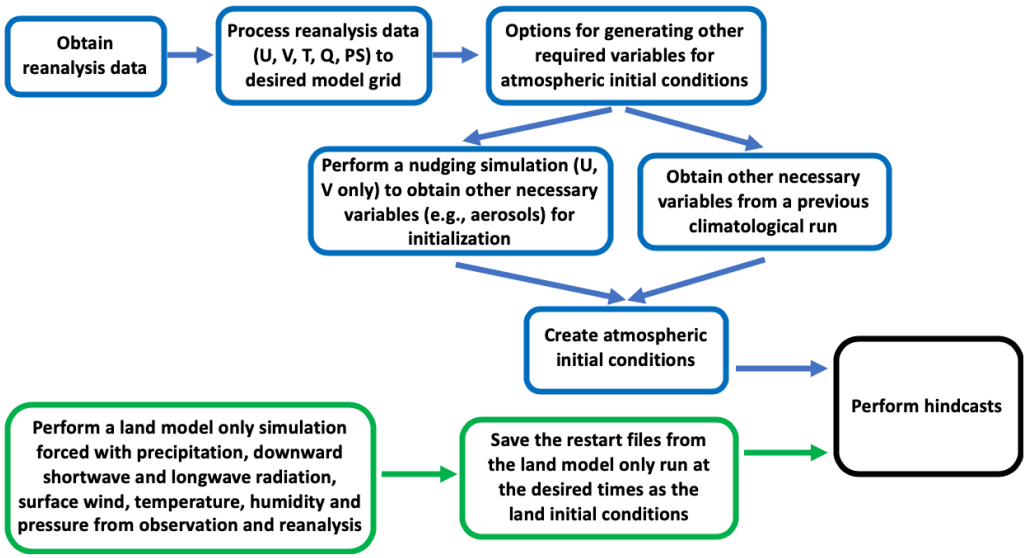
```
https://gcc.gnu.org/wiki/GFortran/
```

ESMF Software for horizontal regridding

```
https://www.earthsystemcog.org/projects/esmf/software/
```

1.5. Workflow of generating initial conditions for hindcast experiments

The steps to perform a hindcast experiment can be summarized by the following flowchart. The blue boxes involve the atmospheric model and the green boxes involve the land model. The following sections will explain the steps and the necessary scripts for generating initial conditions.



2. Obtaining input data

2.1. Obtaining ice & sst data

2.1.1. Weekly option

Weekly 1x1 ice and sst data files can be obtained using FTP from the NOAA Physical Sciences Laboratory (PSL); data is available from 1981 to the present:

```
https://psl.noaa.gov/data/gridded/data.noaa.oisst.v2.html
```

noaa.oisst.v2:
NOAA Optimum Interpolation (OI) Sea Surface Temperature (SST) V2

File name examples are:

```
Ice concentration:        icec.wkmean.1990-present.nc
Sea surface temperature:  sst.wkmean.1990-present.nc
```

The file ending time is not specified, since this is a rolling archive; it is just given as "present". Use the following script to print out the end time to use, and then incorporate it into the file name so that it is easier to keep track of things:

```
python check_endtime.py sst.wkmean.1990-present.nc
```

Output will be something like:

```
2020-6-1 0:0:0.0
```

Then execute:

```
mv icec.wkmean.1990-present.nc \
   icec.wkmean.1990-01-01-2020-06-01.nc
mv sst.wkmean.1990-present.nc \
   sst.wkmean.1990-01-01-2020-06-01.nc
```

2.1.2. Daily option

Daily 0.25x0.25 ice and sst data files can be obtained using FTP from the Earth System Research Laboratory (ESRL); data is available from 1982 to the present:

```
https://psl.noaa.gov/data/gridded/data.noaa.oisst.v2.highres.html
```

File name examples are:

```
Ice concentration:        icec.day.mean.2019.nc
Sea surface temperature:  sst.day.mean.2019.nc
```

Each file covers a one year time period.

