**OpenFLUX**

A MATLAB-based application to perform steady-state and dynamic 13C-MFA based on the EMU framework. The application facilitates the inference of metabolic pathway activities from 13C enrichment data (mass isotopologues) using simple text inputs and scripting interfaces. OpenFLUX is now scripted as a MATLAB class. This creates an OpenFLUX object that acts like a container for data, model, variables and allowable interactions. It also keeps a history of the optimisation solutions. The object is saved to a target folder and is designed to be self-contained/portable for deployment in computing clusters. Specific objects can be called upon to be simulated, optimised or modified.

Author: Lake-Ee Quek

Email: [lake-ee.quek@sydney.edu.au](mailto:lake-ee.quek@sydney.edu.au)

Updated: August 2020

Available input and models:

* OpenFLUX version 2009 steady-state toy model
* Steady-state toy model (similar to version 2009)
* Dynamic toy model
* Adipocyte insulin response model

General workflow for OpenFLUX:

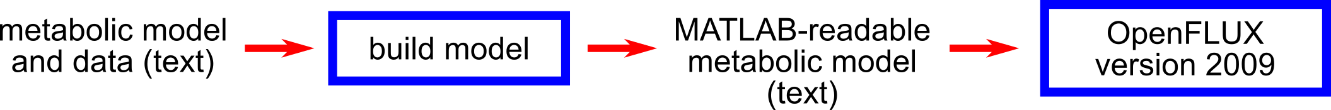
1. Edit OFspec\_\*.m to specify parameters of the OpenFLUX object.
2. Run the required task in OFstartHere.m by toggling on/off the specification file and task.

A clock sitting in the dark

Description automatically generated

Workflow for implementing steady-state OpenFLUX version 2009:

1. Add “mFiles” folder (version 2009) to MATLAB path.
2. Specify model file location in OFstartHere\_2009.m (on line 8).
3. Run OFstartHere\_2009.m to generate MATLAB-readable model files (contained in the same folder as the model file).
4. Run start13OF.m (resume OpenFLUX 2009 workflow).



The following tutorials are intended as step-by-step guidance on how to use OpenFLUX for running different models with different setups.

[Tutorial 1 – dynamic toy model, SBR mode 3](#_Toc47536018)

[Tutorial 2 – dynamic toy model, ODE mode 8](#_Toc47536019)

[Tutorial 3 – Monte-Carlo runs for dynamic SBR toy model 9](#_Toc47536020)

[Note 1 – input model 10](#_Toc47536021)

[Note 2 – input data 11](#_Toc47536022)

[Note 3 – leastSQ script 12](#_Toc47536023)

[Note 4 – optimisation results 12](#_Toc47536024)

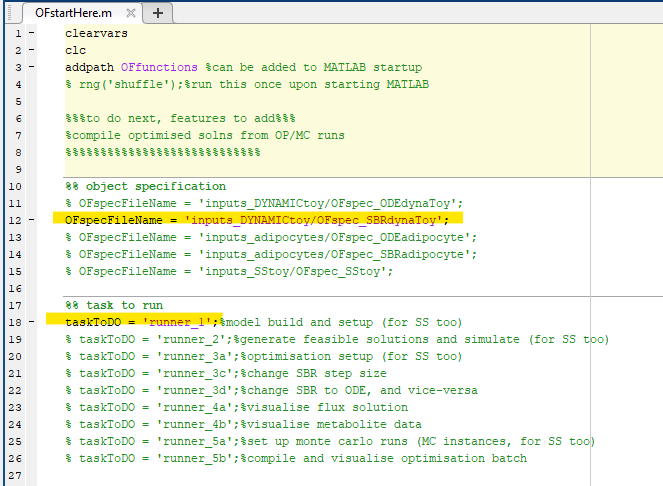
[Note 5 – change SBR to ODE and vice-versa 12](#_Toc47536025)

## Tutorial 1 – dynamic toy model, SBR mode

Stage 1: start-up and build model

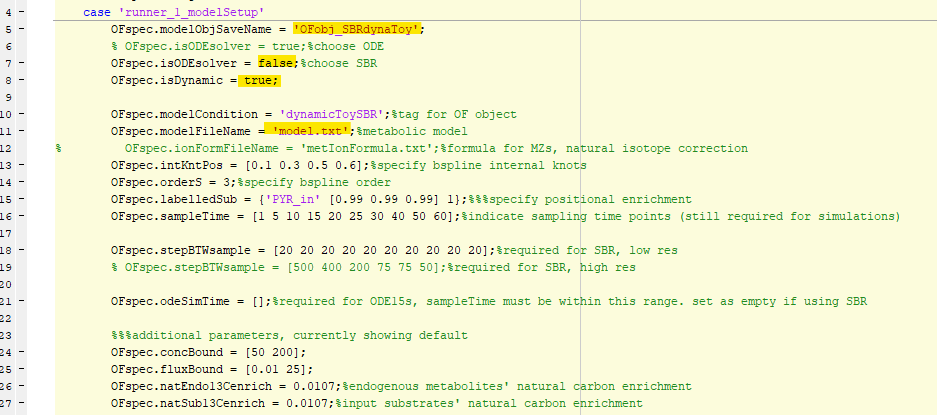
Open script OFstartHere.m in MATLAB editor. This is where you choose the input model/data, and the task to be performed.

