Getting Started with

Bob Yantosca and Melissa Sulprizio

IGC10 Model Clinic

Monday, 06 June 2022

St. Louis, MO, USA

geos-chem-support@g.harvard.edu

Getting started with GEOS-Chem

- GEOS-Chem basics
  - wiki.geos-chem.org/GEOS-Chem_basics
- GEOS-Chem manuals
  - Manual.geos-chem.org – currently points to *Getting Started with GEOS-Chem* on wiki
  - New manual for 14.0.0 in prep – at geos-chem.readthedocs.io
  - Running GEOS-Chem Classic on the Amazon Cloud -- see cloud.geos-chem.org
- Requesting GEOS-Chem support
  - Use Github issues rather than email
  - Open an issue at https://github.com/geoschem/geos-chem/issues/new/choose

GEOS-Chem Classic Workflow

0. First-time setup: Install required libraries for GEOS-Chem Classic
   a. Not covered today, see geos-chem.readthedocs.io for more info
1. **Download the GEOS-Chem Classic source code**
2. **Create a run directory**
3. **Compile the GEOS-Chem Classic source code into an executable file**
4. **Configure GEOS-Chem Classic run-time simulation options**
5. **Download data needed by GEOS-Chem Classic with a "dry-run" simulation**
6. **Run GEOS-Chem Classic**
7. **Examine diagnostic outputs**

Downloading GEOS-Chem Classic

Downloading GEOS-Chem Classic

- Clone the latest GEOS-Chem Classic version as indicated on the front page of the GEOS-Chem wiki. Each version is documented with:
  - Unique DOI
  - Table of updates
  - Updated data directories (if applicable)
  - Benchmark results (for X.Y.0 versions only)

Downloading GEOS-Chem Classic

1) Clone the GCClassic "superproject" from Github:

```
$ git clone -b 14.0.0-rc.0 https://github.com/geoschem/GCClassic.git
$ cd GCClassic
$ git branch 14.0.0-rc.0
$ git checkout 14.0.0-rc.0
```

This will clone the top-level GEOS-Chem Classic "superproject" wrapper to your disk space and make a branch at the same point as the **14.0.0 release candidate** for you to work in. You can also view the revision history with: `$ gitk --all &`

- *NOTE: Normally we would always clone the **main** branch (which always contains the most recent version) and omit the -b option above. But at the time of this clinic 14.0.0 hasn't yet been released, so we will clone the 14.0.0 release candidate.*

Downloading GEOS-Chem Classic

2) Take a moment to look at the source code in the **`src/`** folder:

```
$ ls src/GEOS-Chem

$ ls src/HEMCO
```

The GEOS-Chem and HEMCO folders are empty. This is because we haven't fetched the code into them yet.

Downloading GEOS-Chem Classic

3. Fetch the GEOS-Chem and HEMCO source code:

```
$ git submodule update --init --recursive

Submodule 'docs/source/geos-chem-shared-docs' (https://github.com/geoschem/geos-chem-shared-docs.git)
registered for path 'docs/source/geos-chem-shared-docs'
Submodule 'src/GEOS-Chem' (https://github.com/geoschem/geos-chem.git) registered for path 'src/GEOS-Chem'
Submodule 'src/HEMCO' (https://github.com/geoschem/hemco.git) registered for path 'src/HEMCO'
Cloning into '/home/ubuntu/IGC10/GCClassic/docs/source/geos-chem-shared-docs'...
Cloning into '/home/ubuntu/IGC10/GCClassic/src/GEOS-Chem'...
Cloning into '/home/ubuntu/IGC10/GCClassic/src/HEMCO'...
Submodule path 'docs/source/geos-chem-shared-docs': checked out '228507857eb53740dacf4055ce9268aa8ccf520d'
Submodule path 'src/GEOS-Chem': checked out '7e51a0674aba638c8322fef493ac9251095e8cf4'
Submodule path 'src/HEMCO': checked out '4a66bae48f33e6dc22cda5ec9d4633192dee2f73'
Submodule 'docs/source/geos-chem-shared-docs' (https://github.com/geoschem/geos-chem-shared-docs.git)
registered for path 'src/HEMCO/docs/source/geos-chem-shared-docs'
Cloning into '/home/ubuntu/IGC10/GCClassic/src/HEMCO/docs/source/geos-chem-shared-docs'...
Submodule path 'src/HEMCO/docs/source/geos-chem-shared-docs': checked out
'645401baa35b6a6838b9bedede309a01a311517f'
```

Downloading GEOS-Chem Classic

4) Look again at the GEOS-Chem and HEMCO source code folders:

```
$ ls src/GEOS-Chem
APM/            CMakeScripts/  GeosRad/   History/     KPP/          ObsPack/   REVISIONS
AUTHORS.txt     GTMM/          GeosUtil/  ISORROPIA/   LICENSE.txt   PKUCPL/    run/
CMakeLists.txt  GeosCore/      Headers/   Interfaces/  NcdfUtil/     README.md  test/

$ ls src/HEMCO
AUTHORS.txt      CMakeScripts/    LICENSE.txt   SUPPORT.md   run/
CMakeLists.txt   CONTRIBUTING.md  README.md     docs/        src/
```

This now indicates that the GEOS-Chem and HEMCO codes have been pulled from Github. We may now proceed to the next step.

Creating a GEOS-Chem Classic run directory

Creating a GEOS-Chem Classic run directory

1) From the top-level **GCClassic** folder, issue the following commands:

```
$ cd run
$ ./createRunDir.sh
```

The **createRunDir.sh** script will ask us several questions about the run directory that is to be created.

Creating a GEOS-Chem Classic run directory

```
============================================================
GEOS-CHEM RUN DIRECTORY CREATION
============================================================

------------------------------------------------------------
Choose simulation type:
------------------------------------------------------------
   1. Full chemistry
   2. Aerosols only
   3. CH4
   4. CO2
   5. Hg
   6. POPs
   7. Tagged CH4
   8. Tagged CO
   9. Tagged O3
  10. TransportTracers
  11. Trace metals
3
```

2) The first menu asks us to specify which simulation to use.

For this demo, we'll pick the CH4 simulation (Option #3) since that simulation only has a single species and is fast to run.

