**DataAssimilation\_TechnicalDocumentation:**

This file contains a summary of all the scripts used for the implementation of a data assimilation study. This study aims to optimize the input parameters that will later be used to examine the effects of climate change on tropical forests.

**DataCleaning.py:**

Clean the observational data for the variables that are going to be included in the data assimilation process

***GPP:***

1 - Data is available at BCI\_data\_synthesis from the Center for Tropical Forest Science-Forest Global Earth Observatory (CTFS-ForestGEO)

2 - Data is provided every 30 min and is converted into 1 hour for comparison with simulation

***Soil Carbon:***

1 - Data is available at SoiRespiration\_2013\_2016 from Rubio and Detto 2017

2 - Data is at the daily timescale and the unit is converted from umol m-2 s-1 to KgC m-2 day-1 for comparison with simulation

***Soil Respiration:***

1- Data is available from [Soil carbon loss by experimental warming in a tropical forest](https://figshare.com/articles/dataset/Soil_carbon_loss_by_experimental_warming_in_a_tropical_forest/12144768)

2- Data is at the daily time scale. Data is available from 2016 - 2019.

***Above Ground Biomass:***

1 - Data is available at bci.tree from DRYAD data

2 - Only census from 1985 to 2015 (tree 2 -> tree 8) are taken

3 - Only alive data are taken

4 - Data with no date or no dbh (no agb value) are removed

5 - Data is at the daily timescale and the unit is converted from gC (50 ha)-1 to KgC ha-1 (there is still confusion regarding the units of agb)

**generate\_parameters\_i.py:**

Note: Sep 24: This code needs to be updated to include the new parameters that will be optimized. phiRL, r0\_mort, LMA, and LAImax, and rho\_wood are going to be removed. Constraining phiRL is not feasible as the data for the ratio between leaves and roots is not available.

This script aims to generate the initial values of the parameters and place them in a text file that will be read by the runMCMC.x executable.

*Input required:* Max number of jobs

1. Specify the parameters we want to optimize:
   1. phiRL = 2 (should not affect the flux, only the biomass, this parameter control the relative distribution between stem and fine roots)
   2. Fix LMA by using the data in this case since we are only doing physiological processes) – LMA = 0.03 (check the data for the dominant species to get this parameter)
   3. Fix r0\_mort at 0.02 *(Optional)*
   4. Rho\_wood can be obtained from data (around 300 KgC/m3)
   5. Fix LAI max at 4.5, by using a uniform profile for LMA the tree cannot sustain a high LAImax
   6. Vcmax can be set between 20-100 umol/s/m2
2. Main parameters are Vcmax (Lnbase), gamma\_LN (respiration coefficient), and the soil decomposition rate to be able to fit the soil respiration
   1. Fast SOC turnover rate (fourth one in SOM[5]) K0SOM[4] (check the contribution of the fast and slow SOM to the total respiration)
   2. K0SOM(4) and K0SOM(5)

2- Generate the initial probability distribution (Assumed Uniform for all parameters)

3- Create parameters.txt that has Max number of jobs as rows, and parameters as columns. This file is located in para\_files/

**generate\_parametersMCMC.py:**

1- Read the last accepted parameters (From running.txt).

2- Generate parameters around the currently accepted point using the equation:

*Where r is a uniformly distributed parameter between -0.5 and 0.5.*

*D is a constant (a variance suppressor). (in our case we chose it to be 5)*

*If parameter is out of range:*

* *If c>cmax, put c = cmin + c-cmax*
* *If c<cmin, c = cmax -c + cmin*

These new values will be used to generate a new parameter.txt file.

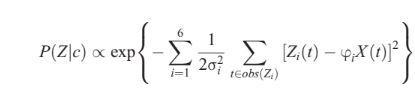
Running.txt > This is the file of the last accepted parameters, it is an nb\_parameters x nb\_simulation. *(Proposed name change: accepted\_running.txt)*

Accepted.txt > This is the file that records all the accepted parameters from all the simulations. *(proposed name change: accepted\_all.txt)*

Parameters.txt > this is the file needed at each iteration for the simulation. It is a nb\_parameters x nb\_simulations file which includes the set of parameters for each simulation at this iteration.

**get\_initial.py:**

This script aims to calculate the likelihood function of the observation such that a parameter c is assumed. The probability can be given by the equation below.

****

Where c is the parameter, Z(t) are the observed data and x(t) are the simulated output.

1- Specify simulated data needed for the comparison.

2- Get the observational data and the simulated data and save it in a dataframe.

3- Calculate the probability per the equation above.

4- Replace the line in prob.txt with the newly calculated value. (Save the value)

This function is only used to get the initial probability of the initial simulations where the parameters were assumed as accepted

**MH\_algo.py:**

This script takes the simulation number as input.

