User guide for the GEOS-Chem adjoint

Overall information on the GEOS-Chem adjoint model:

- The current GEOS-Chem adjoint code is based upon model version v9-02
- There is a brief quick-start guide for the adjoint available online here: http://wiki.seas.harvard.edu/geos-chem/index.php/Quick_Start_Guide .
- Link to the guide for this forward model version: http://acmg.seas.harvard.edu/geos/doc/archive/man.v9-02/
- You can download the GEOS-Chem adjoint model from a GitLab repository from the University of Colorado. However, the code here is customized for CO2 simulations and is customized for geostatistical inverse modeling.
- Note that the included code will run both forward GEOS-Chem simulations and adjoint simulations. The code will first run a forward simulation and then automatically run an adjoint simulation.

SETTING FILE PATHS AND RUN OPTIONS

Important files for setting run options:

- Files in the run directory:
    - input.geos: Set options for the forward model. For example, you can set the run directories, path to the meteorology files. Set information for the emissions in the model runs. Below, I've listed several really important menu items that you'll need to modify:
        - *Input restart file*: the restart file is in binary punch format. We have restart files based on NOAA CarbonTracker for Sept 1, 2014. In the future, we could change the GEOS-Chem code to read in a netcdf restart file (though I haven't done that yet).
        - *OCO2 obs directory*: Set the path to the OCO-2 observations (more on that later). This line is not used if you run an inverse model that only uses in situ observations. Note that this menu option is not in the standard downloadable version of the GEOS-Chem adjoint.
        - *Flask obs directory*: Set the path to the Obspack in situ observations. This line is not used if you run an inverse model that only uses satellite observations. Note that this menu option is not in the standard downloadable version of the GEOS-Chem adjoint.
        - *Emissions file path*: Path to CO2 fluxes for the inverse model. Note that this version of GEOS-Chem only reads in a single, daily flux file for all CO_2 source types. i.e., we don't use different flux files for different source types. The inverse model (i.e., Matlab code) will generate fluxes in this directory, so don't worry about populating this directory with flux files. Just make sure that the "fluxpath" in inversion_run.m matches the path here. Also note that this menu option is not in the standard downloadable version of the GEOS-Chem adjoint. ***If you are running the inverse model, make sure that this path is the same as fluxpath in the script inversion_run.m.***
    - input.gcadj: Set options for the adjoint run. In this file, you can turn off the adjoint (and only run the forward model). You can also run the adjoint in different modes (e.g., run sensitivities only, run inverse model, etc.). Note: if you turn off the adjoint model and try to run only the forward model, GEOS-Chem will crash and produce a segmentation fault on MARCC. I don't know why this error happens. However, if you leave the adjoint model turned on, the forward model will run just fine without producing a segmentation fault. Hence, for the moment, I recommend leaving the adjoint turned on even if you only want to run forward model simulations. Here are some important menu options to look for in this

file:

        - *Do adjoint run   LADJ:* Set this parameter to true for running the inverse model.
        - *Inverse problem   L4DVAR:* Set this parameter to true for running the inverse model.
        - *Include a priori term APSRC:* We set this parameter to false for the inverse model. This parameter determines whether the prior flux term of the cost function and gradient are included in Fortran calculations. In our setup, we calculate these terms in the Matlab code (not in the Fortran code), so we set this term to false.
        - *Number emis time group MMSCL:* Set the number of time periods in the inverse model. This term determines how the gradient is split into different time periods. Note that for the setup here, we estimate fluxes at a daily time resolution. Hence, this term should be equal to the number of days in the inverse modeling simulation. ***Don't forget to set this parameter equal to the number of days in your inverse model***. If you don't, the inverse model will produce weird results (e.g., weird cost function values) but won't necessarily produce an error.
    - run: This file executes the GEOS-Chem adjoint.
        - There are a few paths that need to be specified in this file that point GEOS-Chem to the run directory. You may also need to edit this file to specify your computing environment (e.g., paths to the netcdf libraries and paths to your run directory).
        - This script also checks to make sure that GEOS-Chem produced the outputs that it's supposed to produce. The script will print a message on screen declaring whether the run completed successfully.
        - You can execute the run file directly if you want to run GEOS-Chem but not run the full inverse model. For example, you might want to estimate atmospheric mixing ratios using a flux estimate and compute OCO-2 or flask model-data residuals. To accomplish this task, you can simply execute the run directory to launch GEOS-Chem forward and adjoint runs. The forward model will estimate atmospheric mixing ratios while the adjoint will compute the model-data comparisons.
    - inversion_run.sh: You'll submit this script to the queuing system on your computer cluster to launch the inverse model. Change the slurm options and the file paths for your particular computing system.
    - inversion_run.m: You'll need to modify this file and set file paths that point to your inverse modeling inputs. Note that inversion_run.m will run the inverse model with ocean fluxes held constant while inversion_run_withocean.m will optimize both land and ocean fluxes.
- Files in the fortran code directory (Make sure to set these options before compiling GEOS-Chem.):
    - define.h: Set the type of meteorology, the resolution of the simulations, and the compiler that will be used for the model. By default, this file is set to use “MERRA2” meteorology at a 4x5 spatial resolution.
    - define_adj.h: Tell GEOS-Chem which observations you want to use for the adjoint model. At the moment, you can use OCO-2 observations (denoted as “GOSAT” in define_adj.h), in situ CO2 observations, or both. Simply uncomment the lines corresponding to the observations you want to use.

Important notes on specifying the fluxes:

- The current GEOS-Chem configuration will read in fluxes with a daily time resolution.
- *If you're running the full inverse model*: You need to create the X matrix in the Matlab code (see inversion_run.m). However, you do not need to create flux input files for GEOS-Chem; the inverse model will automatically create flux input files for you.

- *If you are running GEOS-Chem forward model simulations (and not the inverse model)*: You'll need to create daily CO2 flux files (that includes all desired source types) and move those files to the flux directory specified in the input.geos file.
- The fluxes should be in netcdf files, and the files should use the following naming convention: CO2.daily.geos.4x5.YYYY.MM.DD.nc
- The CO2 flux variable in the netcdf file should be titled "CO2_flux"

COMPILING GOES-CHEM

OBSOLETE How to compile the adjoint on MARCC/Bluecrab:

- The GEOS-Chem code is not automatically compiled when you launch the inverse model. Rather, you need to compile the code yourself and copy the GEOS-Chem executable to the run directory.
- I've created an example code directory and run directory on MARCC.
- Path to the example directories: ~/data/smiller/geoschem_adjoint_co2/
- Note that we had to make several modifications to the GEOS-Chem code to get it to compile on MARCC (notably, to properly link GEOS-Chem with the netcdf libraries). We had to modify the Makefile and create a new file called "mx" (described below). Hence, I don't recommend downloading GEOS-Chem from the Gitlab repository and trying to compile the generic version of the code. Rather, I would use the example directory above.

Run the following code to compile the adjoint on MARCC:
[cd into the fortran code directory on MARCC]

```
ml intel/16.0
ml cmake
ml geos/9.02
export ROOT_LIBRARY_DIR=/software/apps/geos/netcdf/4.3.0/intel/16.0
export ROOT_LIBRARY_DIRF=/software/apps/geos/netcdf-fortran/4.2.0/intel/16.0
export GC_INCLUDE="$ROOT_LIBRARY_DIR/include"
export GC_F_INCLUDE="$ROOT_LIBRARY_DIRF/include"
export GC_F_BIN="$ROOT_LIBRARY_DIRF/bin"
export GC_BIN="$ROOT_LIBRARY_DIR/bin"
export
LD_LIBRARY_PATH=/software/apps/geos/netcdf/4.3.0/intel/16.0/lib:/software/apps/geos/netcdf-fortran/4.2.0/intel/16.0/lib:$LD_LIBRARY_PATH

make clean
make SAT_NETCDF=yes USE_MKL=yes
./mx
```

[After you finish compiling GEOS-Chem, you'll need to copy the file "geos" from the Fortran code directory to your run directory.]

The "make" command above will produce the following error. Ignore this error:

```
ld: cannot find -lnetcdf
make: *** [geos] Error 1
```

[When you execute the file "mx", the "mx" file will fix this error.]

