APPENDIX A: Readme file for RiceMM repository

**COMPUTATIONAL ANALYSIS OF A TWO-CELL GENOME SCALE RICE (*Oryza sativa*) METABOLIC MODEL OVEREXPRESSING KEY NADP-ME C4 PATHWAY ENZYMES VIA CONSTRAINTS-BASED MODELLING**

This repository contains all the necessary files needed for generating the pertinent data indicated in the manuscript, as well as supplementary information and generated plots.

**Startup**

The repository may be cloned using the following command:

git clone <https://github.com/rtsantos3/RiceMM>

This repository also uses multiple dependencies and may be intialized using conda. To run the said scripts, it is necessary install all the indicated dependencies indicated in the YAML file contained in the main directory. This may be done using the following command:

Conda create -f environment.yaml

The repository contains the following directories:

* **flux\_results** - Contains all raw numerical fluxes from each of the scripts. Note that due to size constraints the results from “flux sampling” are not included and may be generated from scratch
* **JSON\_maps** - contains files needed for visualizing flux maps. It also includes data taken from flux sampling runs under the “reaction\_data” subdirectory.
* **Scripts** - Contains all the necessary scripts for curation, sensitivity analysis, flux sampling as well as for the generation of various graphical and tabular summaries from the model
* **Model** - Contains both the SBML files that were used in this study. Contains the original “ios2164” model as well as the various 1 cell and 2 cell versions.
* **Plots** - Contains outputs from the visualization scripts
* **Misc** - Contains files used for parametrizing the model to specific values as well as various intermediate outputs from the curation scripts.
* **Src** - contains python files that are used to encode functions that are repeatedly used during modelling.

This repository requires a local version of Gurobi and its Python API (‘gurobipy’) as its solving engine. In case a license is unavailable you may modify the script by modifying the model.solver parameter from “gurobi” to “glpk”, which is slower but does not require any licensed solvers.

Each of the pertinent jupyter notebooks is relatively self-contained (except for commonly-used functions, which are saved in the src directory) and may be run using the base RiceMM environment. However, a separate environment is used for running the jupyter scripts that make use of Escher-FBA. For this please refer to the author’s installation guide.

**OVERVIEW OF FILES IN “SCRIPTS” DIRECTORY:**

The scripts directory contains the following subfolders:

* **Model curation** - Used to initialize the original 1-cell iOS2164 model to the 2-cell model used for simulation and flux sampling.
* **Model Benchmarking** - Contains scripts that were used for running preliminary tests on model, including testing for infeasible cycling as well as various sanity checks (i.e. checks for spurious ATP generation)
* **Model Analysis** - Contains scripts used for generating output for various sensitivity analysis runs as well as for the flux sampling runs. Contains the following files:
  + **Script for generating singular solutions and summaries.ipynb -** used for generating singular solutions from model, used previously for quick benchmarking
  + **Light Scan 2-cell model (Final).ipynb** - Used to generate sensitivity analysis for PPFD availability
  + **CO2 concentration 2-cell model.ipynb** - Used to generate sensitivity analysis for CO2 availability
  + **N-limiting conditions + PPFD 2-cell Trans model (FINAL SCRIPT).ipynb -** Used to generate sensitivity analysis file for NH4 availability, as well as for the combination of NH4 + CO2 file.
  + **Flux Sampling.ipynb** - Used for initializing OPTGP sampler as well as generating the Flux sampling matrices used for this study.
* **Convergence\_stats** - Used to quality check output from Flux Sampling runs. Contains R notebook used for running Convergence diagnostics (Geweke/Raftery-Lewis diagnostics).
* **Model Visualization** - Contains scripts that were used for summarizing and visualizing results from both the sensitivity analysis runs as well as the flux sampling runs.

