Practical Exercises: Flux Balance Analysis using the COnstraint-Based Reconstruction and Analysis (COBRA) Toolbox

Hidde de Jong and Nils Giordano

October 12, 2020

This document is largely based on the tutorial accompanying the review by Orth *et al.* (2010), "What is flux balance analysis?", *Nature Biotechnology*, 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`.

1 The COBRA toolbox

1.1 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. This has also motivated the approach considered in these practical exercises, 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 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 by the COnstraints Based Reconstruction and Analysis (COBRA) toolbox (from the openCOBRA project, see `http://opencobra.sourceforge.net/\linebreakopenCOBRA/`). Its installation is organised around three main components:

- a library to read models in a standardised format;
- a solver to optimise fluxes;

- a toolbox with pre-defined FBA functions.

1.2 COBRA Components

1.2.1 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. It is a computer-readable format that has been widely adopted.

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`), which we will transparently use during the exercises.

1.2.2 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 openCOBRA toolbox. 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).

1.2.3 The COBRA Toolbox

This is a free Matlab package containing predefined functions for performing FBA and related functionalities on a constraint-based model. A Python module is also available with a quite similar syntax. The basic Matlab COBRA functions are listed below.

```
addReaction() % add a new reaction to a model
changeObjective() % change the objective function reaction(s)
changeRxnBounds() % change the upper or lower bounds on reactions
deleteModelGenes() % constrain reactions associated with deleted genes to 0
```

```
drawFlux() % print a flux distribution on a map
findRxnIDs() % get index of a reaction
optimizeCbModel() % perform FBA on a model
printFluxVector() % print the results of an FBA calculation
printRxnFormula() % print the formula of a reaction
readCbModel() % (requires SBML Toolbox) load a model in SBML format
writeCbModel() % (requires SBML Toolbox) write a model in SBML format
```

Exercise 1: Simulation of aerobic and anaerobic growth rates

This section will demonstrate how to perform basic FBA calculations. We will simulate the growth of *E. coli* on glucose under aerobic (high O_2) and anaerobic (low O_2) conditions.

Setting up the model

First, you need to initialize and load the *E. coli* core model:

```
initCobraToolbox
load Ecoli_core_model.mat; % or readCbModel('path/to/model.xml') if
                           % libSMBL is installed
```

Next, to ensure that the biomass reaction is set as the objective function, enter:

```
model = changeObjective(model,'Biomass_Ecoli_core_w_GAM');
```

To inspect the biomass composition in the *E. coli* core model, enter:

```
printRxnFormula(model,'Biomass_Ecoli_core_w_GAM');
```

Question

Comment on the composition of the objective function: which metabolites are involved? Where does the stoichiometry come from?

Next 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 the COBRA Toolbox). Note that by convention, uptake reactions have a negative flux because they are written as export reactions (e.g., `glc_D[e] <==>`).

```
model = changeRxnBounds(model,'EX_glc(e)',-18.5,'l');
```

This changes the lower bound ('l' for 'lower') of the glucose exchange reaction to -18.5, a biologically realistic uptake rate.