How to compile the adjoint on the Johns Hopkins Rockfish cluster:
[Information courtesy of Ruixue Lei]

- The folder "Rockfish" contains the two files you'll need to compile on Rockfish.
- Move these files to your GEOS-Chem code directory.
- While on Rockfish, run the file compile_adjoint.sh . This script will read in the required software modules, set the paths so that GEOS-Chem can find the netcdf software, and will compile the model.
- You'll get an error message at the end of the compiling process. Don't worry about this error message. Next, you'll run "./mx", which will link GEOS-Chem to the netcdf and hdf5 libraries. This step should create the "geos" executable file.
- Move the geos executable file to your run directory.

How to compile the adjoint on the NASA Pleiades cluster:

- Note that NASA has restrictions on the allocation of Matlab licenses. Instead, we recommend running the inverse model using Octave, a free program that is similar to Matlab.
- In the compiling scripts folder, there's an example script showing which software modules to load, and what directory paths need to be set.

MISCELLANEOUS NOTES ON GEOS-CHEM INPUTS AND OUTPUTS

Example Matlab code for writing the flux files as netcdf:
[In this case, let's suppose we have a vector of fluxes called "shat" and the fluxes are arranged sequentially in that vector.]
[Also note: GEOS-Chem requires fluxes to be in units of molec/cm2/s. Many of Scot's flux files are in units of micromol m-2 s-1 and hence the unit conversion at the beginning of the sample script below.]

```
    lons = -180:5:175
    lats = [-89 -86:4:86 89];

    year = [ fill in the year ]
    ntimes = [ fill in with the number of days in your simulation ]
      temp= shat .* 6.02e+13; % convert from micromol m-2 s-1 to  molec/cm2/s, the unit needed for
GEOS-Chem
      temp=reshape(temp,72,46,ntimes); %again here, 72(lon) \times 46(lat) \times days%

      % One additional, important note: the function "netcdf.create" only works if you are currently in
the folder where you intend to write

      for j=1:ntimes,

      % Convert day of year to month and day of month
     [yy month day HH MM] = datevec(datenum(year,1,j));

      if month<10; month1=strcat('0',num2str(month)); else; month1=num2str(month); end;
      if day<10; day1=strcat('0',num2str(day)); else; day1=num2str(day); end;

      % Open the netcdf file
      ncid =
netcdf.create(strcat('CO2.daily.geos.4x5.',num2str(year),".",month1,".",day1,'.nc'),'NETCDF4');

      % Define the dimensions of the netcdf file
      dimid1 = netcdf.defDim(ncid,'lon', length(lons));
      dimid2= netcdf.defDim(ncid,'lat', length(lats));

      % Define new variables for the netcdf file
      my_varID = netcdf.defVar(ncid,'CO2_flux','double',[dimid1 dimid2]);
      my_varID1= netcdf.defVar(ncid,'lon','double',[dimid1]);
      my_varID2= netcdf.defVar(ncid,'lat','double',[dimid2]);
      netcdf.endDef(ncid);

      % Write data to the netcdf variable
      netcdf.putVar(ncid,my_varID,temp(:,:,j);
      netcdf.putVar(ncid,my_varID1,lons);
      netcdf.putVar(ncid,my_varID2,lats);

      % Close the netcdf file
      netcdf.close(ncid);

      end
      clear temp;
```

How to read GEOS-Chem outputs:

Estimated CO_2 mixing ratios:

- By default, GEOS-Chem will write the outputs as a binary punch file called "ctm.bpch".
- Below, I have included a short Matlab script to read the model outputs into Matlab.
- This code uses Matlab functions available in the Github repository. (The code is based off the functions found here but is updated for MERRA2 outputs: http://wiki.seas.harvard.edu/geos-chem/index.php/Matlab_software_tools_for_use_with_GEOS-Chem)

```
[ ctmData ] = readAllBPCHData( 'ctm.bpch','./tracerinfo.dat','./diaginfo.dat',true,true,false); % Read in
all contents of the file
[ ctmData ] =
readBPCHSingle( 'ctm.bpch.1','C_IJ_AVG','T_CO2','./tracerinfo.dat','./diaginfo.dat',true,true,false); %
Only read in modeled atmospheric CO2
```

Estimated gradient:

- GEOS-Chem writes the gradient to a binary punch file. This file is written to the GEOS-Chem run directory with a path/name "OptData/gctm.gdt.01." The Matlab functions described above can also be used to read the gradient into Matlab.
- If running the full inverse model, the Matlab scripts already include lines of code to read in the gradient.
- The script cost_gradient_fun.m provides an example of how to read in the gradient – if you need to write your own code to read in the gradient.