**STEP-BY-STEP GUIDE:**

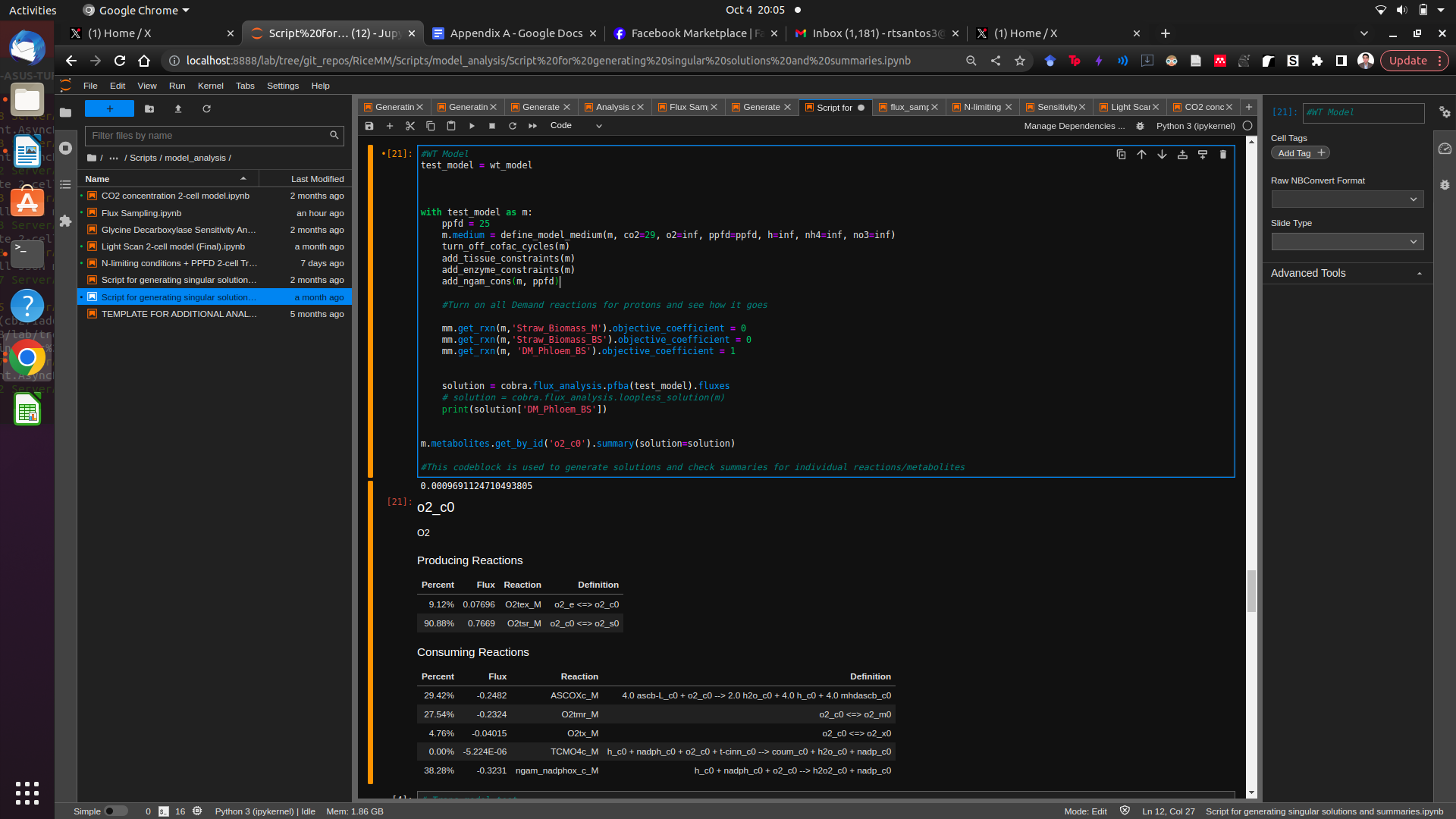
**Curation Step**

To obtain the curated version of the two cell model, use the “Curation and Generation of 2-cell model from ios2164.ipynb” file in the Model curation folder. This outputs both the 1-cell and 2-cell versions of ios2164, including all the pertinent edits and curation steps previously described.

**Generating Singular Solutions**

For the purpose of demonstrating parsimonious Flux Balance Analysis, we can use the **Script for generating singular solutions and summaries.ipynb** file. This script implements the necessary constraints that are used for initializing the two-cell model and may be used for quickly generating results. This script makes use of the bound “summary” function in CobraPy and generates singular flux vectors from each parametrized model.

