Tutorial FASIMU

This tutorial documents version 2.1.6.

Preparation

Install FASIMU as described in the manual. Download the `FASIMU_Tutorial_Example.zip` for example in the directory `/tmp`:

```
cd ~
mkdir FASIMU-Test
cd FASIMU-Test
unzip /tmp/FASIMU_Tutorial_Example.zip
sbml2fa ery.sbml
```

Getting acquainted

As you may have learned in the manual all FASIMU files are human-readable text files. Several files have been generated from the SBML file which will be used for the simulations. The SBML file will not be regarded any more. So to start with it is a good idea to have a look at the information in the files:

```
less reactions
less metabolites
less equilibriums
less enzymes
less simulations
```

Getting started

Start FASIMU with

```
source fasimu
```

A status message appears at the screen including the number of reactions and metabolites. Note that some files have changed now:

```
less reactions
less metabolites
```

Pseudo metabolites ending with _ex and pseudo reactions whose identifier ends with _tr have been added. This is necessary to have full control: the simulations environment can allow the excretion and uptake of any metabolite in the model even if it is not defined to be an exchange metabolite in the original model. Usually you can ignore this but it is important to have this in mind if you analyze intermediate information files. The repeated call of `source fasimu` does not add these pseudo reactions again: the files ending with `.original`:

```
less reactions.original
```

So it is safe to call `source fasimu` in the same directory again.

The solver has been chosen automatically. If an error message about the solver is displayed the installation of the solver on your system might not be correct. The solver can be manually selected with `source fasimu <solver>`, where `<solver>` can be `cplex10`, `cplex9`, `glpk`, `lp_solver`, or `lindo`.

Before any simulation is started the FBA function call together with options can be chosen by setting the variable `optimization_call`. Otherwise the default is chosen:

```
Info: variable 'optimization_call' set to the default of "compute-
FBA".
```

Do the computations

The simulation tasks are defined in the file `simulations`:

```
cat simulations
```

You will see the definition of the simulations: e.g. in the second column (the objective) and the third column (the constraints). The identifier `stdef` among the constraints refers to additional constraints in a separate file (the file extension `.txt` may be omitted).

```
cat stdef.txt
```

This is a convenient way to define standard substrates and waste products. The result of the substitution can be checked:

```
cat simulations_work.fgf
```

You may notice that here all the substrates and waste products are mentioned again which have already been marked as exchange metabolites in the original SBML file. In SBML inward and outward fluxes are not differentiated by the boundaryCondition tag, consequently, they are ignored. However, it is possible to use the original boundaryCondition information in a file with

```
make-standard-exchangables > autostdef.txt
cat autostdef.txt
```

In contrast to the above file `stdef.txt`, here the exchange is allowed in both directions (marked with a = character).

Start the simulations with:

```
simulate
```

Note the progress information, each simulation computed is acknowledged by giving the number of simulations computed already and the total number of simulations. This is an especially useful feature for larger models, computationally harder algorithms, and a long series of simulations.

View the results in text form

An overview is given by

```
cat evaluation.txt
```

You may notice the + characters in the second column indicating that the simulation is successful, e.g. in some cases the production of a certain metabolite, in other cases the non–existence of a solution. The fourth column in the file `simulations` is devoted to this question: a metabolite identifier means that this metabolite must be exchanged over the system boundary. It is also possible to give a reaction identifier there (in which case this reaction must carry a nonzero flux) or a zero indicating that the simulation is considered to be successful if no solution exists.

```
cut -f1,4 simulations
```

For long lists of simulations simplified function is supplied which only shows the failed simulations:

```
negeval
```

In the given example nothing is shown; all simulations show the desired result. The actual flux distributions predicted are contained in `allout.txt`:

```
less allout.txt
```

This file shows all flux distributions of a computation series. The flux distributions are separated by lines with many # characters followed by the name of the simulation. In the next line there is a description of the system boundaries of this particular simulation. The next line gives a short comment whether a solution is computed with optional comments from the solver. Additionally warnings are shown here.

Then the solution is given in a tab-separated format. The first column gives the name of the reaction. The second column gives the flux value. Note that zero fluxes are not recorded in the solution. Next column gives the equilibrium constant just for information, then the reaction written with the metabolite names given in the file `metabolites` follows. The next column hold possible reaction names given in the file `reaction-names`.

For computations where the thermodynamic feasibility is checked a section follows given hypothetical concentrations which are compatible with the flux distribution.

View the results in the visualization packages

Using BiNA

The flux analysis plugin of BiNA requires a special input file which can be generated from allout.txt with the function allout2bina:

```
allout2bina
bina BiNA/All.csv
```

You may notice two things. The node ATP is highly connected and so are the nodes of ADP NAD and NADH. You can set the alias function for such metabolites one by one in BiNA. However fasimu offers an easy way to define some metabolites as aliases right from the start:

```
allout2bina -a ATP ADP NAD NADH
```

Loading the result file in bina as above the nodes of these metabolites are hidden. The BiNA function View->Open children makes them visible but with instances.

