Practical Exercises : Flux Balance Analysis using the python module COBRApy (COnstraint-Based Reconstruction and Analysis)

Hidde de Jong and Nils Giordano

October 21, 2014

This document is largely based on the tutorial accompanying the review by Orth *et al.* (2010), "What is flux balance analysis?", *Nat. Biotechnol.*, 28(3):245-8, as well as a simplified version available at the following address: `https://noppa.aalto.fi/noppa/kurssi/ke-70.4300/viikkoharjoitukset/KE-70_4300_cobra-lisaharjoitus.pdf`.

Evaluation

You will find several questions below that we invite you to answer by writing in the boxes reserved for this. A single scanned document per group must be sent by email in pdf format by the end of the week (26/10/2014, 23:59).

COBRApy

Principle

Over the past few decades, large amounts of data at the genome-scale level have accumulated, allowing the increasingly precise reconstruction of metabolic pathways. At this level of detail, it is almost impossible to obtain an intuitive understanding of how the entire system works without mathematical and computational analysis. Modelling at the genome-scale level may require one to make mathematical abstractions. This is also the case in the approach considered in the practical exercises of this course: Flux Balance Analysis (FBA).

FBA focuses on physicochemical constraints, based on current knowledge, to define the set of feasible flux distributions for a biological network in a given condition. These constraints can include compartmentalization, mass conservation, molecular crowding, and thermodynamic directionality. FBA selects the flux distribution(s) from the set of feasible flux distributions that optimize(s) a particular objective function.

Flux Balance Analysis can be performed using the python module COnstraints Based Reconstruction and Analysis (COBRA) from the openCOBRA project. Its installation is organised around three main components:

- a library to read models in a standardised format (SBML);
- a solver to optimise fluxes (GLPK);

- a module with pre-defined FBA functions (COBRA).

To install COBRApy on your computer, follow the instructions at: http://opencobra.github.io/cobrapy/

COBRA Components

SBML: exchange models in systems biology

Standardisation is often a big problem in the computional sciences. In systems biology, one needs to be able to construct, analyse and exchange models in the same way as one needs to construct, analyse and exchange sequence data in genomics. The systems biology community has developed the Systems Biology Markup Language (SBML) as a standard for publishing models.

The Systems Biology Markup Language (SBML) is an XML-based language that is free, open and benefits from widespread software support and a community of users and developers (http://sbml.org). It can represent many different classes of biological phenomena, including metabolic networks, cell signaling pathways, regulatory networks, infectious diseases, ... Software support is usually based on the SMBL library (http://sbml.org/Software/libSBML).

The LP solver

From a mathematical point of view, the resolution of the optimisation problems in FBA involves linear programming (LP). There exist many LP solvers, three of which can be used by the COBRA module. For these practical exercices, we will use the LP solver in the open-source GNU Linear Programming Kit (GLPK). GLPK is a software package intended for solving large-scale linear programming (LP), mixed integer programming (MIP), and other related problems. It is a set of routines written in ANSI C and organized in the form of a library with functions that can be called from within other programs. Another option is to use Gurobi, which is freely available for academic use (you just need to activate the licence from an academic domain the first time).

The Python module COBRApy

The COBRApy module contains predefined functions for performing FBA and related functionalities on a constraint-based model. A Matlab toolbox is also available with a similar syntax. We chose to use Python for this practical exercise because the software is free and open source, but feel free to go to the openCOBRA website if you want to use the Matlab version at home.

Most of the introduction of this practical exercise is inspired from the documentation available at: https://cobrapy.readthedocs.org/en/latest/. Do not hesitate to use it when you are looking for a particular function.

Getting started

Start by launching an ipython console in a terminal. First, you need to initialize and load a model. We will use the *E. coli* model "iJO1366 central metabolism" (available by default in cobra.test):