A sample result is shown below showing a summary of all reactions contributing to oxygen in the M cell cytoplasm (o2\_c0), parametrized to 25 PPFD with maximizing “DM\_Phloem\_BS” set as the objective function (i.e. maximizing photoassimilate output).



**Sensitivity Analysis (PPFD, CO2 and NH4)**

Sensitivity analysis is done using the scripts described above. For example, to obtain the pFBA-based solutions using PPFD as the main limiting factor, use the **Light Scan 2-cell model (Final).ipynb** file. Each codeblock similar to below initializes the model at a particular input condition, then afterwards generates a flux vector. The loop in turn iterates over the indicated range of values (which in this case ranges from 25-1550).

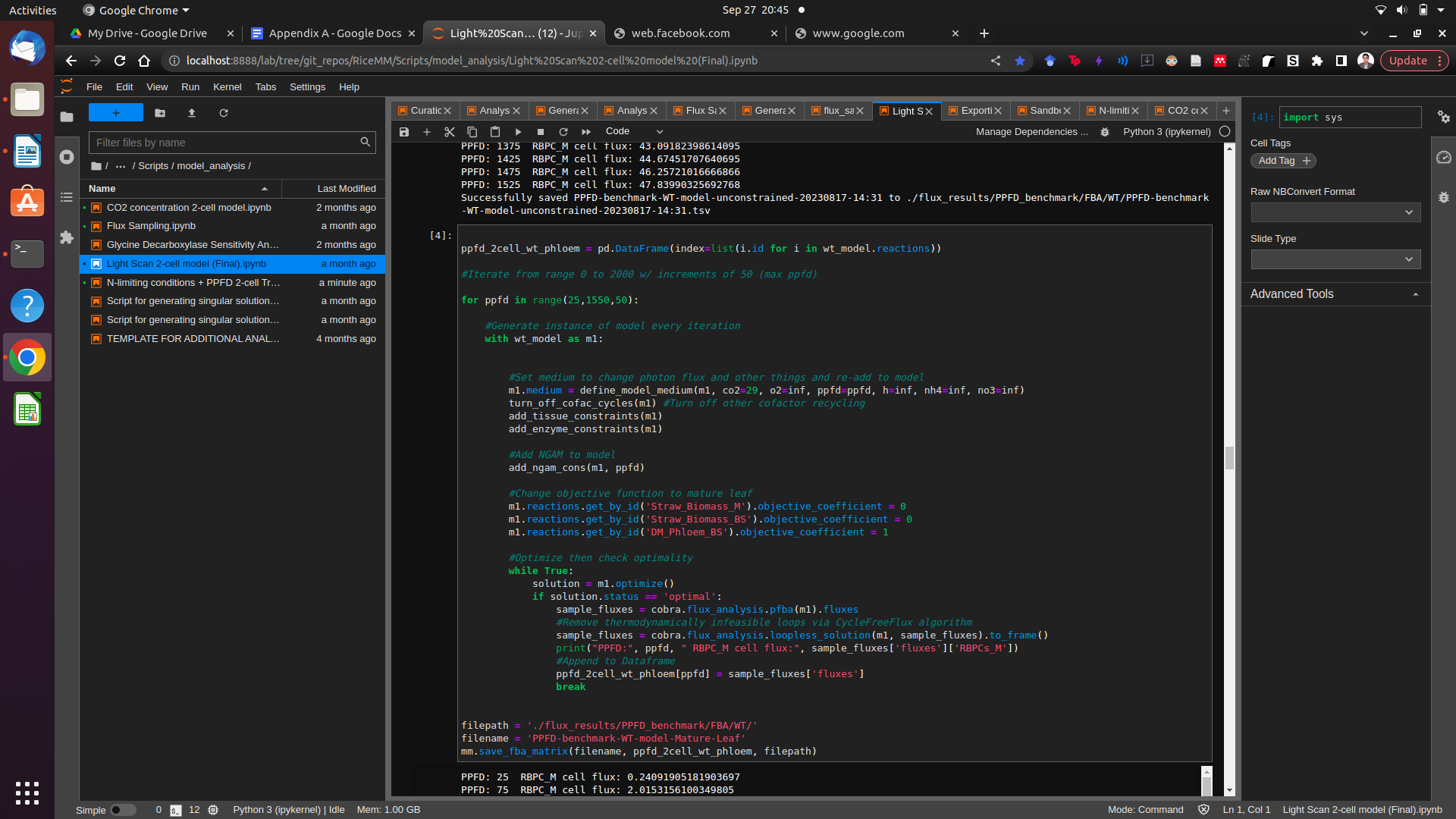
Each model is parametrized using the functions “add\_tissue\_constraints()” and “add\_enzyme\_constraints()”, “add\_ngam\_cons” functions. To add trans-specific parameters use the “add\_trans\_constraints()” and “add\_trans\_reactions()” functions. Lastly, each solution is obtained using the CobraPy function “cobra.flux\_analysis.pfba()” function and is post-processed using the “cobra.flux\_analysis.loopless\_solution()”, which ensures each solution does not carry any infeasible loops.