- *NOTE: User inputs in RED*

Creating a GEOS-Chem Classic run directory

```
------------------------------------------------------------
Choose meteorology source:
------------------------------------------------------------
  1. MERRA-2 (Recommended)
  2. GEOS-FP
  3. GISS ModelE2.1 (GCAP 2.0)
1
```

3) The next menu asks us to specify a meteorological source for GEOS-Chem Classic.

MERRA-2 is a stable reanalysis product that has data from 1980 - present. This is the recommended option. The native resolution is 0.5^{o} x 0.625^{o}.

GEOS-FP is an operational product (i.e. it gets new science updates periodically), so there can be major changes after certain dates. The native resolution is 0.25^{o} x 0.3125^{o}.

GCAP2 is output from the GISS-2 GCM and is useful for long-term historical studies.

Creating a GEOS-Chem Classic run directory

```
------------------------------------------------------------
Choose horizontal resolution:
------------------------------------------------------------
  1. 4.0  x 5.0
  2. 2.0  x 2.5
  3. 0.5  x 0.625
1
```

```
------------------------------------------------------------
Choose horizontal grid domain:
------------------------------------------------------------
  1. Global
  2. Asia
  3. Europe
  4. North America
  5. Custom
```

4) The next menu prompts us to specify a horizontal grid resolution.

For this demo, we'll pick Option 1 (4^o x 5^o).

NOTE: *If 0.5^o x 0.625^o horizontal resolution is selected, a subsequent menu will ask us to choose either a global or nested-grid domain.*

CAVEAT: *Global 0.5^o x 0.625^o require a lot of computing power and disk space!*

Creating a GEOS-Chem Classic run directory

```
------------------------------------------------------------
Choose number of levels:
------------------------------------------------------------
  1. 72 (native)
  2. 47 (reduced)
1
```

5) The next menu prompts us to select the number of vertical levels in your simulation. We'll select 72 levels (Option #1) for our demo.

We recommend to always use 72 levels unless your simulation is very computationally-intensive, or if you are only focusing on the troposphere. In that case you might want to use 47 levels instead.

Creating a GEOS-Chem Classic run directory

```
------------------------------------------------------------
Enter path where the run directory will be created:
------------------------------------------------------------
~/IGC10
Expanding to: /home/ubuntu/IGC10
```

6) The next menu asks in which folder it should install the run directory.

For this demo, we'll install it into the **`IGC10`** folder in the home directory.

NOTE: If a relative path (**`~/IGC10`**) is given, **`createRunDir.sh`** will expand it to an absolute path (**`/home/ubuntu/IGC10`**).

Compiling GEOS-Chem Classic

Compiling GEOS-Chem Classic

1) Navigate to the build folder in the run directory and run **cmake**:

```
$ cd ~/IGC10/gc_4x5_merra2_ch4/build
$ cmake ../CodeDir -DRUNDIR=..
```

You can pass options to **cmake** with the form: **`-DOPTION=value`**

cmake will perform some basic error checks. If it finds that a required software package is missing, it will stop with an error. **cmake** will also print out the compile-time configuration options that have been selected (either by default or manually).

The new manual geos-chem.readthedocs.io contains extensive documentation on using **cmake** with GEOS-Chem.

Compiling GEOS-Chem Classic

2) After **cmake** finishes, compile GEOS-Chem with **make**:

```
$ make -j
…
[100%] Linking Fortran executable ../bin/gcclassic
[100%] Built target gcclassic
```

The **make** command compiles the GEOS-Chem source code into an executable file (**`build/bin/gcclassic`**).

The **`-j`** option tells make to compile as many files as possible simultaneously. This will reduce the overall compilation time.

Compiling GEOS-Chem Classic

3) After the executable has been built, install it to the run directory with:

```
$ make -j install
…
Install the project...
-- Install configuration: "Release"
-- Installing: /home/ubuntu/IGC10/gc_4x5_merra2_CH4/build_info/CMakeCache.txt
-- Installing: /home/ubuntu/IGC10/gc_4x5_merra2_CH4/build_info/summarize_build
-- Installing: /home/ubuntu/IGC10/gc_4x5_merra2_CH4/gcclassic
```

4) Navigate back to the run directory (which is one level higher than **`build/`**):

```
$ cd ..
```

Now we can proceed to the next step!

Configuring GEOS-Chem Classic run-time options

Configuring GEOS-Chem Classic

After compiling GEOS-Chem Classic, your run directory should look like this:

```
$ ls ~/IGC10/gc_4x5_merra2_CH4
CodeDir@                                 OutputDir/          gcclassic*
GEOSChem.Restart.20190101_0000z.nc4      README              geoschem_config.yml
GEOSChem.Restart.20190101_0100z.nc4      archiveRun.sh*      getRunInfo*
HEMCO_Config.rc                          build/              metrics.py*
HEMCO_Config.rc.gmao_metfields           build_info/         runScriptSamples@
HEMCO_Diagn.rc                           cleanRunDir.sh*     rundirConfig/
HEMCO_restart.201901010100.nc            download_data.py*   species_database.yml
HISTORY.rc                               download_data.yml
```

- Files highlighted in **yellow** are configuration files for GEOS-Chem Classic and HEMCO.
- Note: In 14.0.0, **input.geos** is replaced with **geoschem_config.yml**.

Configuring GEOS-Chem Classic

The **geoschem_config.yml** file (new in 14.0.0!) contains the main settings for GEOS-Chem, including simulation settings, grid settings, species list, etc. For our demo, we only need to change the end date of the simulation:

```
#============================================================================
# Simulation settings
#============================================================================
Simulation:
  name: CH4
  start_date: [20190101, 000000]
  end_date: [20190201, 000000]     <==== Change to [20190101, 010000] for 1-hour run
  root_data_dir: /home/ubuntu/ExtData
  met_field: MERRA2
  species_database_file: ./species_database.yml
  debug_printout: false
  use_gcclassic_timers: false
```