Uncomment lines for dynamic SBR toy model and *runner\_1*. Comment out other lines.



Inspect script inputs\_DYNAMICtoy\OFspec\_SBRdynaToy.m in MATLAB editor. Set the OpenFLUX object name. Make sure SBR is chosen (c.f., ODE), and model is set to be dynamic (c.f., steady-state). MATLAB will eventually add folder inputs\_DYNAMICtoy\ to path, and will read model.txtfrom this input folder (multiple MATLAB paths can occasionally cause conflict/mix up).

Specify B-spline order and placement of internal knots. During optimisation, knot placements are randomly allocated. Set simulation/optimisation boundary for metabolite concentrations and fluxes. Specify 13C enrichment of endogenous metabolites and external substrates. In OFspec.labelledSub, more than one labelled substrate can be provided. Specify the sampling time points (can be arbitrarily set for simulations). Specify the number of steps between sampling time points. OFspec.sampleTime and OFspec.stepBTWsample are vectors of the same length.

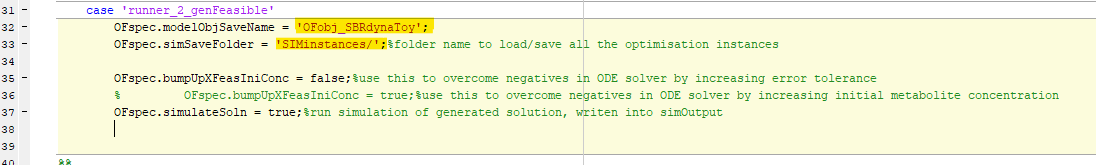


Run OFstartHere. The OpenFLUX object OFobj\_SBRdynaToy.mat will be created in the input folder inputs\_DYNAMICtoy\. The parameters and settings in Step 3 are saved to this object.

Stage 2: check model and run simulations

In OFstartHere, uncomment *runner\_2* and comment out the rest.

Specify the name of the OpenFLUX object (OFobj\_SBRdynaToy.mat) to be loaded and a new target folder in the current directory to save the simulated instances (SIMinstances/).



Run OFstartHere. At the end of *runner\_2* a simulated feasible instance will be created and saved with a time stamp in the target folder. The feasible solution is contained in variable simSave.xFeas. By setting OFspec.simulateSoln = true; the script will also generate simOutput, which contains the simulated fluxes, concentrations and MIDs over time, which can be plotted manually.

Flux matrix has reactions (row) × time (column). The reactions equations are in OF.rxnEQ.

Concentration matrix has metabolites (row) × time (column). The order of metabolites are shown in simOutput.conc.metList.

MIDs are set out as row vectors over time in simOutput.EMU.emuFract. MID names are shown in simOutput.EMU.emuList.

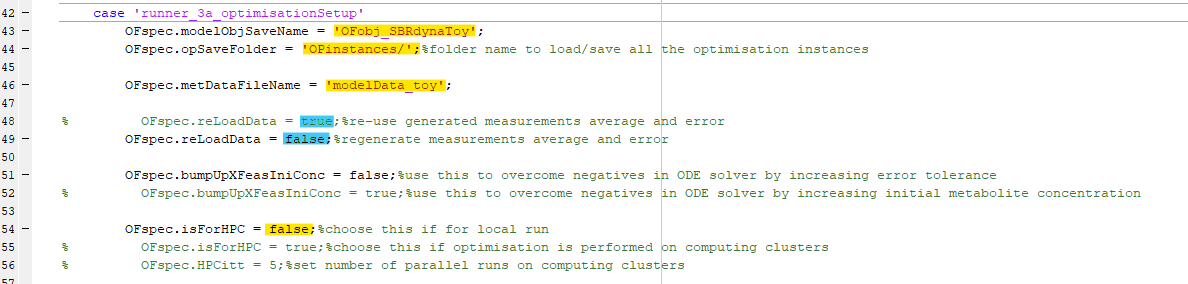
Re-run OFstartHere to generate multiple simulated instances. Users can also modify or separately generate xFeas, and then feed this variable into simOutput = OF.simSoln(xFeas); in the command line to regenerate the outputs.