You may also notice that the exchange metabolites are colored (reddish for input and orange for output) and the internal metabolites have no color. But it is possible to assign colors to metabolites depending on the compartment:

```
allout2bina -c cell[49,234,222] ext[244,44,238]
```

The syntax is straightforward: first the compartment name and then the RGB code in brackets with values ranging from 0 to 255. No space characters are allowed inside the compartment/RGB item.

Using Cytoscape + FluxViz

First, prepare the flux distribution files:

```
allout2valfiles
```

Call Cytoscape, read the file ery.sbml with the plugin SBMLReader2 and read in the val-files. See the manual of FluxViz for more details.

Using CellNetAnalyzer

The flux distribution files are again prepared with:

```
allout2valfiles
```

Copy the files reactions and metabolites into a separate order and create a CellNetAnalyzer model (see CellNetAnalyzer manual for details). Start MatLab and CellNetAnalyzer in the MatLab command window, then load the model just created. Load the val–files with the Option called “Read scenario”.

Further algorithms

Thermodynamic feasibility

This constraint requires equilibrium constants in the file `equilibriums` and concentration ranges defined for metabolite classes for the benefit of easy management. The classes are defined with `concentration-ranges-classes` and each class has common concentration ranges given in `concentration-class-ranges`. Each line holds the strict lower bound, the relaxable lower bound, the envsioned set point, the relaxable upper bound, and the strict upper bound. First, you might to review this information

```
cat equilibriums
cat concentration-class-ranges
cat concentration-ranges-classes
```

Then, start the computation series

```
optimization_call="compute-FBA -T"
simulate
less allout.txt
```

The option –T selects the TR–criterion to be used. In allout.txt you will see not only the reaction but also concentration values compatible with this flux distribution. For more realistic concentration values you may want to switch on the soft bounds (–s option) and the set points (–w) option, available only for CPLEX (quadratic objective).

```
optimization_call="compute-FBA -T -w 0.1 -s 100"
simulate
less allout.txt
```

By default, only the hard bounds are used.

Intake minimization

As opposed to the previous simulations here only the usage of glucose is minimized, thus, calculating the maximal yield with respect to substrate utilization:

```
echo -e "ATPase 2.38\nDPGase 0.494\nGSHox 0.093\nPPrPT 0.0258" \
> fluxfix
optimization_call="compute-FBA"
echo -e "GlcT-min\tmin GlcT\tstdef\tGlcT" > simulations
simulate-single GlcT-min
rm constraintfluxes
```

You can visualize the results of the simulation as above.

Pruning

To remove reactions which can not carry a non-zero flux and to fix the directions of fluxes which can not proceed in the other direction you do the following. The system boundaries must be written in the file with the fixed name stdexch.txt.

```
cp stdef.txt stdexch.txt
optimization_call="compute-FBA -F 0"
prune-network
```

Here the most simple flux minimization protocol is used to test whether a reaction can carry a nonzero flux in one of the directions. However it is also possible to combine pruning with any other computational protocol. The result is stored in the directory `sub`:

```
cd sub
less reactions
```

Shlomi

This algorithm maximizes the number of common occurrence of flux and expression. First, we generate an arbitrary profile:

```
gawk '{print $1,rand()}' reactions.original > expressions
```

Constraints and objectives are not used by this algorithm thus, we generate a “dummy” simulation and then run with:

```
optimization_call="compute-shlomi"
echo -e "Shlomi\tShlomi\t\tGlcT" > simulations
simulate
```

Biomass maximization

As opposed to flux minimization where the input of substrates may not be bounded, in biomass maximization this is necessary for the problem to be feasible. We define this with this mechanism:

```
echo GlcT 1 > constraintfluxes
update-fluxbounds
```

Again, we have to define a dummy simulation task because there is no fixed objective here.

```
optimization_call="compute-FBA -b ATPase GSHox"
echo -e “Biomass max\t0\tstdef\tGlcT” > simulations
simulate
```

FVA

As for biomass maximization, for flux-variability analysis it is important to restrict the exchange fluxes otherwise each flux can grow infinitely. Here, the output is fixed:

```
echo -e "ATPase 2.38\nDPGase 0.494\nGSHox 0.093\nPPrPT 0.0258" >
constraintfluxes
make-FVA-simulationsfile > simulations
cp stdef.txt stdexch.txt
optimization_call="compute-FBA"
simulate
rm constraintfluxes
allout2valfiles
```

```
FVA-valfiles-chart
rm -rf val
```