Configuring GEOS-Chem Classic

The **HEMCO_Config.rc** file specifies how HEMCO will read emissions and other data sets into GEOS-Chem Classic. For our demo, we must change the diagnostic frequency from **Monthly** to **End**.

```
###############################################################################
### BEGIN SECTION SETTINGS
###############################################################################
ROOT:                          /home/ubuntu/ExtData/HEMCO
METDIR:                        /home/ubuntu/ExtData/GEOS_4x5/MERRA2
GCAP2SCENARIO:                 not_used
Logfile:                       HEMCO.log
DiagnFile:                     HEMCO_Diagn.rc
DiagnPrefix:                   ./OutputDir/HEMCO_diagnostics
DiagnFreq:                     Monthly    <==== Change to End
Wildcard:                      *
Separator:                     /
Unit tolerance:                1
Negative values:               2
Only unitless scale factors:  false
Verbose:                       0
Warnings:                      1
```

Configuring GEOS-Chem Classic

The **`HEMCO_Diagn.rc`** file specifies how HEMCO will archive emissions diagnostics. For our demo, we do not need to make any modifications. To disable any diagnostic, add a **`#`** character before its name.

```
# Name                Spec ExtNr Cat Hier Dim OutUnit  LongName
EmisCH4_Total         CH4   -1   -1   -1   2   kg/m2/s  CH4_emissions_from_all_sectors
EmisCH4_Oil           CH4    0    1   -1   2   kg/m2/s  CH4_emissions_from_oil
EmisCH4_Gas           CH4    0    2   -1   2   kg/m2/s  CH4_emissions_from_gas
EmisCH4_Coal          CH4    0    3   -1   2   kg/m2/s  CH4_emissions_from_coal
EmisCH4_Livestock     CH4    0    4   -1   2   kg/m2/s  CH4_emissions_from_livestock
EmisCH4_Landfills     CH4    0    5   -1   2   kg/m2/s  CH4_emissions_from_landfills
EmisCH4_Wastewater    CH4    0    6   -1   2   kg/m2/s  CH4_emissions_from_wastewater
EmisCH4_Rice          CH4    0    7   -1   2   kg/m2/s  CH4_emissions_from_rice
EmisCH4_OtherAnth     CH4    0    8   -1   2   kg/m2/s  CH4_emissions_from_other_anthro…
EmisCH4_BiomassBurn   CH4    0    9   -1   2   kg/m2/s  CH4_emissions_from_biomass_burning
EmisCH4_Wetlands      CH4    0    10  -1   2   kg/m2/s  CH4_emissions_from_wetlands
EmisCH4_Seeps         CH4    0    11  -1   2   kg/m2/s  CH4_emissions_from_geological_seeps
EmisCH4_Lakes         CH4    0    12  -1   2   kg/m2/s  CH4_emissions_from_lakes
EmisCH4_Termites      CH4    0    13  -1   2   kg/m2/s  CH4_emissions_from_termites
EmisCH4_SoilAbsorb    CH4    0    14  -1   2   kg/m2/s  CH4_emissions_from_soil_absorption
```

Configuring GEOS-Chem Classic

The **HISTORY.rc** file specifies how GEOS-Chem Classic will archive diagnostics. To disable any diagnostic, add a **#** character before its name in the **COLLECTIONS** list.

```
#============================================================================
# %%%%% COLLECTION NAME DECLARATIONS %%%%%
#
# To enable a collection, remove the "#" character in front of its name. The
# Restart collection should always be turned on.
#
# NOTE: These are the "default" collections for GEOS-Chem, but you can create
# your own customized diagnostic collections as well.
#============================================================================
COLLECTIONS: 'Restart',
             'CH4','
             'Metrics',
             'SpeciesConc',
              #'Budget',
              #'CloudConvFlux',
              #'ConcAfterChem',
              #'LevelEdgeDiags',
              #'StateMet',
              #'BoundaryConditions',
```

Configuring GEOS-Chem Classic

For our demo, we need to edit the **`SpeciesConc.frequency`** and **`SpeciesConc.duration`** settings in **`HISTORY.rc`** to specify a 1-hour simulation. (By default it is set up for a 1-month simulation.)

```
#============================================================================
# %%%%% THE SpeciesConc COLLECTION %%%%%
#
# GEOS-Chem species concentrations (default = advected species)
#
# Available for all simulations
#============================================================================
  SpeciesConc.template:       '%y4%m2%d2_%h2%n2z.nc4',
  SpeciesConc.frequency:      00000100 000000    <==== change to 00000000 010000
  SpeciesConc.duration:       00000100 000000    <==== change to 00000000 010000
  SpeciesConc.mode:           'time-averaged'
  SpeciesConc.fields:         'SpeciesConc_?ADV?            ',
::
```

Configuring GEOS-Chem Classic

For our demo, we also need to edit the **`CH4.frequency`** and **`CH4.duration`** settings in **`HISTORY.rc`** to specify a 1-hour simulation. (By default it is set up for a 1-month simulation.)

```
#============================================================================
# %%%%% THE CH4 COLLECTION %%%%%
#
# Cloud convective flux (default = advected species)
#
# Available for all simulations
#============================================================================
  CH4.template:               '%y4%m2%d2_%h2%n2z.nc4',
  CH4.frequency:              00000100 000000    <==== change to 00000000 010000
  CH4.duration:               00000100 000000    <==== change to 00000000 010000
  CH4.mode:                   'time-averaged'
  CH4.fields:                 'OHconcAfterChem             ',
                              'LossCH4byClinTrop           ',
                              'LossCH4byOHinTrop           ',
                              'LossCH4inStrat              ',
::
```

Configuring GEOS-Chem Classic

For more information on editing GEOS-Chem Classic configuration files, see:

- The current GEOS-Chem Classic manual (Getting Started with GEOS-Chem, on wiki)
- The new GEOS-Chem Classic manual (geos-chem.readthedocs.io)
- Guide to GEOS-Chem History Diagnostics (on wiki)
- The HEMCO manual (hemco.readthedocs.io)

NOTE: *We are currently migrating the manual content from the GEOS-Chem Wiki to the new ReadTheDocs site. At the time of this writing the migration is not yet complete. We hope to have the ReadTheDocs site fully updated by the official release of GEOS-Chem Classic 14.0.0.*

Downloading data with a “dry-run” simulation

Downloading data with a dry-run simulation

- A “dry-run” is a is a **GEOS-Chem Classic** simulation that steps through time, but does not perform computations or read data files from disk.
- Instead, a dry-run simulation **prints a list of all data files that a regular GEOS-Chem simulation would have read.**
- The dry-run output also denotes **whether each data file was found on disk, or if it is missing**.
- This output can be fed to a **script which will download the missing data files** to your computer system.
- See geos-chem.readthedocs.io for detailed information about the dry-run.