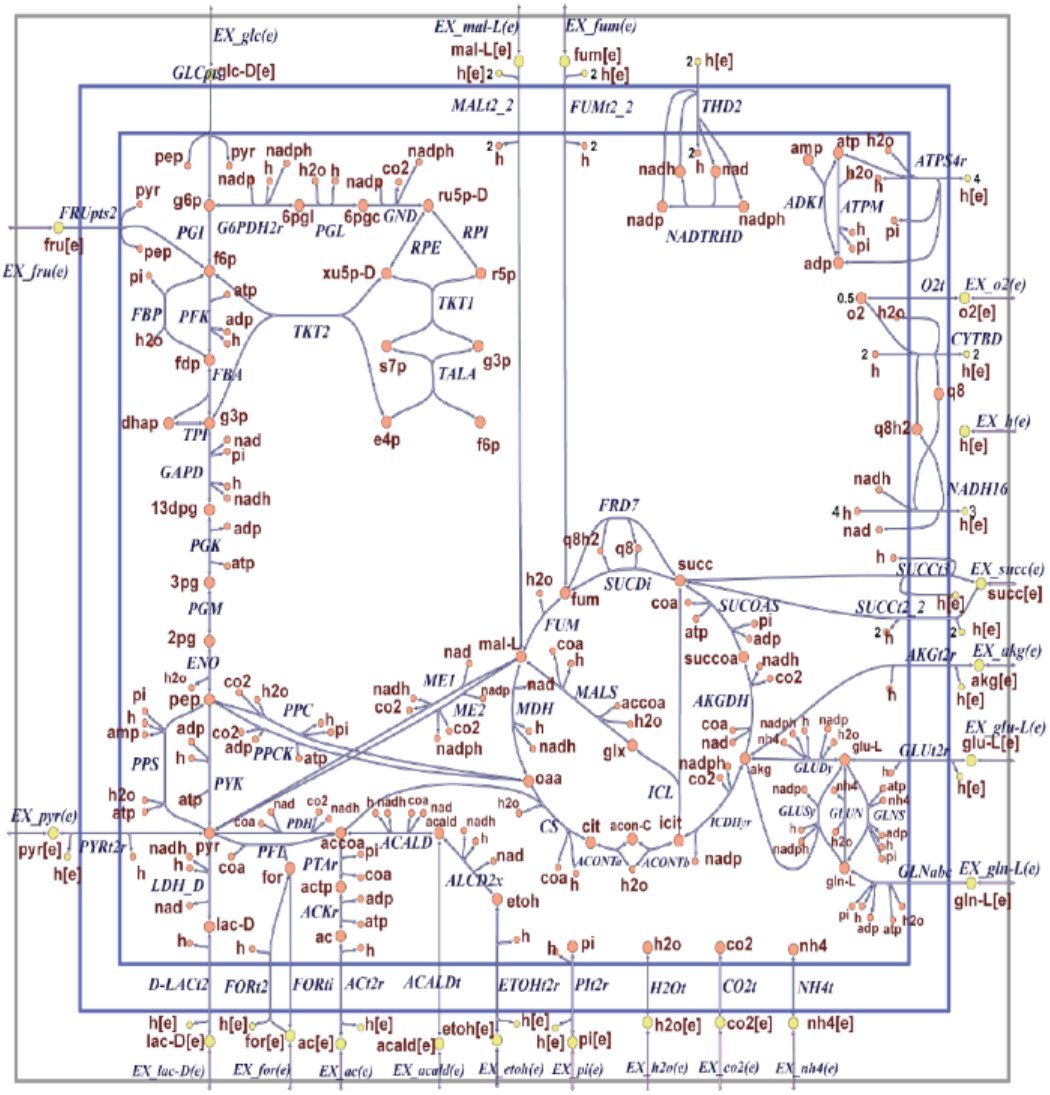

Figure 1: Map of the *E. coli* core metabolic network. Orange circles represent cytosolic metabolites, yellow circles represent extracellular metabolites, and the blue arrows represent reactions. Reaction name abbreviations are uppercase and metabolite name abbreviations are lowercase.

Growth simulation in different oxygen conditions

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

```
model = changeRxnBounds(model,'EX_o2(e)',-1000,'l');
```

We will now perform FBA. Note that we want to maximize the objective function.

```
FBAsolution = optimizeCbModel(model,'max');
```

You can draw the solution on the *E. coli* core map using the following command.

```
map = readCbMap('path/to/the/map.txt'); % location of the map
                                        % in the matlab toolbox directory
options.zeroFluxWidth = 1; % reduce size of 0-flux arrows
options.fileName = 'target.svg'; % choose a file name
drawFlux(map, model, FBAsolution.x, options)
% You can open target.svg with firefox or inkscape
```

Question

What is the growth rate predicted by FBA (`FBAsolution.f`) in aerobic conditions? Which pathways carry significant flux under these conditions? Check the flux vector `FBAsolution.x` and draw the fluxes on the map with `drawFlux()`.

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

```
model = changeRxnBounds(model,'EX_o2(e)',0,'l');
```

Question

What is the growth rate predicted under anaerobic conditions? Which pathways carry significant flux under these conditions? Based on what you know of central carbon metabolism, comment on the discrepancies between the solutions in aerobic and anaerobic conditions (difference in growth rate and fluxes).

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 core *E. coli* model contains

exchange reactions for 13 different 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 use the `changeRxnBounds()` function to set the lower bound of the glucose exchange reaction `EX_glc(e)` to 0.

```
model = changeRxnBounds(model,'EX_glc(e)',0,'l');
```

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

```
model = changeRxnBounds(model,'EX_succ(e)',-20,'l');
```

As in the glucose examples, make sure that `Biomass_Ecoli_core_N(w/GAM)-Nmet2` is set as the objective function, and use `optimizeCbModel()` to perform FBA.

```
FBAsolution = optimizeCbModel(model,'max');
```

Question

What is the growth rate of *E. coli* on succinate both in aerobic an anaerobic conditions?