Note: although the variable OF is visible and the parameters can be modified, manually editing them do not change the hidden/underlying model. Changes ought to be made in OFspec, and then re-run *runner\_2*.

Stage 3: incorporate data and setup optimisation (fit data)

In OFstartHere, uncomment *runner\_3a* and comment out the rest.

Specify the name of the OpenFLUX object (OFobj\_SBRdynaToy.mat) to be loaded and a new target folder in the current directory to save the optimisation instances (OPinstances/).



Specify the file name of the input data (modelData\_toy.txt). OpenFLUX will read input data and estimate MID fractions (and errors by bootstrapping) from the data provided. When doing this for the first time uncomment OFspec.reLoadData = false; in order to generate one version of the input data; a .mat data file is created in the input folder. To reuse the same data file in subsequent occasions, set the right OFspec.metDataFileName and toggle line to OFspec.reLoadData = true; and add suffix \_keep to the .mat file.

Run OFstartHere. At the end of *runner\_3a* an optimisation instance (starts with infeasible solution) will be created and saved with a time stamp in the target folder. Each instance is given a random set of knots (i.e., fixed to this instance).

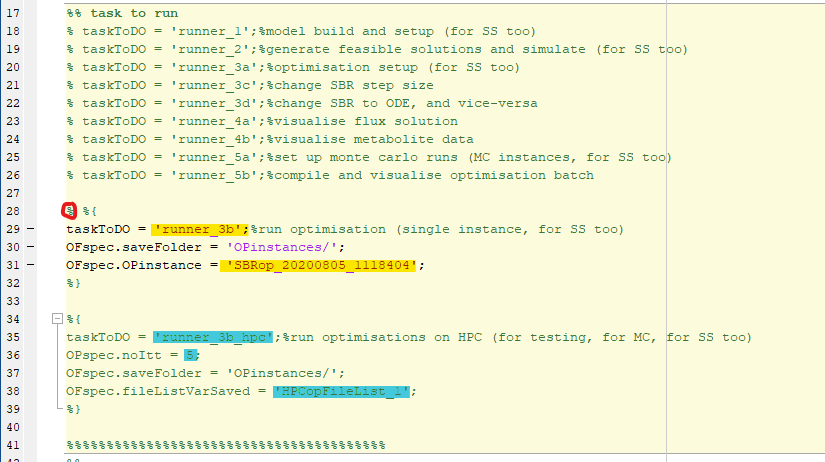
Re-run OFstartHere to generate multiple optimisation instances. Alternatively, uncomment OFspec.isForHPC = true; and specify the number of instances with OFspec.HPCitt = 5;. The list of new instances is stored in HPCopFileList.mat in the current directory; rename this file to avoid it being written over.

Note: The original data is copied into each optimisation instance, which can be displayed/examined using *runner\_4b*. This means that optimisation instances must be regenerated when the original data is modified.

Stage 4: run optimisation (parameter estimation)

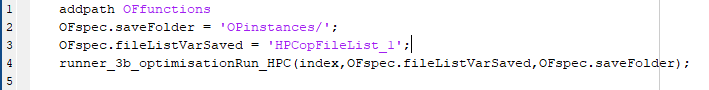
In OFstartHere, uncomment *runner\_3b* and comment out the rest. Specify target folder where optimisation instances have been saved. Specify the file name of one instance to run optimisation. Run OFstartHere. A single optimisation run may take a while to complete.

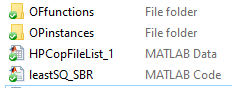
When fmincon stops, the optimisation instance will be updated with the optimisation output as a new entry and saved to same file. If an optimisation has failed to converge, then re-run the same optimisation by running OFstartHere without any changes. Optimisation will resume by using the most recent solution for the initial solution.



runner\_3b\_hpc (high-performance cluster) runs a script that exemplifies how multiple optimisations can be performed locally as a batch. Specify the size of the batch in OFspec.noItt = 5;, the target folder OPinstances/ where optimisation instances are stored, and the .mat file containing a list of optimisation instances created by *runner\_3a*. runner\_3b\_optimisationRun\_HPC will then run the *i*th instance locally. MATLAB parallel computing toolbox (parfor) can be used here.

In a computing cluster, customise a simple batch script around the function runner\_3b\_optimisationRun\_HPC, which may look like below. index is the index of an optimisation instance in HPCopFileList\_1.mat to be run by the computing node.



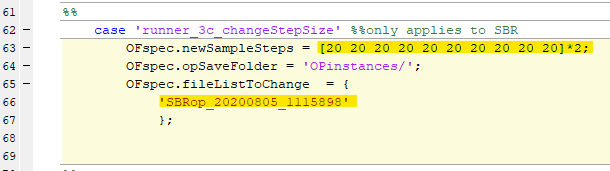


The above files and folders must be visible to MATLAB.