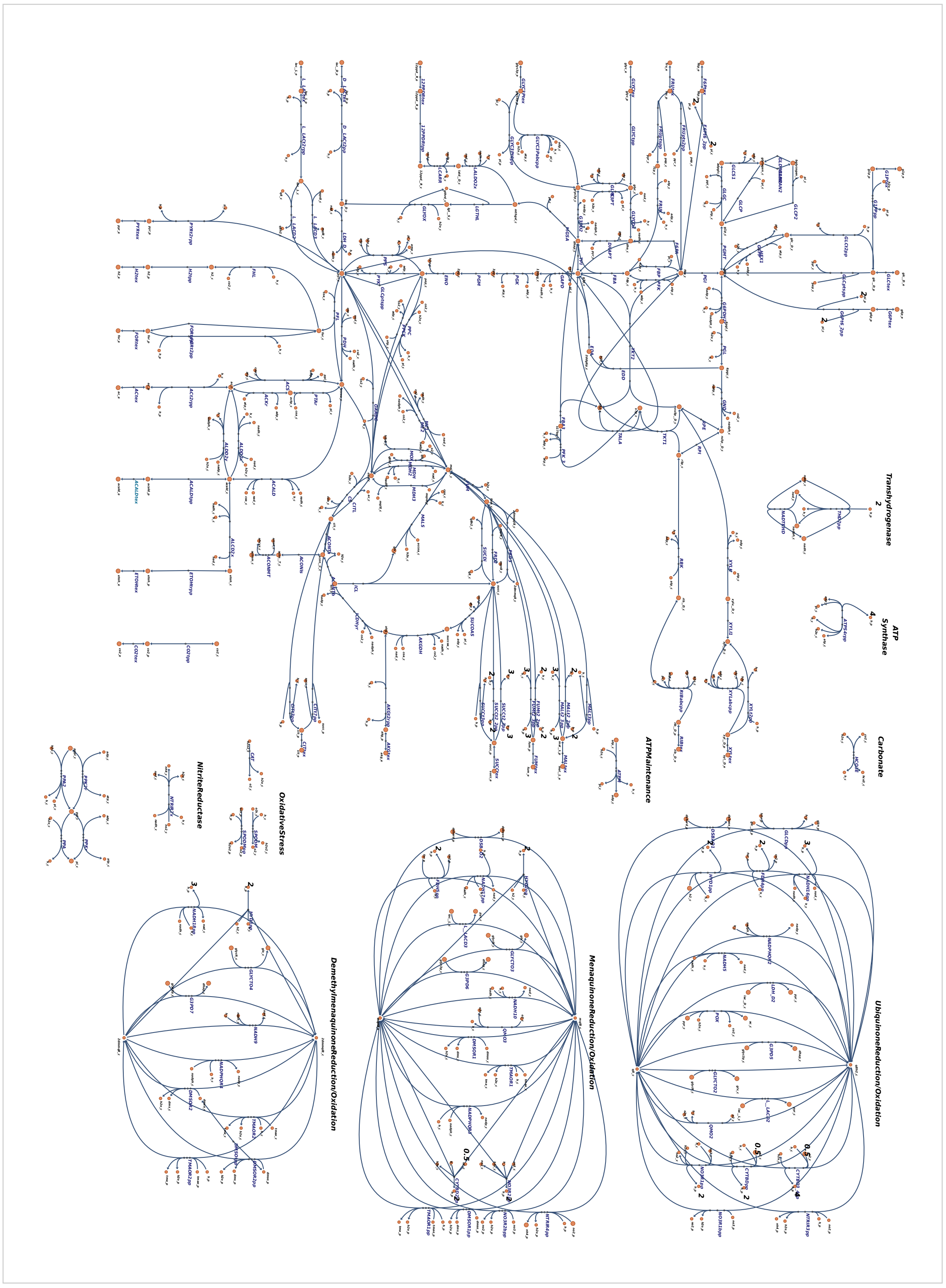

Figure 1: Map of the *E. coli* iJO1366 central metabolism model.

```
import cobra
import cobra.test
model = cobra.io.load_matlab_model(cobra.test.ecoli_mat)
```

For practical reasons we load a Matlab file, but we could have loaded a SMBL file using `cobra.io.read_sbml_model()`. You now have a model object called `model`. You can find all reactions, metabolites, and genes of the model stored in the attributes `reactions`, `metabolites`, and `genes` as lists.

```
print len(model.reactions) # 2546 reactions
print len(model.metabolites) # 1802 metabolites
print len(model.genes) # 1264 genes
```

Each list is made of `Reaction`, `Metabolite` and `Gene` objects.

```
In [x]: model.reactions[29]
Out[x]: <Reaction EX_5dglcn_e at 0x4529f190>
```

If you want to act on a reaction, you will be able to tab-complete the list of elements.

```
# To type what follows,
# try to press tab after model.reactions.EX_gl
print model.reactions.EX_glc_e.name
```

If you want to list all attributes of a `Reaction`, `Metabolite` or `Gene` object, try pressing tab after adding a point.

```
# try to press tab after model.metabolites.etoh_e.
print model.metabolites.etoh_e.name
```

You can also get a list element using its id.

```
In [x]: model.metabolites.get_by_id('etoh_e')
Out[x]: <Metabolite etoh_e at 0x4c323d0>
```

NAMES: . . . .