Downloading data with a dry-run simulation

- **NOTE:** If there is a large GEOS-Chem user group at your institution, someone (i.e. a sysadmin or another user) may have already downloaded GEOS-Chem input data to a shared disk on your computer cluster.
    - In this case you won't need to do the dry-run simulation as the data is already present. You can just point to the shared disk in the **`geoschem_config.yml`** file.
    - If you are not sure, ask someone in your group.

Downloading data with a dry-run simulation

1) Navigate to the run directory:

```
$ cd ~/IGC10/gc_4x5_merra2_CH4
```

2) Run GEOS-Chem Classic with the **`--dryrun`** option and pipe the output to a log file:

```
$ ./gcclassic --dryrun > log.dryrun
```

Downloading data with a dry-run simulation

3) Take a look at the **`log.dryrun`** file:

```
$ less log.dryrun
…

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!! GEOS-CHEM IS IN DRY-RUN MODE!
!!!
!!! You will NOT get output for this run!
!!! Use this command to validate a GEOS-Chem run configuration:
!!!   ./gcclassic --dryrun > log
!!!
!!! REMOVE THE --dryrun ARGUMENT FROM THE COMMAND LINE
!!! BEFORE RUNNING A GEOS-Chem PRODUCTION SIMULATION!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!! Start Date      : 20190101 000000
!!! End Date        : 20190101 010000
!!! Simulation      : CH4
!!! Meteorology     : MERRA2
!!! Grid Resolution  : 4.0x5.0
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

The **`log.dryrun`** file looks like a log file from a regular GEOS-Chem Classic simulation, except for:

(a) It contains warnings like this posted at the start & end of the log file;

… and …

Downloading data with a dry-run simulation

3) Take a look at the **`log.dryrun`** file:

```
$ less log.dryrun
…
HEMCO (INIT): Opening ./HEMCO_Config.rc
HEMCO (INIT): Opening ./HEMCO_Config.rc.gmao_metfields
HEMCO (INIT): Opening ./HEMCO_Diagn.rc
HEMCO: REQUIRED FILE NOT FOUND
/home/ubuntu/ExtData/HEMCO/CH4/v2017-10/GEPA/GEPA_Annual.nc
HEMCO: REQUIRED FILE NOT FOUND
/home/ubuntu/ExtData/HEMCO/CH4/v2020-09/Scarpelli_Mexico/MEX_Tia2
020_oil_2015.nc
HEMCO: REQUIRED FILE NOT FOUND
/home/ubuntu/ExtData/HEMCO/CH4/v2020-09/Scarpelli_Mexico/MEX_Tia2
020_gas_2015.nc
…
```

(b) It contains a list of files needed for the simulation:

Opening means that the file is found on disk.

REQUIRED FILE NOT FOUND means that the file needs to be downloaded.

This information is used by **`download_data.py`**.

Downloading data with a dry-run simulation

4) Run the **`download_data.py`** script to download data to your disk:

- We recommend downloading from the data portal at Washington University in St. Louis (http://geoschemdata.wustl.edu):

```
$ ./download_data.py log.dryrun --washu
```

- But if you are using GEOS-Chem Classic on the Amazon Web Services (AWS) Cloud, then downloading from the **`s3://gcgrid`** data portal instead might be faster:

```
$ ./download_data.py log.dryrun --amazon
```

Caveat: Downloading data outside of the AWS cloud will incur egress fees.

Also note: GCAP2 meteorology is stored at the University of Rochester data portal (use `--ur`)

Downloading data with a dry-run simulation

5) If the data download process stops (e.g. broken connection), then

- Do a new dry-run simulation, and pipe the output to a differently-named log file.
- Use the new log file with **`download_data.py`** to start a new download process.

```
$ ./gcclassic --dryrun > log.dryrun2

$ ./download_data.py log.dryrun2 --washu
```

The new dry-run will pick up only the files that weren't already downloaded by the first (interrupted) dry-run. This will save time!

Downloading data with a dry-run simulation

6) Take a look at the **`log.dryrun.unique`** file:

```
$ less log.dryrun.unique
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!! LIST OF (UNIQUE) FILES REQUIRED FOR THE SIMULATION
!!! Start Date    : 20190101 000000
!!! End Date      : 20190101 010000
!!! Simulation    : CH4
!!! Meteorology   : MERRA2
!!! Grid Resolution  : 4.0x5.0
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
./GEOSChem.Restart.20190101_0000z.nc4 -->
/home/ubuntu/ExtData/GEOSCHEM_RESTARTS/v2020-02/initial_GEOSChem_rst.2x25_CH4.nc
./HEMCO_Config.rc
./HEMCO_Config.rc.gmao_metfields
./HEMCO_Diagn.rc
./HISTORY.rc
./geoschem_config.yml
/home/ubuntu/ExtData/GEOS_4x5/MERRA2/2015/01/MERRA2.20150101.CN.4x5.nc4
/home/ubuntu/ExtData/GEOS_4x5/MERRA2/2019/01/MERRA2.20190101.A1.4x5.nc4 …etc…
```

This log file is created by **`download_data.py`**.

It is a list of data files needed for the simulation (with duplicate instances removed).

This can be useful for documentation purposes.

Running GEOS-Chem Classic

Running GEOS-Chem Classic

Depending on your system, there are two ways you can run GEOS-Chem Classic:

1. **Interactive job** (from a terminal)
2. **Batch job** (with a scheduler such as SLURM, LSF, PBS, etc.)

If you are using a shared computer cluster, chances are that you will need to run GEOS-Chem Classic as a **batch job**. Ask your IT staff for more information.

If you have a dedicated computer system, or are logged into an Amazon Web Services EC2 cloud instance, then you can run GEOS-Chem Classic **interactively**.

In either case, you will need a **run script**. A sample batch job run script has been provided for you in the GEOS-Chem run directory (`runScriptSamples/geoschem.run`).