Note: add input folder (e.g., inputs\_DYNAMICtoy) to MATLAB path in order to make leastSQ\_\* visible. This is not required if other tasks were executed beforehand, or leastSQ\_\* is located in current directory.

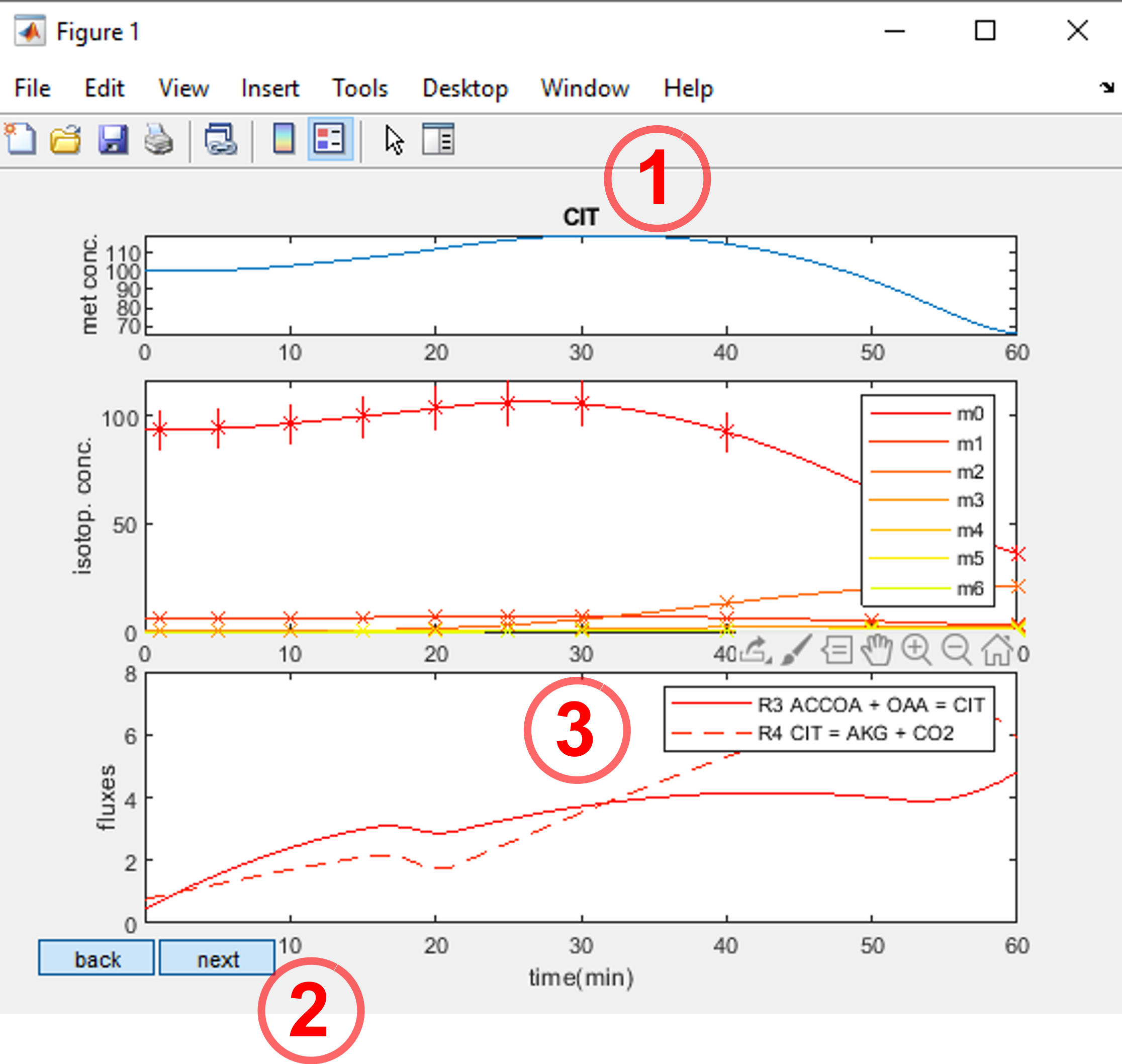
Stage 5: increase SBR step size

SBR algorithm usually starts at low resolution to achieve faster computation, and step size is eventually increased to improve numerical accuracy. In OFspec\_SBRdynaToy, increase step size density by increasing the values in OFspec.newSampleSteps without resizing the vector. Specify the optimisation instances that ought to be modified. Use dir in command line to create this list. Run OFstartHere to update step size of listed optimisation instances.