The input values are mainly manipulated using the “define\_model\_medium()” bound function. In the following example we may observe that only two inputs (CO2 and PPFD) are manipulated in a given instance while the rest of the inputs are set to “inf” (1e6). Output directories may be modified but are set in the “flux\_results” directory by default.

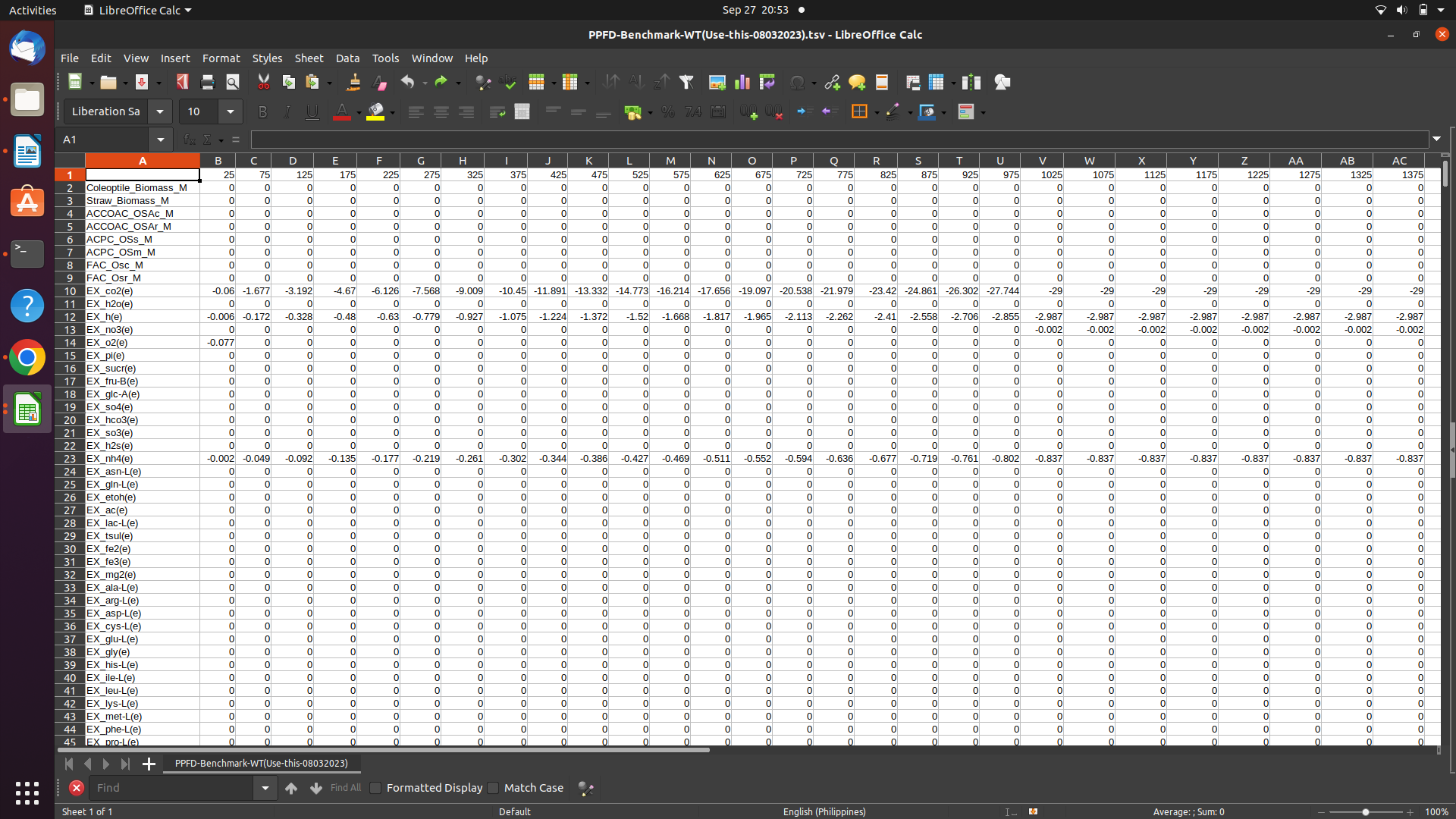
Three particular setups were done for the Sensitivity analysis part:

* PPFD - WT vs. TR models (50-1550 PPFD)
* NH3/PPFD - TR model only
* CO2/PPFD - TR model only
* NH3/CO2 (w/ PPFD as constants) - TR model only

All of the sensitivity analysis scripts follow the same general layout as below:



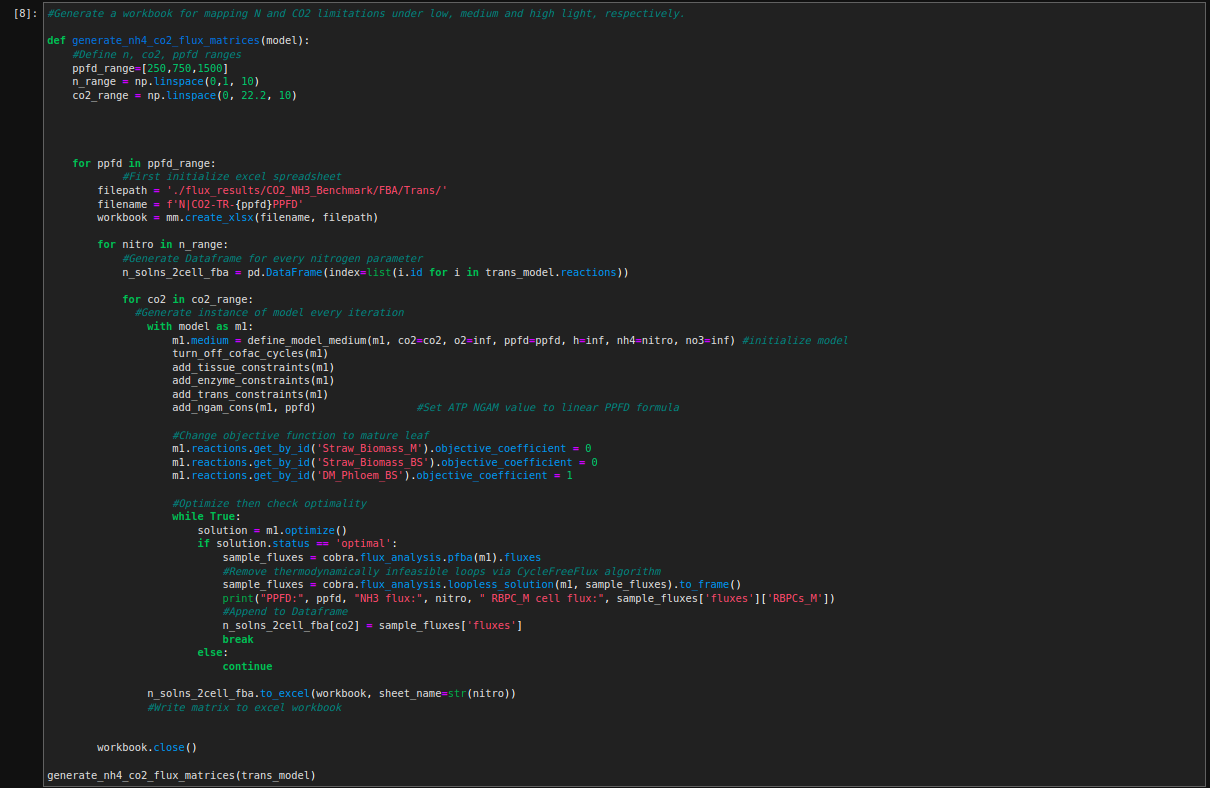
Running the codeblock above generates the following sample .tsv file, composed of generated flux solutions arranged in columns per parametrization. Visualization may then be done by plotting or summarizing reactions of interest using their IDs (found in the first column).