Exercise 1: Computation of aerobic and anaerobic growth rates

This section will show how to perform basic FBA calculations. We will simulate the growth of *E. coli* with glucose as the sole carbon source under aerobic (high O_2) and anaerobic (low O_2) conditions.

In flux balance analysis we need to define an objective function for the optimisation problem. Objective functions are defined by setting the `objective_coefficient` attribute of a reaction object to a value > 0. Normally, this has already been done in the model provided. To find all the reactions with an `objective_coefficient > 0`, just type the following code.

```
for reaction in model.reactions:
        if reaction.objective_coefficient > 0:
                print reaction
                objective = reaction
```

You should identify the flux towards biomass (also called growth rate in FBA) which is used by default in most models. To inspect its composition, enter:

```
print objective.reaction
```

You can print the full names of the reactants and products in the reaction by typing:

```
for reactant in objective.get_reactants():
        print reactant.name
for product in objective.get_products():
        print product.name
```

Question 1.1

Comment on the composition of the objective function: which general class of metabolites can you identify? Where does the stoichiometry come from?

We will now compute the distribution of fluxes maximizing this function, and thus maximizing the growth rate. We will set the maximum glucose uptake rate to 18.5 mmol gDW^{-1} hr^{-1} (millimoles per gram dry cell weight per hour, the default flux units used in COBRA). Note that by convention, uptake reactions have a negative flux because they are written as export reactions (e.g., `glc_D_e <==>`).

```
# Setting the lower_bound of the uptake of glucose
model.reactions.EX_glc_e.lower_bound = -18.5
```

By doing this, we constrain the glucose uptake rate to a maximum value of 18.5 mmol gDW^{-1} hr^{-1}, a biologically realistic uptake rate.

Growth simulation in different oxygen conditions

We will first allow unlimited oxygen availability by changing the boundaries of the oxygen uptake reaction. To do so, we set the lower bound of the oxygen uptake reaction to a large number (making it practically unbounded).

```
model.reactions.EX_o2_e.lower_bound = -1000
```

We will now try to optimize the objective function in these conditions.

```
model.optimize()
```

The solution is stored in `model.solution`. A `Solution` object has several attributes, including

- `f`: the objective value;
- `status`: the exit status of the LP solver (`'optimal'` if a solution has been found, `'infeasible'` if no solution exists);
- `x_dict`: a dictionary of pairs of reaction identifier and flux value;
- `x`: a list for `x_dict`.

For example, the growth rate (the objective value of the solution) can be printed using the following command:

```
print model.solution.f
```

A good way to analyse the solution is to draw the map of the fluxes using the module escher.

```
import escher
solution = model.solution.x_dict
modelname = "e_coli:iJO1366:central_metabolism"
modelmap = escher.Builder(modelname, reaction_data=solution)
modelmap.display_in_browser()
# Press ctrl+C to continue in the console
```

You can change the size-scale or the color-scale of the arrows in *Map>Settings*. For instance, a good way to visualize which fluxes are non-zero is to set the ladder to [0 0 0.01]. If you want to zoom in on a component of the map (reaction or metabolite), try using *ctrl+F*.

Question 1.2

What is the growth rate predicted by FBA (`model.solution.f`) in aerobic conditions? Which pathways carry significant (non-zero) flux under these conditions?

Next, the same simulation is performed under anaerobic conditions. Keep the same model and disable oxygen uptake.

```
model.reactions.EX_o2_e.lower_bound = 0
```

Now perform the optimization and draw the map as before.

Question 1.3

What is the growth rate predicted under anaerobic conditions? Which pathways carry significant flux under these conditions? In particular, list all the metabolites that are significantly excreted by the bacteria (transport reactions by diffusion are indicated by the suffix −`tex`). Based on what you know of central carbon metabolism, comment on the discrepancies between the solutions in aerobic and anaerobic conditions.