Stage 6: visualise feasible/optimised solution

In OFstartHere, uncomment *runner\_4a* and comment out the rest. Specify the optimisation or simulation instance, and the target folder where the file could be found. Run OFstartHere to generate the figures.



① metabolites plotted are ordered alphabetically.

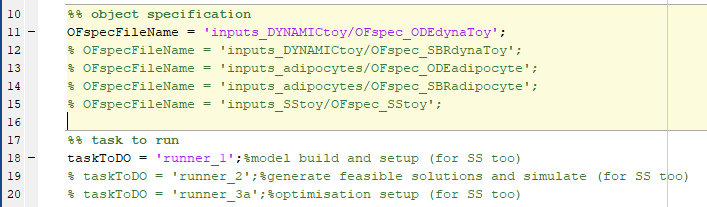
② click “next” or “back” to browse different metabolites

③ fluxes for input (full line) and output (dash line) reactions are plotted.

## Tutorial 2 – dynamic toy model, ODE mode

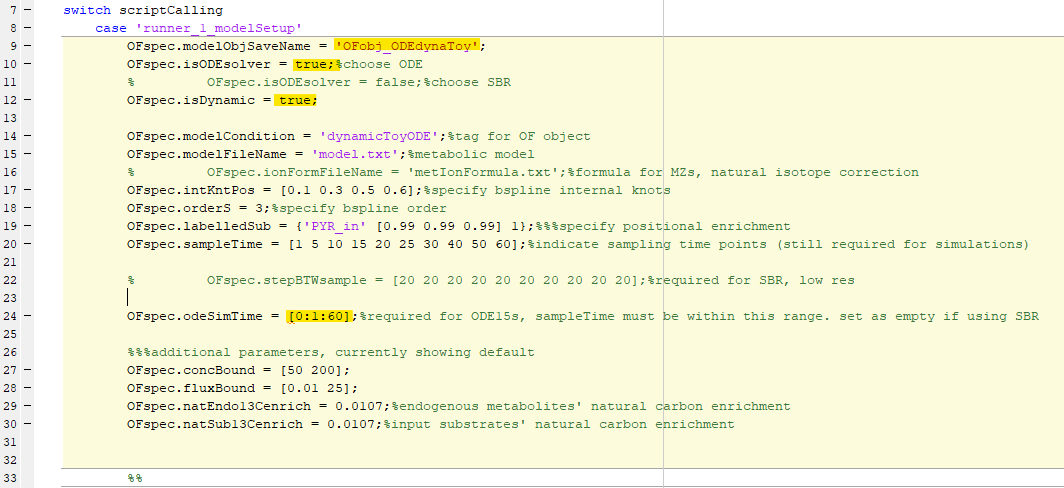
Stage 1: start-up and build model

In OFstartHere.m in MATLAB editor uncomment lines for dynamic ODE toy model and *runner\_1*. Comment out other lines.



Inspect script inputs\_DYNAMICtoy\OFspec\_ODEdynaToy.m in MATLAB editor. Make sure ODE is chosen, and model is set to be dynamic. Set ODE time in OFspec.odeSimTime to range from 0 to the largest sampling time point. SBR step size is not required.

Run OFstartHere. The OpenFLUX object OFobj\_ODEdynaToy.mat will be created in the input folder inputs\_DYNAMICtoy\.



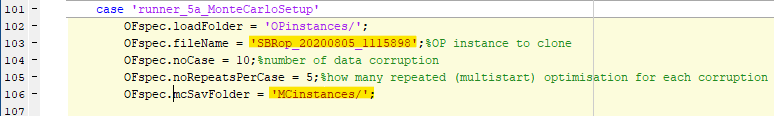
Stage 2: run simulations and optimisations

The steps to accomplish this is the same as Tutorial 1 Stage 2, 3 and 4. To visualise simulated or optimised results use steps from Tutorial 1 Stage 6.

## Tutorial 3 – Monte-Carlo runs for dynamic SBR toy model

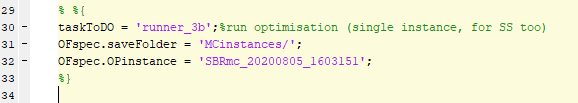
In OFstartHere.m uncomment lines for dynamic SBR toy model and *runner\_5a*. Comment out other lines.

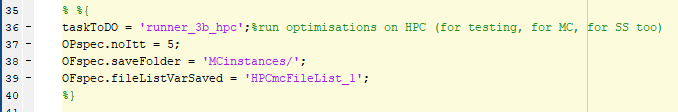
In inputs\_DYNAMICtoy\OFspec\_SBRdynaToy.m pick an optimisation instance to clone. This clones the underlying model and data but not the solution). Specify number of times to perform data corruption (OFspec.noCase), and number of optimisation iteration for each set of corrupted data (OFspec.noRepeatsPerCase). Clones of the latter share the same constraining data but have different knot placement.