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}.

Question

Fill the following table.

	Growth rate (h^{-1})	
Substrate	**Aerobic**	**Anaerobic**
ac		
acald		
akg		
etoh		
fru		
fum		
glc-D		
gln-L		
glu-L		
lac-D		
mal-L		
pyr		
succ		

Question

What is the highest growth rate? Is it in accordance with what you know about bacterial growth?

Exercise 3: Production of co-factors and biomass precursors

FBA can also be used to determine the maximum yields of important cofactors and biosynthetic precursors from glucose and other substrates. This can be done by changing the objective function(s). In this example, we will calculate the maximum yields of the cofactors ATP, NADH, and NADPH from glucose under aerobic conditions. Start by constraining the glucose exchange reaction `EX_glc(e)` to exactly -1 mmol gDW^{-1} hr^{-1}.

```
model = changeRxnBounds(model,'EX_glc(e)',-1,'b'); % 'b' = 'both' boundaries
```

To simulate optimal ATP production, set the ATP maintenance reaction (ATPM) as the objective to be maximized using `changeObjective()`.

```
model = changeObjective(model,'ATPM');
```

ATPM is a stoichiometrically balanced reaction that hydrolyses ATP (`atp[c]`) and produces ADP (`adp[c]`) + inorganic phosphate (`pi[c]`) + a proton (`h[c]`). It works as an objective for maximizing ATP production because in order to consume the maximum amount of ATP, the network must first produce ATP by the most efficient pathways available. In order to

maximise it, the constraint on this reaction should be removed by using `changeRxnBounds()` to set the lower bounds to 0. By default, this reaction has a lower bound of 8.39 mmol gDW^{-1} hr^{-1} to simulate non-growth associated maintenance costs.

```
model = changeRxnBounds(model,'ATPM',0,'l');
```

You are now ready to perform the optimisation.

```
FBAsolution = optimizeCbModel(model,'max');
```

Calculation of the yields of NADH and NADPH one at a time can be performed in a similar manner. First, constrain ATPM to 0 mmol gDW^{-1} hr^{-1} flux ('b'), so that the cell is not required to produce ATP and also cannot consume any ATP using this reaction.

```
model = changeRxnBounds(model,'ATPM',0,'l');
```

You will have to add yourself stoichiometrically balanced NADH and NADPH consuming reactions using the function `addReaction()`, and set these as the objectives using `changeObjective()`.

```
model = addReaction(model,'NADH_drain','nadh[c] -> nad[c] + h[c]');
model = changeObjective(model,'NADH_drain');
```

Finally, optimise the model and display the solution.

```
FBAsolution = optimizeCbModel(model,'max');
disp(FBAsolution.f)
```

The same principle can be applied with NADPH.

Question

Fill in the following table with the yields of the three cofactors from glucose under both aerobic and anaerobic conditions.

Co-factor	**Aerobic yield (mol mol glc^{-1})**	**Anaerobic yield (mol mol glc^{-1})**
ATP		
NADH		
NADPH		

The core *E. coli* model contains 12 basic biosynthetic precursor compounds that are used to build macromolecules such as nucleic acids and proteins. The maximum yield of each of these precursor metabolites from glucose can be calculated by adding a demand reaction for each one (a reaction that consumes the compound without producing anything) and setting

these as the objectives for FBA. Note that the drain reactions for acetyl-CoA (`accoa[c]`) and succinyl-CoA (`succoa[c]`) must produce coenzyme-A (`coa[c]`), since this carrier molecule is not produced from glucose in the core model.

Question

Fill in the following table with the maximum yields of each of the 12 precursors from glucose under aerobic conditions, including the percentage of C molecules from glucose present in the final output (the formula for each metabolite can be found in model.metFormulas). For example, a yield of 2 moles of a 3C compound from 1 mole of 6C glucose is 100% carbon conversion.

Precursor	**Yield (mol mol glc^{-1})**	**Carbon conversion**
3pg		
pep		
pyr		
oaa		
g6p		
f6p		
r5p		
e4p		
g3p		
accoa		
akg		
succoa		

Exercise 4: Simulation of the maximum growth rate following genetic perturbations

Using the function `deleteModelGenes()` you can perform an *in silico* knock-out in a genome-scale model. This function constrains reaction associated with the gene to 0.