Exercise 2: Growth on alternate substrates

Just as FBA was used to calculate growth rates of *E. coli* on glucose in the previous exercise, it can also be used to simulate growth on other substrates. The *E. coli* model used contains exchange reactions for several organic compounds, each of which can be used as the sole carbon source under aerobic conditions. For example, to simulate growth on succinate instead of glucose, first set the lower bound of the glucose exchange reaction `EX_glc_e` to `0` and the lower bound of the oxygen exchange reaction to −`1000`.

```
model.reactions.EX_glc_e.lower_bound = 0
model.reactions.EX_o2_e.lower_bound = -1000
```

Then set the lower bound of the succinate exchange reaction `EX_succ_e` to −`20` mmol gDW^{-1} hr^{-1} (an arbitrary uptake rate).

```
model.reactions.EX_succ_e.lower_bound = -20
```

You can now solve this new optimisation problem.

```
model.optimize()
```

Question 2.1
What is the growth rate of *E. coli* on succinate in aerobic condition? If you try the same optimisation in anaerobic conditions, what is the new growth rate?

Based on what you learned during the previous exercise, predict the growth rates of *E. coli* on all 13 organic substrates under **both aerobic and anaerobic conditions**. For each, use a substrate uptake rate of 20 mmol gDW^{-1} hr^{-1}. Such a task can be easily performed using a **`for`** loop (tips: look at the `get_by_id()` attribute).

Question 2.2
Fill the following table (substrate names and growth rates).

Substrate		**Growth rate (h^{-1})**	
Name	**Id**	**Aerobic**	**Anaerobic**
	ac_e		
	acald_e		
	akg_e		
	etoh_e		
	fru_e		
	fum_e		
	glc_e		
	gln__L_e		
	glu__L_e		
	lac__D_e		
	mal__L_e		
	pyr_e		
	succ_e		

Question 2.3
What is the highest growth rate? Looking at the maps, can you explain why some substrates cannot be used by *E. coli* in anaerobic conditions?

Exercise 3: Simulation of the maximum growth rate following genetic perturbations

You can perform *in-silico* knock-outs in a genome-scale model by setting the reaction fluxes associated with the genes to 0. For example, the gene responsible for the expression of PCK (phosphoglycerate kinase, an enzyme in glycolysis), can be deleted as follows:

```
# Don't forget to reinitialize the model before doing this
model.reactions.EX_glc_e.lower_bound = -10
model.reactions.EX_o2_e.lower_bound = -1000
model.reactions.PCK.lower_bound = 0
model.reactions.PCK.upper_bound = 0
model.optimize()
print model.solution.f
```

Question 3.1

Does the PCK knock-out strain grow? By looking at the flux map, can you explain what is happening?

You can perform all possible single-gene deletions in the model by using the function `cobra.flux_analysis.single_deletion()`. For each gene in the model, it sets the reaction flux corresponding to the enzyme to 0 and performs the optimization. The function returns a dictionary containing the gene identifier and the computed growth rate, and a dictionary containing the gene identifier and the exit status of the optimization process.

```
grates, status = cobra.flux_analysis.single_deletion(model)
```

Before using the `single_deletion()` function, remember to set the constraints bounding the uptake rate of the substrates. If you want to get a list of all deletions with growth rate $> 0.95\ h^{-1}$, you may execute the following code:

```
listgenes = []
for gene in grates:
        if grates[gene] > 0.95:
                listgenes.append(gene)
```

```
print listgenes
```

Using a similar script, try to answer the following questions.

Question 3.2

In an aerobic environment with glucose as the sole carbon source (uptake rate at most 10 mmol gDW^{-1} hr^{-1}), what is the percentage of single-gene deletions that are lethal? (*i.e.*, with a growth rate < 0.01.) Select one of these genes and explain why you think it is essential.

Question 3.3

What is the percentage of single-gene deletions with no measurable effect on growth? (*i.e.*, with a growth rate $\geq 99\%$ of the wild-type growth rate.)

Question 3.4

The same question as above for anaerobic growth. Comment the differences with the result obtained for aerobic growth.