This script is a sampling script that uses a an MH Monte-carlo method which consists off:

1- Calculating the new probability P(Z/c) based on the newly simulated outputs

2- If the new probability is higher than the probability from the last simulation or higher than a randomly generated probability from the proposal distribution, the simulation is considered to be accepted.

3- The new probability will replace the one from the last simulation in prob.txt.

4- This will print 1 to the screen which will be used as a flag for the runMCMC.x script to save the parameters as accepted.

The proposal distribution is usually set as either a uniform distribution or a symmetric Gaussian distribution centered at the current point. In our case we are using the uniform distribution.

**GR\_convergence.py:**

This script executes a G-R test. The G-R test states that if the simulated Markov chain has reached convergence, the within-run variation should be roughly equal to the between-run variation.

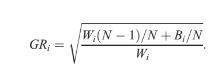
To do that we follow the steps below:

1- Read the data

2- Find the intra-run variance

3- Find the between run variance.

4- Calculate GR for each parameter via the equation below.



Where B is the intra-run variance, W is the between run variance and N is the number of simulations.

**runMCMC.x:**

This executable is responsible for executing the data assimilation, by running the simulation, running the MH\_algo, finding the accepted parameters, and saving the output.

1- Clean the Observational Data

2- Create accepted.txt

3- Provide the maximum number of jobs depending on the computer

4- Generate the parameter$j$.txt files

* $J$ refers to the simulation number
* These files are the ones that will be used for the GR test.
* These files record all the parameters at each iteration for each simulation

5- Specify the number of iterations.

6- Generate initial parameters values – From generate\_parameters\_i.py –

* Currently This file has the size 7\*5. We’re evaluating 7 parameters and we have 5 parallel simulations. *(This is going to change) Proposal: is it possible to generalize? (Updated: we’re evaluating 5 parameters)*
* This file is in the /para\_files/ folder

7- Read the lines for each simulation. And use those as an input for the simulation function.

* The simulation function:
  + Needs the row number and the row values (Value of the parameters) as input.
  + Modify the namelist.
  + Execute the simulation.
  + In the loop of the simulation, generate the probability file for the MH algorithm

8- Calculate the likelihood of observation such that a parameter set using get\_initial.py

* get\_initial.py will place the probability value in prob.txt

9- Place the values of the parameters in parameters$J$.txt

10- Move parameters.txt from ./para\_files/to ./running.txt

Now you can start the MCMC by running the simulation for as many times as needed

11- Generate the MCMC parameters using generate\_parameterMCMC.py

* Here we generate a new parameter set from the old one in running (previously accepted parameters list for each simulation) and save it in ./para\_file/parameters.txt to be used by the simulation

12- Read the lines, execute the simulation using the simulation function and the MH-algorithm function:

* The MH-Algorithm function basically combines the simulation and the MH acceptance algorithm and update the parameters lists
  + It first takes the line from the ./para\_files/parameters.txt where it reads the value of each parameter to prepare it for simulation
  + Then it prepares the folder for the compiled fortran scripts and the namelist by modifying the value of each parameter there
  + Once the simulation is done, the MH\_algo.py takes the results of the simulation and calculates the new iteration probability
  + Then the MH\_algo.py compares this probability with the old one to check if it accepts the parameters or not. If accepted it modifies the probability in ./prob.tx based on its line (simulation nb) and print 1 as a flag for runMCMC.x to read it to save the parameters.
  + If flagged, it changes the parameters values in the ./running.txt based on the line (or simulation number) and saves this parameters values in the ./accepted.txt file
  + If not accepted, nothing is changed and it goes to the next iteration

13 - Run the GR\_convergence.py where it saves the values for each parameter in ./output/GR\_test.csv. Proposal: transpose this file to make it nb\_param x nb\_iteration

14-delete unnecessary files

***Future improvements:***

Look at data and check how many trees there are and the species of the large tree and then define a closed forest to be able to run the model faster to reach equilibrium state.

Only change the size in the initial conditions, we also need to make two PFT (one deciduous and one evergreen) evergreen will contribute to carbon fluxes in the dry period only. This work will be PFT specific.

They both coexist in BCI and they have a different response to climate change especially in terms of water availability.

Start the initial condition with large trees and different cohorts and then run the simulation for one cycle. We don’t need to run the slow processes with the fast processes. We start by constraining photosynthesis and respiration and then we use these coefficients to do a simulation to get the NPP. Using this value fixed in the model, we can run simulation for yearly time steps for the forest demographic (we can run the demographic processes at yearly timestep for the given NPP)

<https://gmd.copernicus.org/articles/14/5217/2021/>

lower frequency data (such as biomass) should have a higher weight for the data assimilation

Start by fitting the long term data and then focus on short term (high frequency fluxes)