```
#!/bin/bash

#SBATCH -c 8
#SBATCH -N 1
#SBATCH -t 0-12:00
#SBATCH -p MYQUEUE
#SBATCH --mem=15000
#SBATCH --mail-type=END

############################################################
### Sample GEOS-Chem run script for SLURM
### You can increase the number of cores with -c and memory wit
### particularly if you are running at very fine resolution (e.
############################################################

# Set the proper # of threads for OpenMP
# SLURM_CPUS_PER_TASK ensures this matches the number you set w
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

# Run GEOS_Chem.  The "time" command will return CPU and wall t
# Stdout and stderr will be directed to the "GC.log" log file
# (you can change the log file name below if you wish)
srun -c $OMP_NUM_THREADS time -p ./gcclassic > GC.log 2>&1

# Exit normally
exit 0
```

Sample GEOS-Chem run script for batch job submission (geoschem.run)

Red: SLURM-specific commands. These will be ignored if your system doesn't have a scheduler (such as on AWS cloud).

In this example, we request from the scheduler:

- **8 cores**
- **1 node**
- **12 hours run time**
- **Execution in queue MYQUEUE**
- **15 GB memory for the run**
- **Email you upon job completion**

The rest of the script will run GEOS-Chem and pipe its output to a log file.

Submit job to SLURM scheduler:

```
$ sbatch geoschem.run
```

```
#!/bin/bash

################################################################################
### Sample GEOS-Chem run script for interactive submission
################################################################################

# Set the proper # of threads for OpenMP
# In this case, use 24 cores
export OMP_NUM_THREADS=24

# Run GEOS_Chem.  The "time" command will return CPU and wall
# Stdout and stderr will be directed to the "GC.log" log file
# (you can change the log file name below if you wish)
time -p ./gcclassic > GC.log 2>&1

# Exit normally
exit 0
```

Sample GEOS-Chem run script for interactive submission (geoschem-int.run)

In this example run script example, we have stripped out all of the SLURM commands.

NOTE: We must now manually set the **`OMP_NUM_THREADS`** environment variable to specify the number of cores that GEOS-Chem Classic should use.

Run interactively:

```
$ ./geoschem-int.run
```

Running GEOS-Chem Classic

When GEOS-Chem Classic finishes, your run directory should look like this:

```
$ ls ~/IGC10/gc_4x5_merra2_CH4
CodeDir@                                HISTORY.rc          gcclassic*
GC.log                                  OutputDir/          geoschem_config.yml
GEOSChem.Restart.20190101_0000z.nc4     README              getRunInfo*
GEOSChem.Restart.20190101_0100z.nc4     archiveRun.sh*      log.dryrun
HEMCO.log                               build/              log.dryrun.unique
HEMCO_Config.rc                         build_info/         metrics.py*
HEMCO_Config.rc.gmao_metfields          cleanRunDir.sh*     runScriptSamples@
HEMCO_Diagn.rc                          download_data.py*   rundirConfig/
HEMCO_restart.201901010100.nc           download_data.yml   species_database.yml
```

- Files highlighted in **yellow** are log files produced by GEOS-Chem Classic and HEMCO.
- Files highlighted in **red** are restart files produced by GEOS-Chem Classic and HEMCO.
- GEOS-Chem Classic and HEMCO create diagnostic output files in the **OutputDir/** folder.

Running GEOS-Chem Classic

A listing of the **`OutputDir/`** folder in the run directory should look like this:

```
$ ls -1 ~/IGC10/gc_4x5_merra2_CH4/OutputDir
GEOSChem.CH4.20190101_0000z.nc4
GEOSChem.Metrics.20190101_0000z.nc4
GEOSChem.SpeciesConc.20190101_0000z.nc4
HEMCO_diagnostics.201901010000.nc
```

A separate file for each of the diagnostic collection (more later) that was activated in the **`HISTORY.rc`** configuration file (one level up) has been created. A separate file containing HEMCO emission diagnostics has also been placed in **`OutputDir/`**.

Analyzing output from GEOS-Chem Classic

Analyzing output from GEOS-Chem Classic

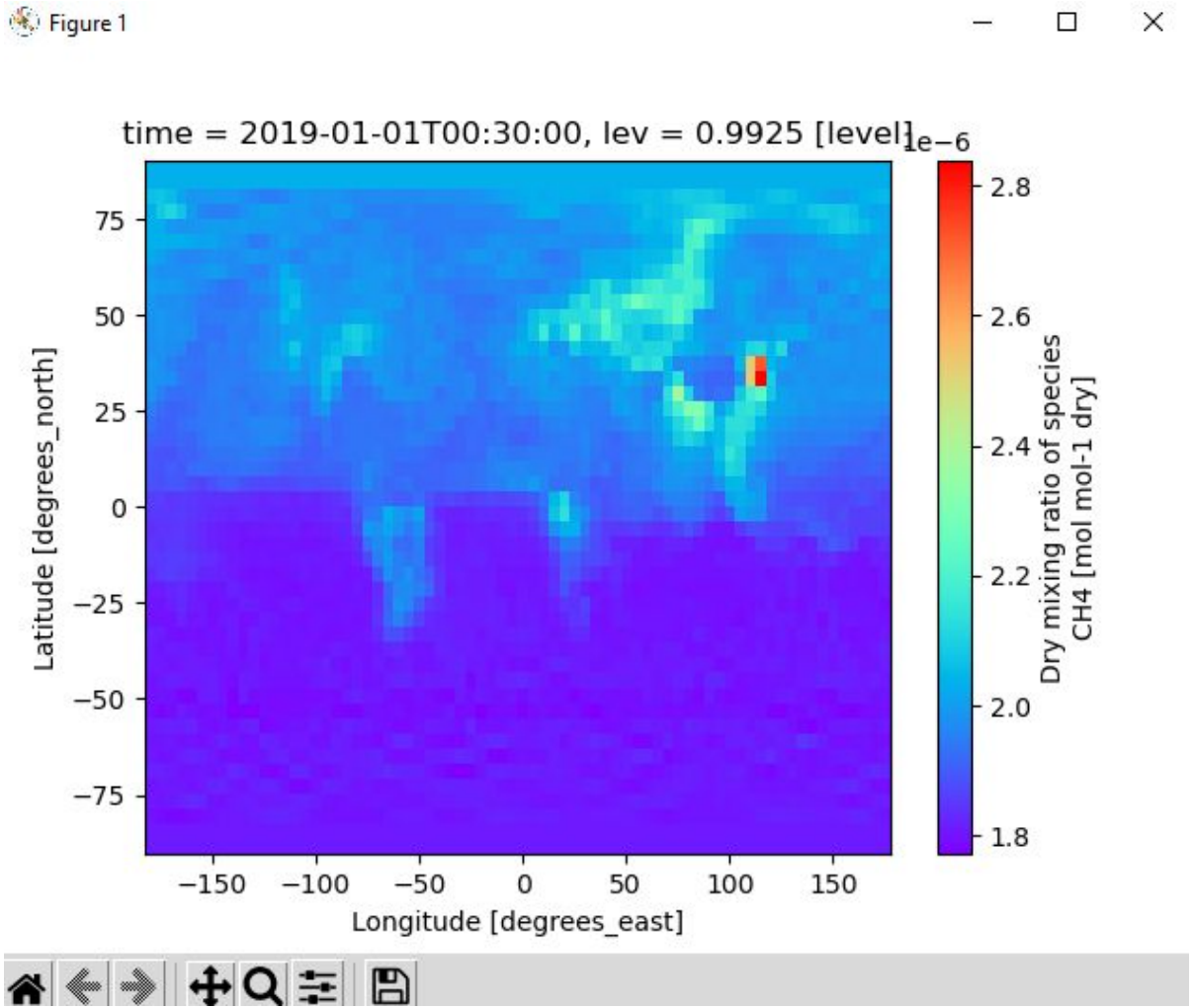
- Most GEOS-Chem Classic and HEMCO diagnostic quantities are archived to COARDS-compliant netCDF files.
- We can use Python to visualize and analyze the data contained in these diagnostic files.
- Python is open-source and easily usable on the Amazon Web Services Cloud (i.e. no need to worry about proprietary software and license fees).

Analyzing output from GEOS-Chem Classic

Here is a summary of GEOS-Chem Classic's **diagnostic outputs**. For more information please see the new GEOS-Chem manual at geos-chem.readthedocs.io.

Diagnostic	***Types of output***	***Settings are specified in:***
netCDF diagnostics (aka "History")	Gridded instantaneous or time-averaged output for species conc's and other quantities *(netCDF)*	The HISTORY.rc file *(included in the run directory)*
HEMCO diagnostics	Gridded instantaneous or time-averaged output for emissions *(netCDF)*	The HEMCO_Diagn.rc file *(included in the run directory)*
ObsPack diagnostics	Point output of species concentrations & met fields at surface stations *(netCDF)*	Data files adhering to the NOAA ObsPack format *(not included in the run directory)*
Planeflight diagnostics	Point output of species concentrations, met fields, J-values, and other various quantities along flight tracks *(text)*	The Planeflight.dat file *(not included in the run directory)*