Download all the individual yearly files covering the desired start time to end time. As in the weekly data, this is a rolling archive. Use the following script to print out the end date of the last file, giving it an argument of the name of the last file:

```
python check_endtime.py sst.day.mean.2019.nc
```

Output will be something like:

```
2019-12-31 0:0:0.0
```

Then use UVCDAT's cdscan to put all of the ice netcdf files into a single xml file and all of the sst netcdf files into another xml file:

```
cdscan -x icec.dymean.2000-01-01-2019-12-31.xml icec.day.mean.*.nc
```

```
cdscan -x sst.dymean.2000-01-01-2019-12-31.xml  sst.day.mean.*.nc
```

2.2. Obtaining and processing ERA5 model level data

The ERA5 model level data is obtained from the Climate Data Store (CDS) infrastructure; reference is:

https://cds.climate.copernicus.eu/#!/search?text=ERA5&type=dataset
https://confluence.ecmwf.int/display/CKB/How+to+download+ERA5

The model level ERA5 data is recommended because it has the full 137 vertical levels, which provides higher vertical resolution than the pressure level data.

These are large data sets and need long uninterrupted times to transfer. Note that the server can be taken off-line and connections can go down, so retrievals can fail occasionally and must be re-started. Because of this, fine-grained restarts are employed (i.e., hourly).

Get the ERA5 data from CDS using something like:

```
python get_ERA5.py 2000 1 1 0 2000 12 31 23
```

Arguments are:

```
year_start month_start day_start hour_start year_end month_end
day_end hour_end
```

Output files will look like:

```
UVTQPS-20000101_00.grib
```

And will contain the variables U, V, T, Q, Divergence, Velocity Potential, Surface Geopotentail and PS (lnps), one file every 1 hours over the specified time. The data is in packed spherical harmonic coefficients on a reduced gaussian grid.

The next step requires the use of CDO (Climate Data Operators):

https://code.mpimet.mpg.de/projects/cdo/wiki

CDO is used to:

- Transfer data from the reduced to the full gaussian grid, N480.
- Take the spherical harmonics and generate values on the full gaussian grid.
- Make a grads control file, so that CDMS can read the data.

Put the data on the N480 grid using something like:

```
python ERA5toN480nc.py 2000 1 1 0 2000 1 31 23 datapath/
```

Required arguments are:

```
year_start month_start day_start hour_start year_end month_end
day_end hour_end data_dir
```

Optional arguments are:

```
cdo_exe
```

Output files will look like:

```
UVTQPS-20000101_00.nc       # Hourly netcdf data on a N480 gaussian
                            # grid.
```

Then use UVCDAT's cdscan to put groups of the ".ctl" files (e.g., monthly) into a single xml file:

```
cdscan -x UVTQPS-20000101-20000131_N480.xml *.nc
```

2.3. Obtaining Interim data

Note that ERA Interim is being phased out. ECMWF suggests users to migrate to ERA5.

The Interim data is obtained from the ECMWF MARS (Meteorological Archival and Retrieval System) server; reference is:

```
https://www.ecmwf.int/services/archive
```

The interface to this server is implemented by:

```
ecmwf.py
```

These are large data sets and need long uninterrupted times to transfer. Note that the server can be taken off-line and connections can go down, so retrievals can fail occasionally and must be re-started. Because of this, fine-grained restarts are employed (i.e., daily).

Get the Interim data from ECMWF MARS server using something like:

```
$CAPTUTILS/get_interim.py 2000 1 1 2000 12 31
```

Arguments are:

```
year_start month_start day_start year_end month_end day_end
```

Output files will look like:

```
UVTQPS-20000101_00.grib
```

And will contain the variables U, V, T, Q, Divergence, Velocity Potential, Surface Geopotentail and PS (lnps), one file every 6 hours over the specified time. The data is in packed spherical harmonic coefficients on a reduced gaussian grid.

The next step requires the use of CDO:

- Transfer data from the reduced to the full gaussian grid, T255.
- Take the spherical harmonics and generate values on the full gaussian grid.
- Make a grads control file, so that CDMS can read the data.