Exercise 4: Alternate optimal solutions: flux variability analysis

The flux distribution calculated by FBA is often not unique. In many cases, it is possible for a biological system to achieve the same objective value by using alternate pathways, so phenotypically different alternate optimal solutions are possible. A method that uses FBA to identify alternate optimal solutions is Flux Variability Analysis (FVA). This is a method that identifies the maximum and minimum possible fluxes through a particular reaction with the objective value constrained to be close to or equal to its optimal value. Performing FVA on a single reaction using the basic COBRA module functions is simple. First, calculate the maximal growth rate like before. Get the optimal solution (`model.solution.f`), and then set both the lower and upper bounds of the objective reaction to exactly this value. Next, set the reaction of interest as the objective using `model.change_objective()`, and use FBA to minimize and maximize this new objective in two separate steps. This will give the minimum and maximum possible fluxes through this reaction while contributing to the optimal objective value. All the steps described above are performed by the script below, in the case of growth on succinate.

```
model.reactions.EX_glc_e.lower_bound = 0
model.reactions.EX_succ_e.lower_bound = -20
solution = model.optimize()
obj = model.reactions.Ec_biomass_iJO1366_core_53p95M
obj.lower_bound = solution.f
obj.upper_bound = solution.f

model.change_objective('ME1');
ME1_max = model.optimize(objective_sense='maximize')
ME1_min = model.optimize(objective_sense='minimize')

modelname = "e_coli:iJO1366:central_metabolism"
modelmap_max = escher.Builder(modelname,
        reaction_data=ME1_max.x_dict)
modelmap_min = escher.Builder(modelname,
        reaction_data=ME1_min.x_dict)
modelmap_max.display_in_browser()
#ctrl+C
modelmap_min.display_in_browser()
```

Question 4.1

Draw the two alternative flux distributions on the map when the `ME1` reaction is set to either its minimum or maximum possible flux. Comment on the differences between the two flux distributions.

The COBRApy module includes a built-in function for performing global FVA called `cobra.flux_analysis.flux_variability_analysis()`. This function performs the steps described above on every reaction in a model.

Question 4.2

For aerobic growth on succinate, with the maximum uptake rate set to -20 mmol gDW^{-1}

hr^{-1}, what is the percentage of reactions that are completely determined, that is, reactions for which FVA returns strictly equal maximum and minimum fluxes? Comment this result.

Exercice 5: Industrial contest

Like *Saccharomyces cerevisiae*, *E. coli* is capable of producing ethanol from glucose in certain conditions. In this question we aim at maximizing this production rate. Before doing expensive experiments, you want to test *in silico* what could be the best growth conditions and genome modifications to optimize ethanol production. We choose a glucose uptake rate equal to 10 mmol gDW^{-1} hr^{-1} in anaerobic conditions.

Question 5.1

By changing the objective function to ethanol excretion, what is the maximum attainable carbon yield? (*i.e.*, the ratio of the number of carbon atoms included in the ethanol produced versus glucose consumed.) What is the growth rate in this condition? (stored in `model.solution.x_dict`.)

Bacteria do not naturally optimize ethanol production, but are believed to maximize their growth rate. Therefore, in order to find a more realistic solution, we need to find ways to improve ethanol production while the overall objective of the cell remains growth-rate maximization.

Question 5.2

Which conditions can you find that most increase the ethanol excretion rate? Which growth rate is attained in these conditions? What is the carbon yield? You can choose every (combination of) substrates (with a maximum uptake rate of 10 mmol gDW^{-1} hr^{-1} for each substrate). All gene deletions are allowed, but no more than 5 in a single strain.

We have asked you to solve the above question by trial-and-error, but one should be aware that several extensions of standard FBA have been developed to address the above question in a principled and more rigorous way. For example, MOMA (Minimization of Metabolic Adjustment) has been proposed for growth-rate maximization in mutant strains (Burgard *et al.* (2003), *Biotechnol. Bioeng.*, 84(6):647-57), while OptKnock and its derivatives use bilevel optimization methods to find gene deletions that force the production of desired compounds as a byproduct of optimal growth (Segrè *et al.* (2002), *Proc. Natl. Acad. Sci. USA*, 99(23):15112-7). MOMA and OptKnock have been integrated as functions in some of the COBRA implementations. (See for instance `cobra.flux_analysis.moma`.)