```
#!/usr/bin/env python

### quickplot.py: Read and plot GEOS-Chem output
### using only minimal Python commands
import xarray as xr
import matplotlib.pyplot as plt

# Read the GEOS-Chem species conc's into an xarray Dataset object
ds = xr.open_dataset("OutputDir/GEOSChem.SpeciesConc.20190101_0000z.nc4")

# Create an xarray DataArray object (which is a variable in a Dataset)
# containing the CH4 concentrations for the surface level (lev=0)
# and the first timestamp in the file (time=0)
dr = ds["SpeciesConc_CH4"].isel(time=0, lev=0)

# Plot the data using the matplotlib default "rainbow" color table
xr.plot.imshow(dr, cmap="rainbow")
plt.show()
```



Quick-and-dirty plot made from the **`quickplot.py`** script on the previous slide.

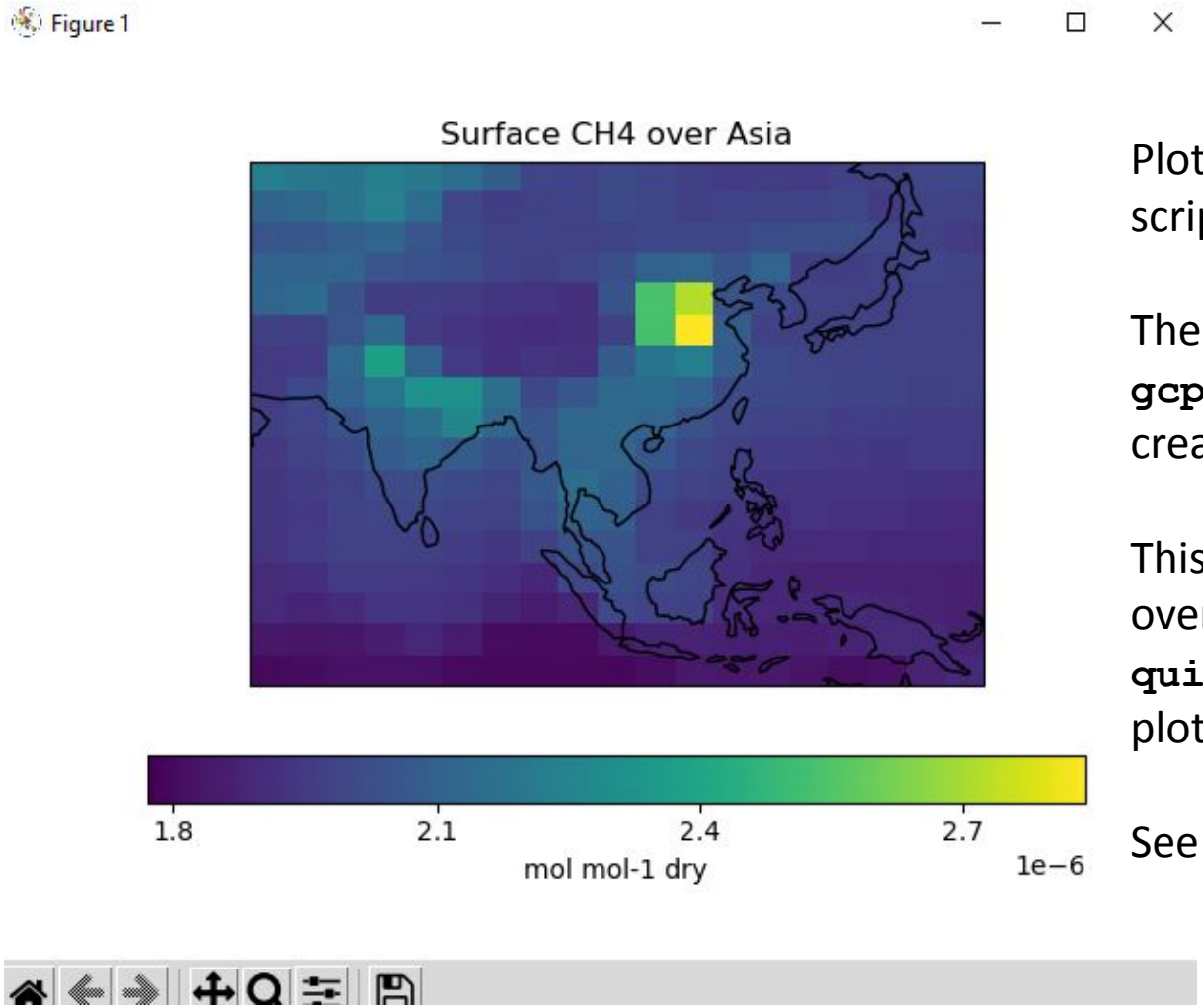
Useful for sanity checks!

Of course, we can refine the plotting output to add country boundaries, title, etc.

GCPy: Python toolkit for GEOS-Chem

- Install with conda
  - See https://gcpy.readthedocs.io for details
- Contains:
  - Several useful general-purpose routines and example scripts
  - Routines and scripts for creating plots & tables for GEOS-Chem benchmark simulations