Put the data on the T255 grid using something like:

```
python interim2t255.py 2000 1 2001 12 datapath/
```

Required arguments are:

```
year_start month_start year_end month_end
data_dir
```

Optional arguments are:

```
cdo_exe
```

Output files will look like:

```
UVTQPS-20000101_T255.grib      # Daily grib data on a T255 gaussian
                               # grid.
UVTQPS-20000101_T255.grib.gmp  # A grads daily mapping file, needed
                               # for direct access.
UVTQPS-20000101_T255.grib.ctl  # A grads daily control file, used by
                               # CDMS to read the data.
```

Then use UVCDAT's cdscan to put groups of the ".ctl" files (e.g., monthly) into a single xml file:

```
cdscan -x UVTQPS-20000101-20000131_T255.xml *.ctl
```

3. Input data processing

3.1. SST/ice data processing

3.1.1. Weekly option

Copy the following script:

```
cp $CAPTUTILS/oisst_weekly.py .
```

Note that currently, the resolution is hard-coded in this script and will need to be modified as appropriate.

Arguments are:

```
weekly_sst_dir
weekly_sst_filename
```

After editing the script as needed, execute something like:

```
python oisst_weekly.py $SSTDIR/ \
                       sst.wkmean.1990-01-01-2019-12-31.nc
```

This will combine the separate ice and sst files into a single file with a name like:

```
sst_weekly_cdcunits_1x1_1990-01-01-2019-12-31.nc
```

3.1.2. Daily option

The downloaded daily ice/sst data has missing values over the continents, which would cause model runs to fail, so an NCAR code is used to fill in the grid:

```
fill_msg_grid.f
```

Note that the f2py3 steps below will need to be done by someone that has write access to $CAPTUTILS and $CAPTUTILS/zfortran. Normal users should be able to just skip these steps as the needed file:

```
fill_msg_grid.so
```

Will already be in $CAPTUTILS.

Create a .pyf signature file by executing (be sure to re-source env_lc.csh as necessary):

```
cd $CAPTUTILS/zfortran  # Or copy fill_msg_grid.f to local work
                        # space.
f2py3 -h fill_msg_grid.pyf -m fill_msg_grid fill_msg_grid.f
```

Edit the pyf file and add intent(inout) to xio and a as follows:

```
double precision dimension(mx,ny),intent(inout) :: xio
double precision dimension(il,jl),intent(inout) :: a
```

Build a Python-callable .so version of the code by executing:

```
f2py3 -c --fcompiler=gnu95 fill_msg_grid.pyf fill_msg_grid.f
```

Which produces:

```
fill_msg_grid.so
```

Which will be imported into:

```
oisst_daily.py
```

Now put things in their proper places:

```
mv fill_msg_grid.so $CAPTUTILS/zfortran  # As necessary.
cp $CAPTUTILS/zfortran/fill_msg_grid.so $CAPTUTILS
```

Lastly execute something like:

```
python oisst_daily.py $CSSEFDATA/sst \
                      sst.dymean.2000-01-01-2012-01-11.xml \
                       ONEXONE
```

Required arguments are:

```
daily_sst_dir
daily_sst_filename
```

Optional arguments are:

```
targ_grid_res       # Target grid resolution; choices are
                      'HALFXHALF', 'ONEXONE', or 'TWOXTWO'
                      ['ONEXONE'].
```

This will combine the separate ice and sst files into a single file with a name like:

```
sst_daily_cdcunits_1x1_2000-01-01-2012-01-11.nc
```

3.2. ERA5 & Interim data processing

3.2.1. Generating ESMF mapping files for horizontal interpolation

Use the ESMF regridding utility, ESMF_RegridWeightGen, to generate mapping files for later use in doing horizontal interpolation; reference is:

```
https://esmf-org.github.io/801branch_docs/ESMF_usrdoc/
```

These mapping files are produced using two SCRIP grid descriptor files, source and destination, and the ESMF software.