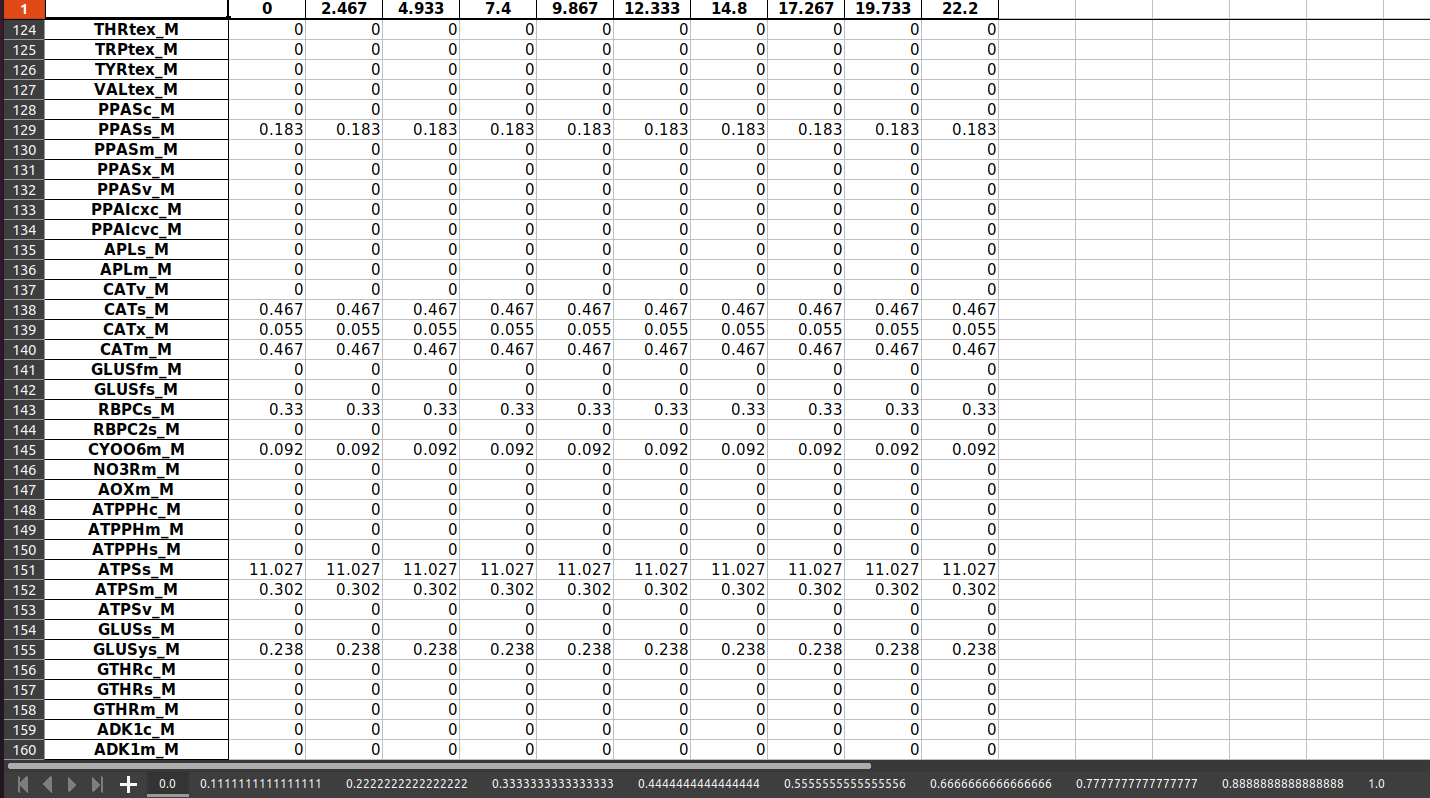
**Bivariate flux optimization**

The same methodology applies for the bivariate flux optimization done using CO2 and NH3 as the main input constraints. For this particular set up, three light parametrizations (250, 750, and 1500) were chosen and were inputted as constants for each given CO2 and NH3 range.