Estimated cost function:

- The GEOS-Chem cost function is written to the file "OptData/cfn.01".
- The cost function file is in plain text format and therefore does not require any special function to read.

How to specify observations for the GEOS-Chem adjoint model:

- The path to the observations for the adjoint are coded into the file "input.geos" in the run directory.
- This version of GEOS-Chem is programmed to read in observations from daily netcdf files.
- The folder "misc_code" includes a Matlab script to process OCO-2 observations for the GEOS-Chem adjoint ("process_obs.m").
- GEOS-Chem expects that the OCO-2 observation files and in situ observation files will have a certain filename.
  - For OCO-2 observations, files should be named oco2_LtCO2_YYYYMMDD.nc
  - For in situ observations, files should follow the OBSPACK naming convention: obspack_co2_1_OCO2MIP_v3.2_2021-05-20.YYYYMMDD.nc
  - For more details or to change the file name convention, see the Fortran files obs_operators/gosat_co2_mod.f and obs_operators/flask_co2_mod.f (look for the objects "FILE_PREFIX" and "FILE_SUFFIX").
- Overall file structure for the observation operators in GEOS-Chem
  - The files obs_operators/gosat_co2_mod.f and obs_operators/flask_co2_mod.f contain the observation operators for OCO-2 and the in situ observations. Even though the first file is named "gosat_", it nevertheless contains the operator for OCO-2.
  - These observation operators are called by the module adjoint/geos_chem_adj_mod.f, which is the script that directs the overall adjoint model run.

Notes on the model-data residual files:

- I've modified the observation operators for OCO-2 and for in situ observations to output a csv file that contains model-data residuals.
- The output file for oco2 is called residuals_oco2.csv and for in situ is residuals_insitu.csv
- The observation ID for the Obspack is a character string. Hence, I had to write these observation IDs to a separate file titled "residuals_situ_identifier.csv". The remaining 5 fields contained in "residuals_insitu.csv" are all numeric data fields.
- The columns of the csv file are as follows:
  - In situ observation file: the first column is the date the observation was collected.
  - Observation ID (Obspack ID or sounding ID) (saved to a separate file for in situ observations – see above)
  - latitude
  - longitude
  - CO2 model estimate
  - CO2 Observation value
  - Prescribed model-data mismatch error
  - Assimilation flag – flag that determines whether observations should be assimilated in an inverse model for the OCO-2 MIP (observations with a flag of 1 should be assimilated).
- When the GEOS-Chem adjoint is run, these files will be written in the run directory. In addition, GEOS-Chem will append the residual information to any existing file. Hence, I recommend deleting any existing residual_XX.csv files before running GEOS-Chem
- Note that I have included lines in the Matlab inverse modeling code that will rename any

existing residuals_XX.csv files at the beginning of each iteration. Specifically, the files will be renamed with the iteration number. At the end of the inverse modeling run, you should have a record of the model-data residuals at each iteration of the inverse model.

<u>CHANGES MADE TO GEOS-CHEM FOR GEOSTATISTICAL INVERSE MODELING:</u>

Brief summary of the changes we made to GEOS-Chem for the GIM:

- There are a lot of files in the GEOS-Chem model code directory that are not relevant for a CO2 simulation. However, there are a few specific files that are really important for the CO2 simulations and that had to be modified for the GIM run:
    - modified/co2_mod.f:
        - This script reads in CO2 fluxes for the forward GEOS-Chem model run.
        - This code is relatively long, but we only use a small part of this code for our runs. Specifically, we lump all of the CO2 fluxes (e.g., biosphere, fossil fuel, ocean, etc) into a single set of flux files. By contrast, this script is set up assuming that you have a different set of flux files for every different CO2 flux type.
        - Because we put all of our CO2 fluxes into a single set of flux files, we only turn on one flux category in input.gcadj (in this case, the daily biospheric flux category) and turn off all other flux categories.
        - Hence, in co2_mod.f, the only function that gets called is "READ_BBIO_DAILYAVERAGE".
        - We modified "READ_BBIO_DAILYAVERAGE" to read in the fluxes from a netcdf file instead of reading in the fluxes from a binary punch file.
    - modified/input_mod.f:
        - This script reads in run options from the file "input.gcadj".
        - I don't think I modified this script at all for the inverse model described here. With that said, I think it's useful to know which fortran script is reading in input.gcadj (e.g., in case there are any problems you need to debug).
    - adjoint/geos_chem_adj_mod.f: This script does a lot of the heavy lifting in the adjoint model. Specifically, it calls the individual components of the adjoint (e.g., the adjoint of transport, adjoint of chemistry, etc.). It also calls the observational operator. We edited this code to add the observational operator for the in situ observations.
    - adjoint/co2_adj_mod.f:
        - This file reads in CO2 fluxes for the adjoint. In the repository version of the adjoint code, the gradient is multiplied by the CO2 fluxes in this script (see the next section for more explanation).
        - Note that this script calls "READ_BBIO_DAILYAVERAGE", defined above in modified/co2_mod.f
        - We modified this code so that the observational component of the gradient isn't multiplied by the CO2 fluxes. (Again, see more explanation in the next section.)
    - adjoint/input_adj_mod.f:
        - This file reads in adjoint model options from the file "input.gcadj".
        - I don't think we made any modifications to this file.
    - obs_operators/interpolate.f90:
        - Junjie Liu provided the observation operator for the in situ observations. Junjie's script uses several interpolation functions that do not exist in the standard version of the GEOS-Chem adjoint.

        - Junjie provided the file interpolation.f90, which contains these functions, and I added that file to this version of the GEOS-Chem adjoint.
    - directory_mod.f, modified/input_mod.f:
        - I added new menu items to input.geos that allows the user to specify the path to the fluxes, path to the OCO-2 observations, and path to the in situ observations.
        - In order to add those menu items, I had to add new objects to directory_mod.f. That file creates objects that hold directory names.
        - Then I added commands to modified/input_mod.f to read those new menu items from "input.geos". The file input_mod.f if the fortran module that will read in menu items from input.geos.
    - obs_operators/flask_co2_mod.f, obs_operators/gosat_co2_mod.f:
        - The observation operators.
        - We modified these files in a couple ways: (1) We moved the path to the observation files out of the operator file and set it as an input in input.geos. (2) We also added code to write the model-data residuals to file as a csv file.
    - modified/dao_mod.f
        - By default, the observation operator (gosat_co2_mod.f) uses dry atmospheric pressure to interpolate the GEOS-Chem modeled column to OCO-2 observations. However, I noticed that Junjie Liu and the NASA-CMS system uses wet atmospheric pressure to interpolate the modeled column to OCO-2 observations.
        - I added functions to this script that will compute wet surface pressure and wet pressure of the GEOS-Chem atmospheric column. These functions are adapted from functions contained in the master branch repository on the GEOS-Chem adjoint Gitlab account.
        - Note that I modified GET_WET_PRESSURE_PROFILE to calculate the pressure mid-point of each GEOS-Chem level instead of the pressure level edges. This setup matches the subroutine GET_PCENTER that was originally called in gosat_co2_mod.f.
        - Also, note that the wet surface pressure is about ~3 hPa different from the dry pressure (equivalent to 25-30m of elevation change).