Get the SE/ne30 SCRIP grid descriptor file; it will look something like:

```
ne30np4_pentagons_091226.nc
```

Generate an Interim SCRIP grid descriptor file using something like:

```
python make_scrip.py \
   reanalysis_datapath/ \
   UVTQPS-20000101-20000131_N480.xml \
   U \
   ERA5_N480
```

Translate output file with:

```
nccopy -k 2 ERA5_N480_scrip.nc ERA5_N480_scrip_n3.nc
```

Output grid descriptor file will be:

```
ERA5_N480_scrip_n3.nc
```

To generate ECMWF/ERA5 to ne30np4 maps with **ESMF_RegridWeightGen**:

```
ESMF_RegridWeightGen \
   -d ne30np4_pentagons_091226.nc \
   -i \
   -m bilinear \
   -s ERA5_N480_scrip_n3.nc \
   -w map_ERA5N480_to_ne30np4_bilin.nc
```

3.2.2. Wrapping Fortran horizontal filtering routines (CESM FV grid only)

The interpolation to the FV grid will use a conservative scheme in a later section. These interpolated fields will need an additional smoothing to minimize noise at the startup of the forecast. The FORTRAN routines filter25.f90 and filterG.f90 accomplish this smoothing. **Note that they are only currently used for the FV grid.**

Note that the f2py3 steps below will need to be done by someone that has write access to $CAPTUTILS and $CAPTUTILS/zfortran. Normal users should be able to just skip these steps as the needed files:

```
filter25.so
filter.so
```

Will already be in $CAPTUTILS.

Create a .pyf signature file by executing (be sure to re-source env_lc.csh as necessary):

```
cd $CAPTUTILS/zfortran  # Or copy filter25.f & filterG.f to local
                        # work space.
f2py3 -h filter25.pyf -m filter25 filter25.f
f2py3 -h filterG.pyf  -m filterG  filterG.f
```

Edit the pyf files and add intent(in,out) to zi and f as follows:

```
real intent(in,out),dimension(li,lj) :: zi  # filter25.f
real intent(in,out),dimension(n,m)   :: f   # filterG.f
```

Build a Python-callable .so version of the code by executing:

```
f2py3 -c --fcompiler=gnu95 filter25.pyf filter25.f
f2py3 -c --fcompiler=gnu95 filterG.pyf  filterG.f
```

Which produces:

```
filter25.so
filterG.so
```

Which will later be imported into:

`filter_field.py`

Lastly put things in their proper places:

```
mv filter25.so $CAPTUTILS/zfortran  # As necessary.
cp $CAPTUTILS/zfortran/filter25.so $CAPTUTILS
mv filterG.so $CAPTUTILS/zfortran   # As necessary.
cp $CAPTUTILS/zfortran/filterG.so  $CAPTUTILS
```

3.2.3. Wrapping Fortran vertical interpolation routines (SE/FV)

The FORTRAN routines used for the vertical interpolation follow the procedures used at ECMWF for initializing the model using foreign analyses. **Note that this vertical interpolation code is used for both the SE and FV grids.** The following shows how to build a python module from the Fortran code.

Note that the f2py3 steps below will need to be done by someone that has write access to $CAPTUTILS and $CAPTUTILS/zfortran. Normal users should be able to just skip these steps as the needed file:

```
vertical_interpolation.so
```

Will already be in $CAPTUTILS.

Create a .pyf signature file by executing:

```
cd $CAPTUTILS/zfortran  # Or copy vertical_interpolation.f90 to
                        # local work space.
```

```
f2py3 -h vertical_interpolation_code.pyf \
      -m vertical_interpolation_code \
      vertical_interpolation_code.f90
```

Edit the pyf files and add intent(out) to ts, t_new, xxo, xxo, and ps_new as follows:

```
real*8 dimension(plon,plat),intent(out),depend(plon,plat) :: ts
real*8 dimension(plev,plon,plat),intent(out),depend(plev,plon,plat)
:: t_new
real*8
dimension(plevo,plon,plat),intent(out),depend(plevo,plon,plat) ::
xxo
real*8
dimension(plevo,plon,plat),intent(out),depend(plevo,plon,plat) ::
xxo

real*8 dimension(plon,plat),intent(out),depend(plon,plat) :: ps_new
```