Question

Determine the flux through the biomass objective function using FBA and the *E. coli* core metabolic network with the given loss of function mutation of the gene(s) and main carbon substrates (table below).

Main carbon substrate	**Substrate uptake rate (mmol gDW^{-1} hr^{-1})**	**Aerobic growth**	**Loss of function (mutation)**	**Flux through (hr^{-1})**
D-glucose	10	Yes	`ackA` (b2296)	
D-lactate	20	Yes	`ackA` (b2296)	
D-oxoglutarate	13	Yes	`ackA` (b2296)	
D-glucose	10	Yes	`pck` (b2403)	
D-oxoglutarate	13	Yes	`pck` (b2403)	
D-glucose	10	Yes	`tpi` (b3919)	
D-lactate	20	Yes	`tpi` (b3919)	
D-glucose	10	Yes	`atpABCDEFGHI` (b3731-b3738)	
D-lactate	20	Yes	`atpABCDEFGHI` (b3731-b3738)	

Exercise 5: 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 Toolbox functions is simple. First, use functions `changeRxnBounds`, `changeObjective`, and `optimizeCbModel` to perform FBA as described in the previous examples. Get the optimal objective value Z (`FBAsolution.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, 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. For example, consider the variability of the malic enzyme reaction (`ME1`) in *E. coli* growing on succinate. The minimum possible flux through this reaction is 0 mmol gDW^{-1} hr^{-1} and the maximum is 6.49 mmol gDW^{-1} hr^{-1}. In one alternate optimal solution, the `ME1` reaction is used, but in another, it is not used at all. The full code to set the model to these conditions and perform FVA on this reaction is given below.

```
model = changeRxnBounds(model,'EX_glc(e)',0,'l');
model = changeRxnBounds(model,'EX_succ(e)',-20,'l');
```

```
FBAsolution = optimizeCbModel(model,'max');
model = changeRxnBounds(model,'Biomass_Ecoli_core_N(w/GAM)-Nmet2',FBAsolution.f,'b');
model = changeObjective(model,'ME1');
FBAsolutionMin = optimizeCbModel(model,'min');
FBAsolutionMax = optimizeCbModel(model,'max');
optmin.fileName = 'min.svg'
optmax.fileName = 'max.svg'
drawFlux(map,model,FBAsolutionMin, optmin);
drawFlux(map,model,FBAsolutionMax, optmax);
```

Question

Draw the two alternative flux distributions on the map of the *E. coli* network when the `ME1` reaction is set to either its minimum or maximum possible flux. Comment on differences between the two fluxes distribution.

The COBRA Toolbox includes a built-in function for performing global FVA called `fluxVariability()`. This function is useful because it performs FVA on every reaction in a model. When FVA is performed on every reaction in the *E. coli* core model for growth on succinate, eight reactions are found to be significantly variable.

Question

Fill in the table below with the minimal and maximal flux values for these 8 reactions.

Reaction	**Minimum flux** (**mmol gDW^{-1} hr^{-1}**)	**Maximum flux** (**mmol gDW^{-1} hr^{-1}**)
FRD7		
MDH		
ME1		
ME2		
NADTRHD		
PPCK		
PYK		
SUCDi		

2 Growth rate contest

The exercises in the previous section have introduced (some of) the functionalities of the COBRA toolbox by means of a core model of *E. coli* metabolism. In this section, we will use a genome-scale model of the same system, called Ec_iAF1260_flux2. Notice that it is a huge model with thousands of reactions, which means that contrary to the core model, it becomes difficult to visualize the flux distributions returned by the LP solver (in fact, no map of this model is currently available).

We will use this genome-scale model to investigate a fundamental question: what is the maximum growth rate that *E. coli* can attain when growing on the carbon sources of Exercise 2? We do not limit the oxygen uptake rate `EX_o2(e)`, but set the maximum substrate uptake rate to 20 mmol gDW^{-1} hr^{-1}. Moreover, we fix the objective function `Ec_biomass_iAF1260_core_59p81M` and the lower bound of the maintenance flux (`ATPM`, 8.39 mmol gDW^{-1} hr^{-1}). You are free to enrich the growth medium with other compounds the cell is able to take up (at a maximum rate of 20 mmol gDW^{-1} hr^{-1}) and to change the genetic background of the cell.