Run OFstartHere.m. HPCmcFileList.mat is generated in the current directory. It contains mappings how each Monte-Carlo optimisation instances are related. Each optimisation instance carries information about the clone source (OF.mcCloneSource) and mappings (OF.mcCaseRep), and the original (column 2) and corrupted (column 4, 6) data in OF.dataMet.

Use *runner\_3b* or *runner\_3b\_hpc* to perform the optimisation as indicated in Tutorial 1 Stage 4. Edit target folder and optimisation instances.



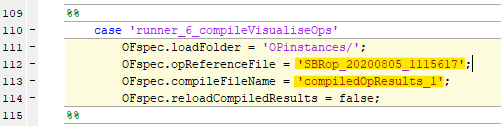


Note: input folder (e.g., inputs\_DYNAMICtoy) needs to be on MATLAB path to make leastSQ\_\* visible.

## Tutorial 4 – compile optimisation (and Monte-Carlo) results

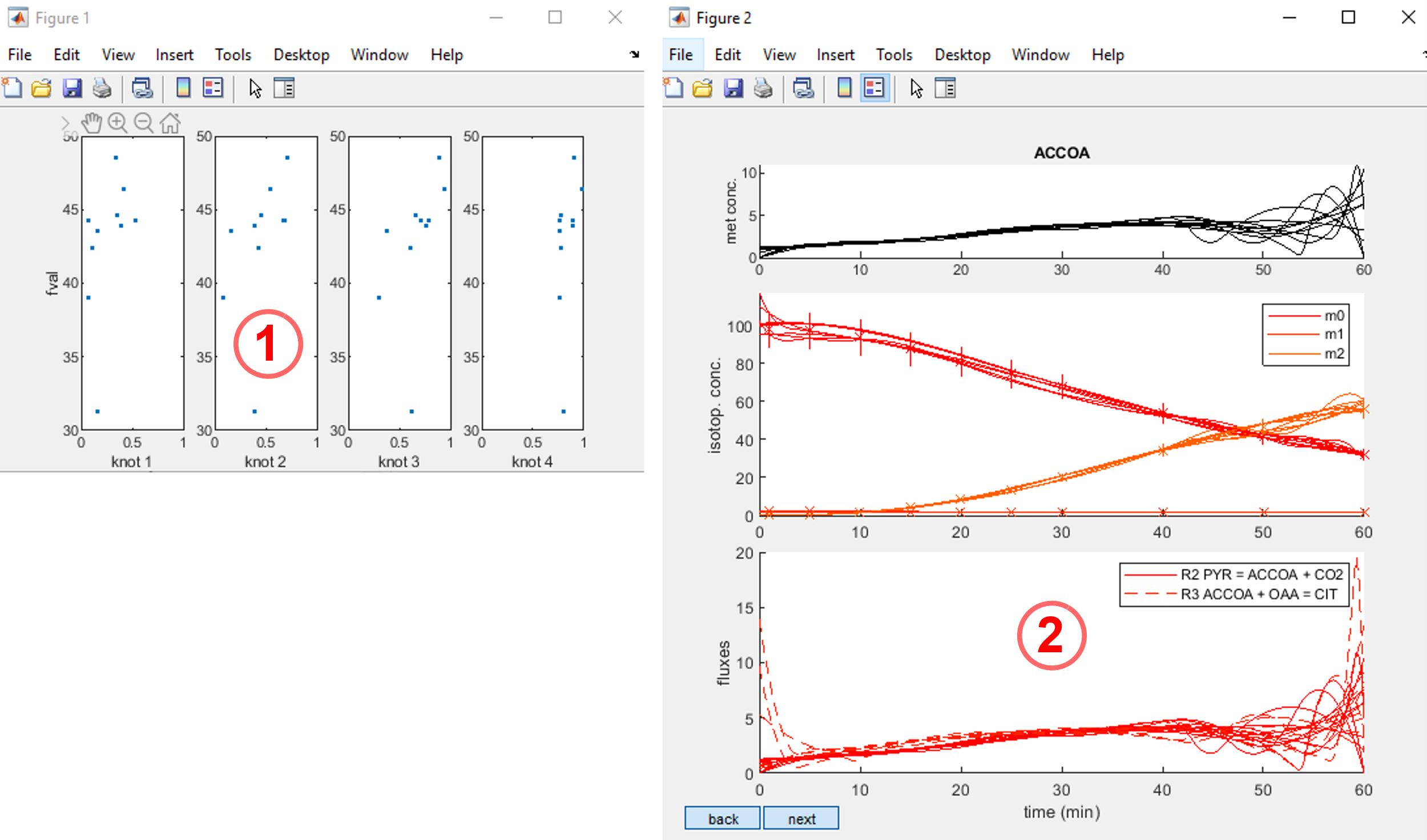
In OFstartHere.m uncomment lines for dynamic SBR toy model and *runner\_6*. Comment out other lines.