A note about the GEOS-Chem adjoint and emissions scaling factors:

- By default, the GEOS-Chem adjoint will estimate emissions scaling factors, and the code is designed to optimize scaling factors, not to optimize emissions directly.
- Below, I detail how GEOS-Chem is designed to optimize emissions scaling factors (instead of emissions) and how we had to modify the code accordingly.
- The gradient (i.e., the first derivative) of the cost function will be different if you are estimating emissions scaling factors than if you are estimating emissions.
    - Observation component of the gradient if you are estimating emissions: $H^TR^{-1}(z – Hs)$
    - Observation component of the gradient if you are estimating emissions scaling factors: s.* H^TR^{-1}(z – H[s.*scale_factor]) [where .* indicates scalar multiplication. Note that here we took the derivative of the cost function with respect to "scale_factor", not with respect to "s".]
    - Prior component of the gradient if you are estimating emissions: Q^{-1}(s – s_a) OR Gs (depending on whether you are running a Bayesian synthesis inversion or GIM).
    - Prior component of the gradient if you are estimating emissions scaling factors: Q^{-1}(scale_factor – prior_scale_factor)
- Key components of the GEOS-Chem code for the observational component of the gradient:
    - co2_adj_mod.f contains a lot of repetitive code – it repeats similar code for every different

CO2 flux type. Each repeated section of code has something similar to the following: EMS_SF_ADJ(I,J,M,IDADJ_ECO2nte) = EMS_SF_ADJ(I,J,M,IDADJ_ECO2nte) + E_CO2_ADJ * E_CO2
  - In this code, the observational component of the CO2 adjoint (E_CO2_ADJ) gets multiplied by the emissions (E_CO2). This multiplication yields s.* H^TR^{-1}(z – H[s.*scale_factor]) .
  - In our inverse model, we don't want to multiply the gradient times emissions (since we're solving for emissions directly, not for emissions scaling factors).
  - Hence, in our version of the code, we have deleted the emissions term: EMS_SF_ADJ(I,J,M,IDADJ_ECO2nte) = EMS_SF_ADJ(I,J,M,IDADJ_ECO2nte) + E_CO2_ADJ
- Key components of the GEOS-Chem code for the prior component of the gradient:
  - Note that we calculate the prior component of the gradient in our Matlab code, not in Fortran. Hence, we don't actually use the fortran code pertaining to the prior (and you should make sure to shut off the prior in the run options file input.gcadj).
  - The function geos_chem_adj_mod.f includes a subroutine entitled "CALC_APRIORI_CO2". It is called after the observational components of the gradient have been calculated. It then adds the prior component to the existing gradient estimate.
  - REG = EMS_SF(I,J,M,N) – EMS_SF0(I,J,M,N)
  - EMS_SF_ADJ(I,J,M,N) = EMS_SF_ADJ(I,J,M,N) + REG_PARAM_EMS(N) * S2_INV * REG
  - In the code above (from CALC_APRIORI_CO2), "EMS_SF" are the estimated scaling factors and "EMS_SF0" are the apriori scaling factors. Hence, the code calculates the prior component with respect to scaling factors, not with respect to emissions.
  - In addition, note that " REG_PARAM_EMS" is the regularization parameter, and it is specified in the run options file input.gcadj (set to one by default).
- Other important notes:
  - By default, GEOS-Chem will use prior scaling factors of 1 in its calculations. You'll see scaling factor objects in the code (usually denoted "EMS_SF" or "EMS_SF0"). With that said, in our version of the code those scaling factors are always set to 1. In other words, they have no real purpose in the code and are just a legacy.
  - Also note that GEOS-Chem uses the opposite sign convention for the gradient than we use in our inverse modeling code.
    - We formulate the observational component of the gradient as $g = -H^TR^{-1}(z - Hs)$.
    - GEOS-Chem omits the negative signs on the gradient (on both the observational and prior components, though we only use the observational component for our purposes).
    - Hence, in cost_gradient_fun.m, we have to flip the sign on the gradient after we read it in.

<u>MISCELLANEOUS RUN NOTES:</u>

An important note about the L-BFGS algorithm and job runtime

- The L-BFGS algorithm does not converge linearly on a solution. It usually starts by exploring the probability space, and then it will start gradually reducing the cost function. When it's exploring the probability space (which can take 6-10 iterations), L-BFGS will often produce weird solutions, and the cost function will often increase from one iteration to the next.
- Here's the reason why I think this pattern happens: L-BFGS makes a simple approximation for the Hessian and updates that approximation at every iteration of the algorithm. In the first

couple iterations, I suspect that the algorithm is refining its guess for the Hessian.

