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

Hidde de Jong, Nils Giordano & Daniel Kahn

November 5, 2015

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.

1 General 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 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.gitnet.io/openCOBRA/). 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.

2 COBRA Components

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 SBML library (http://sbml.org/Software/libSBML).

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).

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 quite a similar syntax (maybe a little more object-oriented). We chose to use Matlab in this PE to avoid compatibility issues, but feel free to check on the openCOBRA website if you want to use the Python version at home. All COBRA functions are documented at http://opencobra.github.io/cobratoolbox/docs/index.html, which is also available as your_cobra_directory/docs/index.html. 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('ecoli_core_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

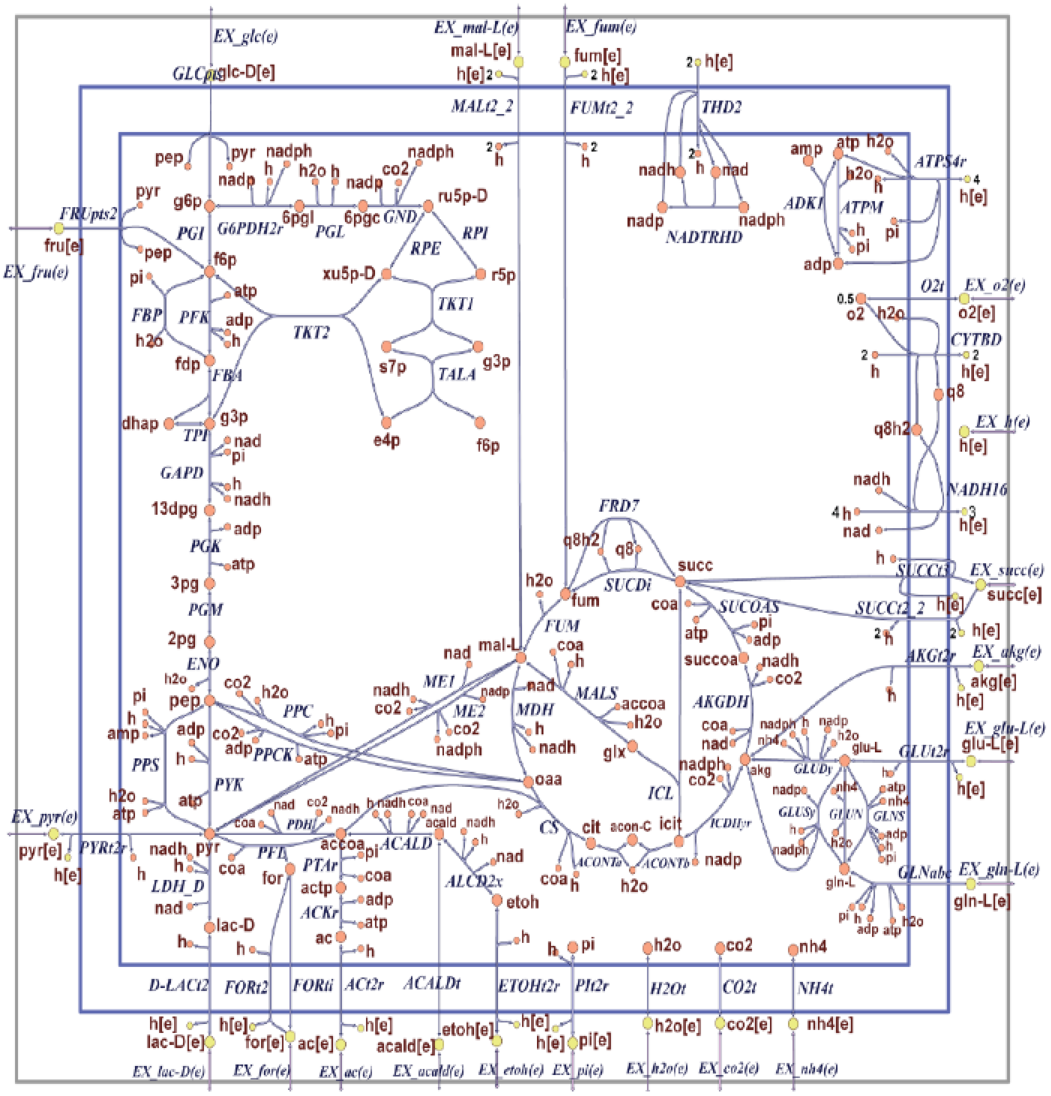

Figure 1: Map of the *E. coli* core metabolic network. Orange circles represent cytosolic metabolites, yellow circles represent extracellular metabolites, and the blue arrows represent reactions. Reaction name abbreviations are uppercase and metabolite name abbreviations are lowercase.

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.

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('ecoli_core_map.txt');
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_w_GAM` 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? Predict also the growth rates on other organic substrates under both aerobic and anaerobic conditions. For each, use a substrate uptake rate of 20 mmol gDW^{-1} hr^{-1}.

	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		

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 ATP 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 optimise the model and display the solution.

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

Question

> Compute and comment the ATP yield per mole of glucose under aerobic and anaerobic conditions.

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

Using the function `changeRxnBounds()` you can block any particular reaction in a genome-scale model by setting both bounds to 0. For example:

```
model = changeRxnBounds(model,'ACKr',0,'b');
```

Question

Determine the *E. coli* growth rate on glucose or lactate using the core metabolic network after the losses of function indicated below.

Main carbon substrate	**Substrate uptake rate (mmol gDW^{-1} hr^{-1})**	**Aerobic growth**	**Loss of function (mutation)**	**Growth rate (hr^{-1})**
D-glucose	10	Yes	`ACKr`	
D-lactate	20	Yes	`ACKr`	
D-glucose	10	Yes	`PPCK`	
D-lactate	20	Yes	`PPCK`	
D-glucose	10	Yes	`TPI`	
D-lactate	20	Yes	`TPI`	
D-glucose	10	Yes	`ATPS4r`	
D-lactate	20	Yes	`ATPS4r`	

Exercise 5: Alternate optimal solutions: flux variability analysis

The flux distribution calculated by FBA is most usually not unique. In most 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. The COBRA Toolbox also includes a built-in function for performing global FVA called `fluxVariability()`. For instance, to run flux variability for growth of *E. coli* on succinate, we use the following commands:

```
model = changeRxnBounds(model,{'EX_glc(e)' 'EX_succ(e)'},[0 -20],'l');
[min_flux max_flux] = fluxVariability(model);
```

Question

Perform FVA on the core *E. coli* model for growth on succinate and fill in the table below with the minimal and maximal flux values. Comment on the multiplicity of FBA solutions.

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