Specify the folder containing the optimisation results. Set an optimisation instance as reference. Specify the file name the compiled results will be saved under; the .mat file is saved in the current directory. When compiling, the script will filter away instances with different model, data, algorithm and setup. If the aim is to re-generate the MATLAB figures from (existing) compiled results, then set OFspec.reloadCompiledResults = true;.



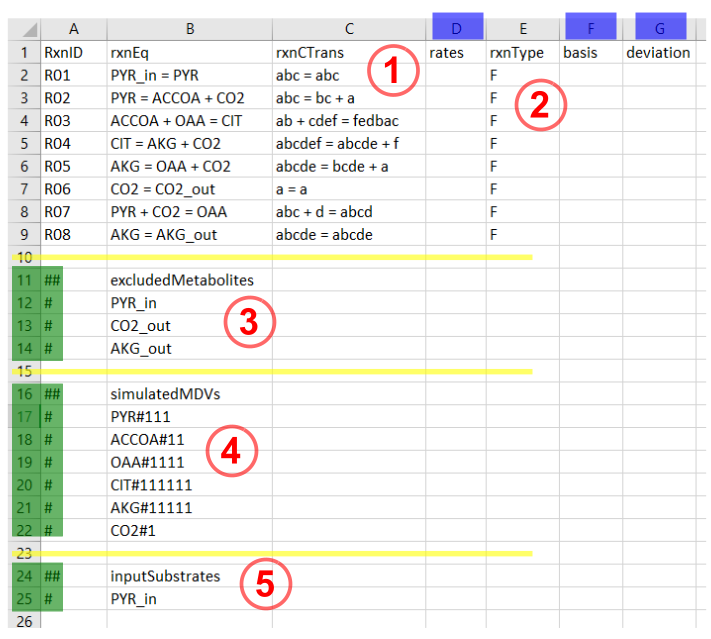
Run OFstartHere.m to plot a composite graph of knot placement and simulated outputs. The results are contained in the variable compiledResults.

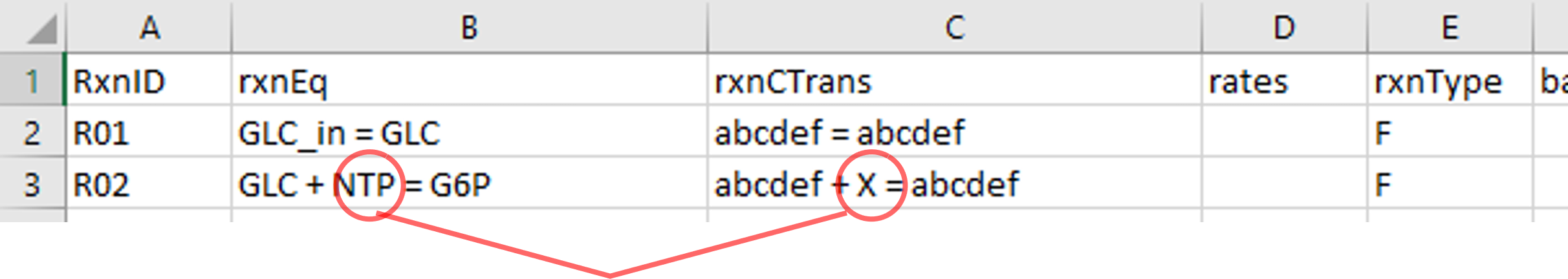
Note: For Monte-Carlo results, compiled results ought to be filtered such that the best solution from each case is kept by setting OFspec.mcBestSoln = true;.

① knot placements

② composite plot of concentrations, MIDs and fluxes.

## Note 1 – input model (dynamic)





➀ Use lower case Latin alphabet. “Space” is required. Use “X” to exclude metabolites (matching position in Column B) from EMU balances.

② All reactions are “F” type for a dynamic model.

③ Input-output (external) metabolites excluded from flux balancing.

④ List all internal metabolites. Can add external metabolites to this list if they are to be simulated. The possible EMUs for 3-carbon pyruvate can be PYR#111, PYR#110, PYR#101, PYR#011, PYR#001, PYR#010 and PYR#100.