- The above information is important to keep in mind when running an inverse model:
    - If you use the “unlimited” queue on MARCC, there's no time limit on inversion_run.sh. However, if you use the “shared” queue, there's a 72 hour time limit on all runs. If you use the shared queue, remember that L-BFGS will restart the “exploring phase” every time you restart the inverse model. i.e., you'll lose computational efficiency and waste 6-10 iterations each time you re-launch the inverse model.
    - Don't cancel your inverse modeling runs within the first 6-10 iterations of L-BFGS. Otherwise, you'll get a bogus flux estimate and bogus restart files. Instead, make sure that you allow L-BFGS to run a minimum of 10 iterations before using any of the inverse modeling outputs.

An important note on the GEOS-Chem initial conditions:

- GEOS-Chem requires a CO2 initial condition to initiate the model runs.
- At the moment, I'm using Zichong's initial condition for Sept 1, 2014, created from CarbonTracker. At the time of writing, the initial condition must be in binary punch format, though we could change the GEOS-Chem code to read in initial conditions with a differnet format.
- The initial condition can have an enormous impact on the flux estimate, and I recommend being very careful with in setting the initial condition.

How to compare GEOS-Chem simulations against TCCON observations:

- We've written an observation operator for TCCON, contained in the folder obs_operators. This file is called “tccon_co2_mod.f”. If GEOS-Chem is compiled with this file, then the adjoint will read in TCCON observations in place of OCO-2 observations and write out a file containing model-data residuals.
- To compile GEOS-Chem with this option, rename the file “tccon_co2_mod.f” to “gosat_co2_mod.f”. By switching these files, the adjoint model will read in TCCON observations in place of OCO-2 observations. Next, launch the “run” file in the run directory; GEOS-Chem will run a single forward and adjoint simulation, and write a file containing model-data residuals for TCCON observations.
- In the folder “misc_code”, there's a script called “process_tccon.m.” This script will reformat the TCCON observations provided for the OCO-2 MIP into the format required by the GEOS-Chem observation operator.
- The observation identifier for TCCON is 16 digits long. However, Fortran will only write integers to file with up to 15 digit precision. Hence, I create a new observation identifier in process_tccon.m that's shorter than the original 16-digit identifier. After running GEOS-Chem, use the script cat_tccon.r to swap my shorter identifier with the original 16-digit TCCON observation identifier.

How to launch GEOS-Chem from the Matlab code:

- The inverse model is written in Matlab. It launches the GEOS-Chem model (a Fortran executable) and will then wait for GEOS-Chem to finish before proceeding with inverse modeling calculations.
- There are two ways to launch GEOS-Chem from the Matlab code: (1) Use the “unix” command in Matlab to execute the GEOS-Chem executable from Matlab. (2) Submit GEOS-Chem as a separate computing job and wait for the job to complete.
- By default, the attached code will use option #1 above, but there is commented code showing

how we've implemented option #2 on the MARCC cluster at JHU.

MISCELLANEOUS R AND MATLAB SCRIPTS FOR CREATING INVERSE MODELING INPUTS AND ANALYZING OUTPUTS

These scripts are contained in the folder “misc_code”.

- cat_tccon.r:
  - The GEOS-Chem adjoint can produce a csv file with model-data residuals relative to the TCCON observations.
  - This script will concatenate these csv files from GEOS-Chem runs generated for different years.
  - In addition, this script will add the 16-digit observation ID to each model-data residual.
- distmat.r:
  - The inverse model requires an matrix that gives the distances (in km) between each grid box in the inverse modeling domain. This matrix is a symmetric matrix with zeros on the diagonals.
  - This script will compute that matrix and save it to file.
- landmap.r:
  - This script creates a map that indicates whether each model grid cell corresponds to a lannd or ocean region.
- process_obs.m:
  - The inverse model requires daily OCO-2 observation files.
  - This script will process the OCO-2 observation file provided by the MIP and create daily observation files.
- process_tccon.m
  - This script will convert the TCCON observation created for the OCO-2 MIP to the format that is read in by the GEOS-Chem TCCON observation operator.
  - See earlier in the document for more details on comparing GEOS-Chem outputs with TCCON observations.