```
#!/usr/bin/env python

# singleplot.py: This example is from the GCPy manual page:
# https://gcpy.readthedocs.io/en/stable/Single_panel.html

# Imports
import xarray as xr
import matplotlib.pyplot as plt
import gcpy.plot as gcplot

# Read CH4 data from GEOS-Chem
ds = xr.open_dataset("OutputDir/GEOSChem.SpeciesConc.20190101_0000z.nc4")

# Get CH4 at the surface and for the 1st time index in the file
sfc_ch4 = ds["SpeciesConc_CH4"].isel(lev=0, time=0)

# Create a plot of CH4 over the AS nested grid region used in GEOS-Chem Classic
gcplot.single_panel(
     sfc_ch4,
     title="Surface CH4 over Asia",
     comap = plt.cm.viridis,           # Use Viridis color map
     log_color_scale=False,            # Use a linear color scale
     extent=[60, 150, -11, 55]         # [min lon, max lon, min lat, max lat]
)
plt.show()
```



Plot created by the **`singleplot.py`** script on the previous slide.

The **`singleplot.py`** example calls **`gcpy.plot.single_panel`** to create the plot.

This gives us much more control over the plotting options than the **`quickplot.py`** "quick and dirty" plotting example.

See gcpy.readthedocs.io

Debugging GEOS-Chem

Debugging GEOS-Chem

- Check GEOS-Chem and HEMCO log files for error messages
- Build GEOS-Chem Classic with debugging flags and run a short simulation:

```
$ cd build
$ cmake ../CodeDir -DRUNDIR=.. -DCMAKE_BUILD_TYPE=Debug
$ make -j
$ make -j install
```

- Add print statements at strategic locations in your code:
  - Print values of variables and arrays at a single grid box.
  - Print the sum of an array -- Use the Fortran **`SUM()`** function.
  - Print array min and max values -- Use **`MINVAL()`** and **`MAXVAL()`** functions.
- For more tips, see: http://wiki.geos-chem.org/Debugging_GEOS-Chem

Submitting help requests to the GCST

Open a new Github issue at https://github.com/geoschem/geos-chem/issues/new/choose:

Items to include	***Description***
A description of the issue	Describe how and when the run failed, which simulation and options you used.
`geoschem_config.yml` file	Contains input options for your simulation.
`HEMCO_Config.rc` file	Contains emissions options for your simulation.
GEOS-Chem log file	Contains information printed out by your GEOS-Chem simulation. This file should include a detailed error message if your run dies with an error. TIP: Turn on the debug print option in **`geoschem_config.yml`** for more debug output.
`HEMCO.log` file	Contains information about how emissions, met fields, and other input data are read and processed by HEMCO. Will contain an error message if your simulation dies in HEMCO. TIP: Set **`Warnings`** and **`Verbose`** to 3 for more debug output.
AMI ID and instance info	How your AWS cloud computing instance was initialized (if applicable)

Questions?

GCST Help Desk during IGC10

T/W/Thu 1-3 pm

Hillman 50 or Hillman 70 (TBA)

Reference Section: GEOS-Chem diagnostics

netCDF diagnostics (aka “History”)

- Diagnostic outputs are divided into several **COLLECTIONS**, which are specified in the HISTORY.rc file in your GEOS-Chem run directory.
- When you create a GEOS-Chem run directory, the HISTORY.rc file there will have several pre-defined collections. But you can add your own as well.
- Each collection saves a specific type of diagnostic output. You can pick:
  - Archival frequency and type
  - File names (and destination folder)
  - Quantities that will be archived

```
EXPID:   ./OutputDir/GEOSChem
# Turn off a collection by placing a # character
# in front of its name in the COLLECTIONS list:
COLLECTIONS: 'inst',
             'avg6hr',
::
  inst.template:        '%y4%m2%d2_%h2%n2z.nc4',
  inst.frequency:        010000,
  inst.duration:         240000,
  inst.mode:            'instantaneous',
  inst.fields:          'SpeciesConc_NO',
                        'SpeciesConc_O3,
                        'SpeciesConc_PAN',
                        'SpeciesConc_CO',
                        ... ETC ...
::
  avg6hr.template:      '%y4%m2%d2_%h2%n2z.nc4',
  avg6hr.frequency:     002000,
  avg6hr.duration:      240000,
  avg6hr.mode:          'time-averaged',
  avg6hr.fields:        'Met_U10M',
                        'Met_T',
                        'SpeciesConc_CO',    ::
```

The HISTORY.rc file

(NOTE: This is an idealized example)

ORANGE: output dir + file prefix

Two COLLECTIONS are defined:

- inst (hourly instantaneous output)
- avg6hr (6-hr time-averaged output)

Each COLLECTION contains several diagnostic quantities (aka FIELDS):

- Blue text = slices from array `State_Diag%SpeciesConc`
- Magenta text = met fields stored in State_Met

Double colons :: are separators

Example 1: Collection w/ instantaneous output