Build a Python-callable .so version of the code by executing:

```
f2py3 -c --fcompiler=gnu95 \
      vertical_interpolation_code.pyf \
      vertical_interpolation_code.f90
```

Which produces:

```
vertical_interpolation_code.so
```

Which will later be imported into:

era52e3sm_ne30.py
era52e3sm_ne120.py
interim2cesm_se.py
interim2cesm_fv.py

Lastly put things in their proper places:

```
mv vertical_interpolation.so \
   $CAPTUTILS/zfortran  # As necessary.

cp $CAPTUTILS/zfortran/vertical_interpolation_code.so \
   $CAPTUTILS
```

Note that ESMF horizontal regridding currently does not work with extreme high resolution regridding (e.g., ERA5 N480 grid to SE ne1024 grid). One solution is to use **TempestRemap** (https://github.com/ClimateGlobalChange/tempestremap). Scripts and instruction on extreme high-resolution remapping with TempestRemap will be provided in the future update.

3.2.4. Converting ERA5 state files to E3SM/EAM ne30/ne120

Note that these scripts are only for E3SM/EAM ne30/ne120. Nevertheless, these scripts can be used for CESM2/CAM6 with minimum changes.

The ERA5 data has its vertical structure defined by:

get_ab_era5.py

Which uses the following two data files:

```
era5_137lev_ab_full.dat
```

```
era5_137lev_ab_half.dat
```

These two files are first produced by copying the data at:

```
https://www.ecmwf.int/en/forecasts/documentation-and-support/137-model-levels
```

To produce the text file:

```
ERA5_L137.txt
```

And then running:

```
make_ab_ERA5.py
```

Once the two ".dat" files above have been produced, they can just be copied for use from:

```
$CAPTUTILS/zdata
```

To wherever either of the following scripts will be executed:

```
era52e3sm_ne30.py
era52e3sm_ne120.py
```

For ne30 regridding, copy the following two files to wherever **era52e3sm_ne30.py** will be executed:

```
cp $CAPTUTILS/zdata/era5_137lev_ab_full.dat <wherever>
cp $CAPTUTILS/zdata/era5_137lev_ab_half.dat <wherever>
```

Convert ERA5 state files to E3SM ne30 netcdf files using something like (this version uses ESMF mapping files):

```
python era52e3sm_ne30.py \
   2000 1 1 0 \
   24 \
   era5datadir/ \
   UVTQPS-20000101-20000131_N480.xml \
   mapfiledir/ \
   map_ERA5N480_to_ne30np4_bilin.nc \
   hybridfiledir/cami_mam3_Linoz_ne30np4_L72_c160214.nc \
   topofiledir/USGS-gtopo30_ne30np4_16xdel2-PFC-consistentSGH.nc
```

This script imports:

```
data_era5  # Provides data interface to ERA5 nc data.
vertical_interpolation_code  # Described in a previous section.
other misc. modules          # Note that no filtering is needed.
```

Required arguments are:

```
year, month, day, hour
ntimes                  # Number of hourly time intervals.
state_xmlfile_dir       # Must also contain the needed ERA5 data
state_xmlfile_name
map_file_dir
map_file_name
hybrid_file
topo_file
```

Optional arguments are:

```
-ft=$ | --file-tag=$
```

The following file will be produced:

```
era5_ne30_2000010100_2000010123_TQUV.nc
```

For ne120 regridding, copy the following two files to wherever **era52e3sm_ne120.py** will be executed:

```
cp $CAPTUTILS/zdata/era5_137lev_ab_full.dat <wherever>
cp $CAPTUTILS/zdata/era5_137lev_ab_half.dat <wherever>
```

Convert ERA5 state files to E3SM ne120 netcdf files using something like (this version uses ESMF mapping files):

```
python era52e3sm_ne120.py \
   2000 1 1 0 \
   24 \
   era5datadir/ \
   UVTQPS-20000101-20000131_N480.xml \
   mapfiledir/ \
   map_ERA5N480_to_ne120np4_bilin.nc \
   hybridfiledir/cami_mam3_Linoz_0000-01-ne120np4_L72_c160318.nc \
   topofiledir/USGS-gtopo30_ne120np4_16xdel2-PFC-consistentSGH.nc
```