The function “generate\_nh4\_co2\_flux\_matrices()” Present in the “N-limiting conditions + PPFD 2-cell Trans model (FINAL SCRIPT).ipynb” was used to generate this file. A summary of the codeblock is shown below.



This method in turn generates a 3 dimensional array that is encoded as a spreadsheet file. Note that for this particular set up, the columns represent the various CO2 parametrizations (from 0 to 22.2 umol CO2/m2s) and each sheet represents the NH3 parametrizations (from 0 to 1.0 umol NH3/m2s).

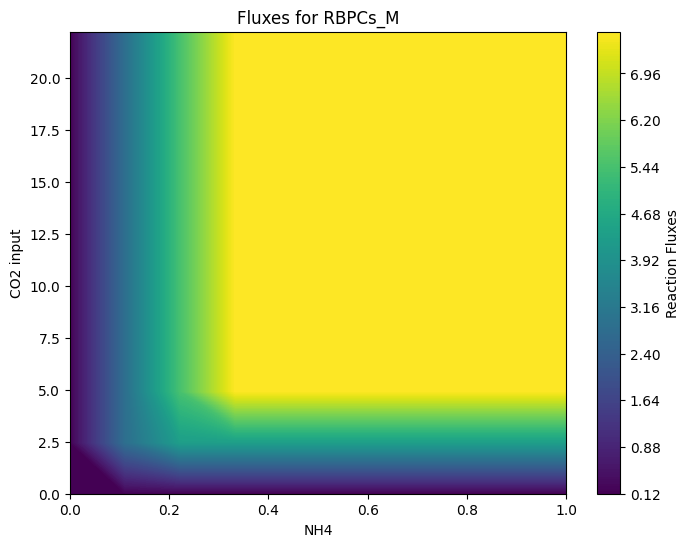


**Methods for Visualization**

Visualization of these sensitivity analyses may be done with the use of the following methods:

* Simple line charts (For Light scans) – Was done using the “Analysis of light-related reactions.ipynb” script in the “Scripts/Model\_Visualization” folder
* Contour Charts - Used for highlighting which particular conditions maximize/minimize flux. This plot type was generated using the “Sensitivity\_analysis\_plot\_benchmarks.ipynb”file.

A sample plot is shown below, showing fluxes for Rubisco under varying NH3/CO2 conditions (under 250 PPFD). A large-format graph that was presented in Figure 18 is also generated using the same script.



**Generation of Flux Samples**

The Flux sampler was initialized and parametrized using the “Flux sampling.ipynb” script in the “Scripts/model\_analysis” directory. The Flux sampler script makes use of the OPTGP sampler described previously in the methods section.

The parametrization step includes the folowing:

1.) Loading the base model from the model folder; 2.) Initializing model bounds based on input constraints; 3.) Using Flux Variability Analysis to constrain the model’s reaction upper/lower bounds prior to sampling; and 4.) Initializing OPTGP sampler for the actual flux sampling step.

In the “Flux sampling.ipynb” script steps 1-3 is fulfilled by the “parametrize\_model()” function. This function parametrizes a given model and outputs a model instance that contains all pertinent cosntraints needed to parametrize a model to a given condition.