```
inst.template:    '%y4%m2%d2_%h2%n2z.nc4',
inst.frequency:   010000,
inst.duration:    240000,
inst.mode:        'instantaneous',
inst.fields:      'SpeciesConc_NO',
                  'SpeciesConc_O3',
                  'SpeciesConc_PAN',
                  ... ETC ...
::
```

Instantaneous output
Defined by the **"mode"** tag.

Frequency
Determines how often output is written to disk.

Duration
Determines how often a new file is written.

Result of this example:
Data is updated and saved out to disk each hour.

A new file is created each day.

Each file will have 24 timestamps (hourly data).

Example 2: Collection w/ time-averaged output

```
avg6hr.template:      '%y4%m2%d2_%h2%n2z.nc4',
avg6hr.frequency:     060000,
avg6hr.duration:      240000,
avg6hr.mode:          'time-averaged',
avg6hr.fields:        'Met_U10M',
                      'Met_T',
                      'SpeciesConc_CO',
::
```

Time-averaged
Defined by the **"mode"** tag.

Frequency
Determines the averaging period for the data.
Data will be updated every dynamic "heartbeat" timestep, and a counter of updates will be incremented.

Duration:
Determines how often a new file is written.

Result of this example:
Fields are updated every dynamic timestep, and averaged into 6-hour intervals.

A new file is written each day.

Each file will have 4 timestamps (6-hr averages).

Example 3: Collection w/ monthly-mean output

```
avg6hr.template:      '%y4%m2%d2_%h2%n2z.nc4',
avg6hr.frequency:     00000100 000000,
avg6hr.duration:      00010000 000000,
avg6hr.mode:          'time-averaged',
avg6hr.fields:        'Met_U10M',          'GIGCchem',
                      'Met_T',             'GIGCchem',
                      'SpeciesConc_CO',    'GIGCchem'

::
```

Two sets of digits are interpreted as **YYYYMMDD hhmmss.**

One set of digits is interpreted as **hhmmss**.

The **Frequency** and **Duration** intervals may also be specified for longer periods, such as months or years.

For long simulations, we *highly* recommend saving monthly output. This will also allow you to restart your simulation from the last archived month, if it should die unexpectedly.

Result of this example: Fields will be updated every dynamic timestep, and averaged into monthly intervals. A new file will be created once per year.

Example 4: Using wildcards in collections

```
avg6hr.template:      '%y4%m2%d2.nc4_%h2%n2z.nc4',
avg6hr.frequency:     00000100 000000,
avg6hr.duration:      00010000 000000,
avg6hr.mode:          'time-averaged',
avg6hr.fields:        'Met_U10M',
                      'Met_T',
                      'SpeciesConc_?ADV?',
::
```

You can also specify wildcards for species names. This will prevent you from having to list several species individually. (Currently in GEOS-Chem "Classic" only)

?ADV?	= all advected species	**?PHO?**	= all photolysis species
?AER?	= all aerosol species	**?KPP?**	= all species in KPP mechanism
?GAS?	= all gas-phase species	**?VAR?**	= all active KPP species
?DRY?	= all drydep species	**?FIX?**	= all inactive KPP species
?WET?	= all wetdep species	**?ALL?**	= all species

HEMCO diagnostics

- The HEMCO emissions component archives its own emission diagnostics.
- Diagnostic options are specified in the HEMCO_Diagn.rc file.
    - Each GC run directory that you create will have a **`HEMCO_Diagn.rc`** file appropriate for that simulation.
    - You shouldn't have to update the **`HEMCO_Diagn.rc`** file unless you are adding/removing emissions from the default configuration.
- HEMCO diagnostic output will be sent to:
  `OutputDir/HEMCO_diagnostics*.nc`

The HEMCO_Diagn.rc file. The Extension, Category, Hierarchy values correspond to the HEMCO_Config.rc file.

-1 indicates a sum over all Extensions, Categories, or Hierarchies.

```
###############################################################################
#####   ACET emissions                                                    #####
###############################################################################
EmisACET_Total      ACET    -1     -1   -1    3   kg/m2/s  ACET_emission_flux_from_all_sectors
EmisACET_Anthro     ACET    0      1    -1    3   kg/m2/s  ACET_emission_flux_from_anthropogenic
EmisACET_BioBurn    ACET    111    -1   -1    2   kg/m2/s  ACET_emission_flux_from_biomass_burning
EmisACET_Biofuel    ACET    0      2    -1    2   kg/m2/s  ACET_emission_flux_from_biofuel
EmisACET_Biogenic   ACET    108    -1   -1    2   kg/m2/s  ACET_emission_flux_from_biogenic_sources
EmisACET_Ocean      ACET    101    -1   -1    2   kg/m2/s  ACET_emission_flux_from_ocean

###############################################################################
#####  ALD2 emissions                                                     #####
###############################################################################
EmisALD2_Total      ALD2    -1     -1   -1    3   kg/m2/s  ALD2_emission_flux_from_all_sectors
EmisALD2_Anthro     ALD2    0      1    -1    3   kg/m2/s  ALD2_emission_flux_from_anthropogenic
EmisALD2_BioBurn    ALD2    111    -1   -1    2   kg/m2/s  ALD2_emission_flux_from_biomass_burning
EmisALD2_Biofuel    ALD2    0      2    -1    2   kg/m2/s  ALD2_emission_flux_from_biofuel
EmisALD2_Biogenic   ALD2    108    -1   -1    2   kg/m2/s  ALD2_emission_flux_from_biogenic_sources
EmisALD2_Ocean      ALD2    101    -1   -1    2   kg/m2/s  ALD2_emission_flux_from_ocean
EmisALD2_PlantDecay ALD2    0      4    -1    2   kg/m2/s  ALD2_emission_flux_from_decaying_plants
EmisALD2_Ship       ALD2    0      10   -1    2   kg/m2/s  ALD2_emission_flux_from_ships
```