This script imports:

```
data_era5                  # Provides data interface to ERA5 nc data.
vertical_interpolation_code  # Described in a previous section.
other misc. modules          # Note that no filtering is needed.
```

Required arguments are:

```
year, month, day, hour
ntimes                   # Number of hourly time intervals.
state_xmlfile_dir        # Must also contain the needed ERA5 data
state_xmlfile_name
map_file_dir
map_file_name
hybrid_file
topo_file
```

Optional arguments are:

```
-ft=$ | --file-tag=$
```

The following file will be produced:

```
era5_ne120_2000010100_2000010123_TQUV.nc
```

3.2.5. Converting Interim state files to CESM FV/SE

Note that Interim data was only used for CESM/CAM5 FV/SE hindcasts. Nevertheless, these scripts can be used for E3SM/EAM SE with minimum changes.

To save time, a separate set of Interim scripts has been used below, although the differences between them and their ERA5 counterparts are minimal.

The Interim data has its vertical structure defined by:

`get_ab_interim.py`

Which uses the following two data files:

```
erainterim_60lev_ab_full.dat
erainterim_60lev_ab_half.dat
```

These two files are first produced by copying the data at:

https://www.ecmwf.int/en/forecasts/documentation-and-support/60-model-levels/

To produce the text file:

```
model_levels_interim_60lev.txt
```

And then running:

`make_ab_interim.py`

Once the two ".dat" files above have been produced, they can just be copied for use from:

```
$CAPTUTILS/zdata
```

To wherever either of the following scripts will be executed:

```
interim2cesm_se.py
interim2cesm_fv.py
```

For FV regridding, copy the following two files to wherever interim2cesm_fv.py will be executed:

```
cp $CAPTUTILS/zdata/erainterim_60lev_ab_full.dat <wherever>
cp $CAPTUTILS/zdata/erainterim_60lev_ab_half.dat <wherever>
```

Convert Interim state xml files to CESM/FV netcdf files using something like:

```
$CAPTUTILS/interim2cesm_fv.py \
   2009 1 1 0 \
   5 \
   $CAPTDATA/interim \
   UVTQPS-20090101-20090131_T255.xml \
   atm/cam/inic/fv/cami-chem_1990-01-01_0.9x1.25_L30_c080724.nc \
   atm/cam/topo/USGS-gtopo30_0.9x1.25_remap_c051027.nc
```

This script imports:

```
data_interim  # Provides data interface to Interim grib data;
              # note that data_interim imports:
              #     get_ab
filter_field  # Described in previous filtering section.
vertical_interpolation_code  # Described in a previous section.
other misc. modules
```

Required arguments are:

```
year, month, day, hour
ntimes                  # Number of 6 hour time intervals.
state_xmlfile_dir       # Must also contain the needed interim data
                        # ".ctl" and ".grib" files!
state_xmlfile_name
hybrid_file
topo_file
```

Optional arguments are:

```
-ft=$ | --file-tag=$
```

Note that INFILE_PREF in the script will be prepended to hybrid_file and topo_file, if their first character is not "/".

The following file will be produced:

```
interim_fv_2009010100_2009010200_TQUSVS.nc
```

For SE regridding, copy the following two files to wherever interim2cesm_se.py will be executed:

```
cp $CAPTUTILS/zdata/erainterim_60lev_ab_full.dat <wherever>
cp $CAPTUTILS/zdata/erainterim_60lev_ab_half.dat <wherever>
```

Convert Interim state xml files to CESM/SE netcdf files using something like (this version uses ESMF mapping files):

```
$CAPTUTILS/interim2cesm_se.py \
   2009 1 1 0 \
   5 \
   interimdatadir/ \
   UVTQPS-20090101-20090131_T255.xml \
   mapfiledir/ \
   map_interim_t255_to_ne30np4_bilin.nc \
   caminitialfiledir/camchemi_0012-01-01_ne30np4_L30_c110210.nc \
   topofiledir/USGS-gtopo30_ne30np4_20xdel2.nc
```