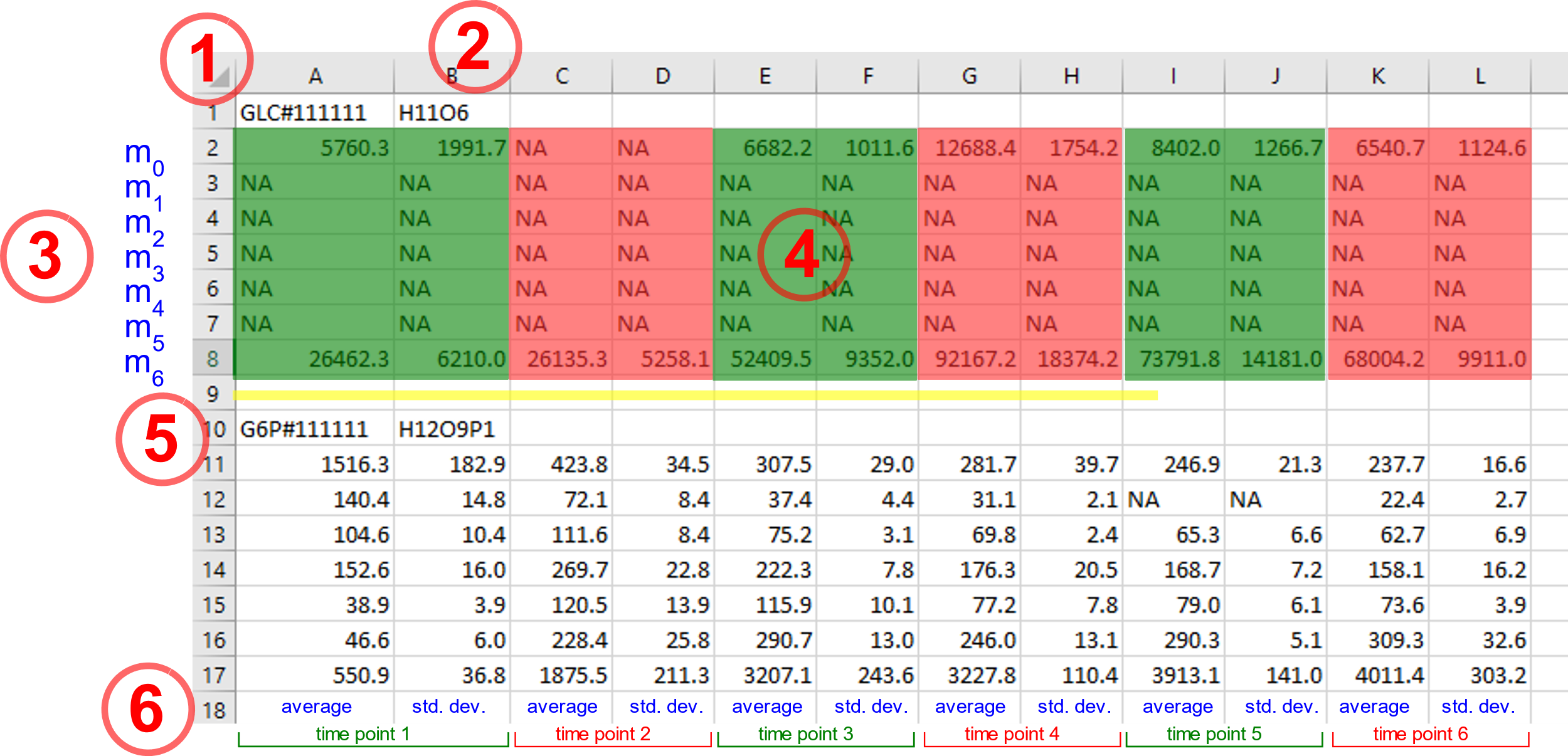
⑤ List input substrates that are not naturally labelled. Naming in OFspec\_\* must match these.

**Blue**: these columns are not used for dynamic model

**Green**: keep these signposts. “##” for headers; “#” for members under the header.

**Yellow**: Provide empty rows to separate tables.

## Note 2 – input data (dynamic)



➀ Metabolite naming convention must match a member of “simulatedMDVs” in the model table.

② Ion formula of interfering (eg: non-backbone carbon) atom for natural enrichment correction.

③ Mass isotopomer distribution of glucose. This vector needs to be in absolute (c.f., fractional) quantity in order to calculate metabolite abundance.

④ “NA” to represent missing values

➄ Next metabolite. Leave an empty row between tables (yellow line)

➅ Time course data. Average and standard deviation provided side-by-side. Number of column pairs must match with OFspec.sampleTime in OFspec\_\*. Steady-state data follows the same structure, but has only one pair of columns (a single data point) with averages expressed as fractional enrichments.

## Note 3 – leastSQ script

## Note 4 – optimisation results

## Note 5 – change SBR to ODE and vice-versa