This step includes a FVA-based preprocessing step to further constrain model bounds as well as introduce a "soft" version of the parsimony constraint via pFVA. A summary of inputs and outputs is shown below, taken from the documentation of the function:

INPUTS:

model: the model to be used for parametrization

ppfd\_low/ppfd\_high: contains the lower and upper bound for the light flux ranges

co2: Maximal CO2 (used to define either WT/TR models

if\_trans: boolean; adds a step that would add the trans specific reacctions as well as the trans related constraints

loopless: boolean; adds "loopless" argument to the FVA step; generally prevents loops but may be unstable

frac\_optimum: the minimum flux to the objective function defined in the FVA step. If set to 0.9, a model having a flux value for the obj. function of 1 will have a lower and upper bounds of 0.9 and 1.0 set as a constraint, respectively.

pruning: Boolean; use in order to remove any reactions that were observed to not carry any flux.

The following options are used for saving any intermediate values for reuse:

intermediate\_fva\_results: Directory; used for saving the intermediate FVA results for the preprocessing step (used for debuggging)

save\_final\_bounds\_file: Directory; used for saving the final lower and upper bounds of a given file.

Lastly, a preprocessed model may be reused by reparametrizing model and afterwards reapplying previously computed lower/upper bounds to the parametrized model.

fva\_bounds\_file: directory; uses a formatted FVA file as input and reapplies previously computed bounds.

OUTPUTS:

Returns a model instance that is parametrized for flux sampling.

Initialization of the Flux sampler is done using the “generate\_flux\_samples()” function. The following shows a summary of the inputs and outputs of the said function:

INPUTS:

model: a cobra model instance that was previously parametrized prior to sampling.

num\_samples: Number of samples to generate

batch\_size: How many samples are generated at a given time

output\_filename: Filename for flux sample file

thinning: Number of samples discarded for a given single flux vector. A thinning coefficient of 10000 implies that 1 out of 10000 samples is saved.

processes: Number of threads (for multiprocessing). Default is 7.

nproj: How often to reproject the sampling point into the feasibility space.

OUTPUTS:

a .csv file containing an M x N matrix with each column specific for the samples for an M number of reactions

**Visualization of Flux Sampling results**

Visualization is achieved using the scripts included in the “**model\_visualization”** file.

* **Analysis of cellular economies (ATP, NADPH, NAD)**
* Contains specific scripts for generating the budget plots used in Figures 16, 19, 20, and 21. Summarizes each given sample mean by multiplying each reaction to its coefficient and obtaining the sum of all fluxes on a given pertinent pathway. May be summarized on the **reaction** and **pathway** level, with the **reaction** level showing the contribution of each individual reaction on the summary plot. This may be modified based on which particular reaction group/pathway is needed for analysis.
* **Generating\_flux\_sample\_heatmap**
* Contains scripts that generate heatmaps showing relative flux differences between the WT and TR models. Used in figures 8 and 9 in the manuscript. Makes use of the generated flux sample CSVs and outputs a 3xM heatmap showing log-fold differences between the two parametrized model states.
* **Generating\_flux\_sample\_reaction\_charts**
* Contains scripts used to generate summary bar/violin plots for the flux samples. Used in figures 14 and 15. Inputs raw flux sample files and outputs a set of barcharts based on a given list of reactions (taken by filtering reactions based on which reactions contain a particular metabolite, which reactions are part of a given pathway, etc.).
* **Generate flux sample correlation charts**
* Contains scripts used for generating the correlation charts used for figures 22-26. This inputs the raw flux sampling CSVs and generates pairwise comparisons between the reactions of each individual flux sampling parametrizations. Generates a scatter plot containing the individual correlation coefficients per comparison. May be modified in order to indicate which (singular) reaction to compare with as well as which particular list of reactions to compare to.
* **Generate 2-cell JSON map**
* This file contains scripts used in generating flux maps using Escher-FBA**,** used in figures 12 and 17. Note that this script requires a fresh environment due to outdated dependencies used by the library. This script is used to generate a widget that may be used to overlay flux values over a given file, as well as format a raw flux sampling matrix values to a compatible format.

Two sets of data have been prepared from the 6 sampling runs. First are the means of each given reaction, while the second are comparisons between each particular light parametrization (250, 750, and 1500). These sample datasets are available in the “json\_maps/reaction\_data” subfolder. A readily-available sample of the JSON map (exported as HTML) is also available in the “json\_maps/html\_map.html” file. A sample map from the said file is shown below and may be modified based on necessary reactions/reaction data for visualization.