This script imports:

```
data_interim  # Provides data interface to Interim grib data.
```

```
vertical_interpolation_code  # Described in a previous section.
other misc. modules          # Note that no filtering is needed.
```

Required arguments are:

```
year, month, day, hour
ntimes                   # Number of 6 hour time intervals.
state_xmlfile_dir        # Must also contain the needed interim data
                         # ".ctl" and ".grib" files!
state_xmlfile_name
map_file_dir
map_file_name
hybrid_file
topo_file
```

Optional arguments are:

```
-ft=$ | --file-tag=$
```

The following file will be produced:

```
interim_se_20009010100_2009010200_TQUV.nc
```

4. E3SM model and simulations

Note that all the detailed information about E3SM and nudging code can be found in the following links:

4.1. General E3SM info

General web site for E3SM can be found at:
https://e3sm.org/model/

4.2. Create E3SM/EAM nudging functionality

Nudging functionality for E3SM can be found at:
https://github.com/E3SM-Project/E3SM/tree/archive/Sun_JAMES_2019_nudging/

A compact version on how to perform the E3SM normal AMIP-type and nudging simulations will be provided in the future update of this document.

5. Create initial conditions for climate model hindcast experiments

5.1. Create atmospheric initial conditions using model climatology

First way to generate atmospheric initial conditions is to use state variables from reanalysis and other necessary variables from a previous climatological run. Using this option, one needs to have initial files (".i" file) from a previous climatological run at the desired starting date(s). The file looks like "**casename.cam.i.2010-01-01-00000.nc**". A quick way to achieve this is to start an AMIP-type simulation and save the initial file(s) (".i"). This is achieved by adding **inithist='DAILY'** or other desired frequency ('NONE', '6-HOURLY', 'DAILY', 'MONTHLY',
'YEARLY', 'CAMIOP', 'ENDOFRUN') in **usr_nl_cam**. The default for inithist is 'YEARLY'. Once the file(s) is ready, use the **create_eam_ic_nonudge.py** to generate the atmospheric initial conditions using something like:

```
$CAPTUTILS/create_eam_ic_nonudge.py \
   input_file_name \
   era5datadir/UVTQPS-20000101-20000131_N480.xml \
   target_dir/ \
   year month day hour
```

These scripts currently have the following arguments:

```
case_id          #File name without the ".nc"
state_xml_file   #
targ_dir
year
month
day
hour
```

Go to the directory containing the restart files for the nudging spinup runs:

```
cd $STUDYDIR/base_capt/zsave/run0001/restarts
```

5.2. Create atmospheric initial conditions using a nudging simulation

Similar to previous section, the initial files (".i" file), however, is from a nudging simulation and the initial file(s) (".i") is saved during the process. Once the file(s) is ready, use the **create_eam_ic_nudge.py** to generate the atmospheric initial conditions using something like:

```
$CAPTUTILS/create_eam_ic_nudge.py \
   input_file_name \
   era5datadir/UVTQPS-20000101-20000131_N480.xml \
   target_dir/ \
   year month day hour
```

These scripts currently have the following arguments:

```
case_id          #File name without the ".nc"
state_xml_file   #ERA5 state data
targ_dir
year
month
day
```

```
hour
```

Note that this is the preferred method in obtaining better atmospheric initial conditions since the nudging simulation provides a better variable state closer to observations.

5.3. Create land initial conditions

The current recommend way to generate land initial conditions is to perform a land component only simulation forced with **precipitation**, **downward shortwave and longwave radiation**, **surface wind**, **temperature**, **humidity and pressure** from observations or reanalysis. This is achieved by creating an "**I**" case with the E3SM and perform the land-only simulations for a few forcing cycles for the land model to properly spin up. After that, one can use the restart files from the model run as the initial conditions. A restart file would look like "**casename.clm2.r.2010-01-01-00000.nc**".