# Primer on Non-FBA based COBRA methods


## Author(s); Jeff Orth, Ines Thiele, Ronan Fleming

## Reviewer(s);


## INTRODUCTION

In this tutorial, several basic examples of how FBA can be used to analyze constraint-based models, as described in [0], are presented. Many more COBRA methods, including some that are covered in this tutorial, are also presented in the COBRA Toolbox v3.0 paper [ref]. A map of the core model is shown in Figure 1. Formal reaction name abbreviations are listed in blue text, formal metabolite name abbreviations are listed in purple text.

## Beginner COBRA methods

Example 9. Analysis of the topological features of the S matrix

Example 10. Characterization of functional states by random sampling

Example 9. Topological features of the S matrix: metabolite connectivity and reaction participation

Visualization of the S matrix

The core E. coli model S matrix can be visualized by using the spy command in Matlab. This command will represent all non-zero entries in S with a dot, as shown in Figure 11. The code to produce this figure is:

```
spy(model.S);
```

Figure 11 The (72*95) S matrix of the E. coli core model. All non-zero entries are marked with a dot.

Determination of the number of reactions the metabolites occur in

The S matrix can be converted into a binary matrix (Sbin), by replacing all non-zero elements in S with a ‘1’ in Sbin. Then, all ones in each row of the Sbin can be summed to determine the number of reactions a metabolite occurs in. The full code to binarize the S matrix and calculate and plot metabolite connectivity is:

```
Sbin = zeros(size(model.S));
Sbin(find(model.S))=1;
```

```
for i = 1 : length(model.mets)
metConnectivity(i,1) = sum(Sbin(i,:));
end

loglog(sort(metConnectivity,'descend'),'*')
xlabel('metabolite number (rank ordered) - log scale');
ylabel('number of reactions - log scale')
```

As Figure 12 shows, there are very few metabolites that are highly connected, while most metabolites participate only in a few reactions. The approximate linear appearance of the curve of connectivities is surprising and corresponds to a power law distribution of metabolite connectivity. The few highly connected metabolites are “global” players, similar to hubs in protein-protein-interaction networks, while the low connectivity metabolites are “local” players, many of which only occur in linear pathways. The power law distribution indicates that the networks are scale-free17, 18.

Figure 12 Connectivity of the core E. coli metabolites (loglog plot).

Reaction participation is the number of metabolites per reaction

To determine reaction participation for every reaction, the number of non-zero elements per column in Sbin is counted:

```
for i = 1 : length(model.rxns)
rxnParticipation(i,1) = sum(Sbin(:,i));
end
```

In the E. coli core model, there are on average 3.8 metabolites per reaction (mean(rxnParticipation)). The most common type of reaction in the E. coli core metabolic network is the bi-linear reaction involving two substrates and two products.

How many metabolites are co-occurring in two reactions?

To determine the number of metabolites common to two reactions, one needs to calculate the compound adjacency matrix (Acomp); Acomp = Sbin * SbinT. This multiplication leads to a square matrix of size m*m, where m is the number of metabolites in the network.

```
Acomp = Sbin*Sbin';
```

The diagonal elements correspond to the metabolite connectivity (computed above), and can be extracted as follows:

```
metConnectivity = diag(Acomp);
```

The off-diagonal elements correspond to the number of reactions two metabolites are co-occurring in. The cofactor pair nad[c] and nadh[c] occurs in 12 reactions together. Moreover, the connectivity of both metabolites is 12, thus, they only occur as pairs in the core E. coli model.

Is there any correlation between reaction essentiality and metabolite connectivity19?

One way to address this question is to delete all reactions in the network that are associated with a metabolite and ask if the network can still produce biomass. To find all reactions associated with a metabolite, one has to just scan through the corresponding rows in the S matrix.  Then the associated genes can be knocked out, and FBA can be used to predict if growth is possible.  The full code to perform these calculations and plot the results is:

```
growthDel = zeros(length(model.mets),1);
for i = 1 : length(model.mets)
rxnID = find(model.S(i,:));
for j = 1 : length(rxnID)
modelDel = deleteModelRxn(model,model.rxns(rxnID(j)));
FBAsolutionDel = optimizeCbModel(modelDel);
if FBAsolutionDel.f <= 1e-6
growthDel(i,1) = growthDel(i,1) + 1;
end
```

```
end
end

lethalityFraction = growthDel./metConnectivity;
semilogx(metConnectivity,lethalityFraction,'*')
xlabel('metabolite connectivity - log scale');
ylabel('average lethality fraction');
```

The results for the E. coli core model (Figure 13) show that some less connected metabolites have a higher lethality fraction than highly connected metabolites. This has been found to be true for other, more complex metabolic networks19. In fact, for the E. coli core model, the average lethality fraction lies between 0.2 and 0.5 for the majority of the metabolites, regardless of their connectivity. There are a few compounds that have a connectivity of 2 and have a lethality fraction of 1 (e.g., cit[c], glc-D[e]). These metabolites often occur in a linear pathway, at the end of which an essential biomass precursor is produced.

Figure 13 Correlation between metabolite connectivity and average lethality of reactions producing or consuming a particular metabolite. Less connected metabolites tend to occur in a higher fraction of essential reactions.

Example 10. Characterization of functional states by random sampling

How probable is a flux through a reaction?

While flux variability analysis determines the minimum and maximum value of a reaction flux given some network constraints, Monte-Carlo-sampling can be used to determine a probability flux distribution for each network reaction20, 21.

Sampling is an unbiased method, compared to many other biased methods in constraint-based modeling. Monte-Carlo-sampling involves a random walk from a starting point through the entire space. As the network size increases the number of steps required to reach all regions of the high-dimensional space will increase, making unbiased sampling more time-consuming. Consequently, a slightly biased approach (hit-and-run sampling) has been adapted for sampling of metabolic networks21 that provides the sampling algorithm with a set of ‘warm up points’ (warmupPts)which are randomly generated within the feasible steady-state solution space. The hit-and-run sampler determines the geometric center (cg) of the high-dimensional solution space. Using this geometric center, the direction from a point x to the next step is biased by choosing a point x1

along the line of x and cg, whereby the step size to x1 is randomly determined. This approach has been shown to accelerate the random walk through the space. To ensure that the set of sampling points does indeed represent the solution space, i) there must be sufficient points (i.e., as the size of the solution space increases the number of stored points must increase, nFiles*pointsPerFile gives the total number of stored points to the ACHRSampler); and ii) all points must be random, i.e., the chosen path between two subsequent sampling points has to be untraceable. Therefore, the number of points between two stored points (stepsPerPoint) must be chosen appropriately. For this latter property, it is also the case that the number of steps per points must increase as the dimensionality of the steady-state solution space increases. Note that the following calculations can be quite time consuming, even with a small scale model.

To sample the solution space of the E. coli core model with its default constraints (growth on glucose, aerobic);

```
warmupPts= createHRWarmupRand(model,500);
ACHRSampler(model,warmupPts,'samplingResults',10,5000,1000,warmupPts(:,1));
```

To the load the calculated data:

```
samples = loadSamples('samplingResults', 10, 5000);
```

Next, FVA (Example 3) can be used to determine the minimum and maximum possible fluxes so the sampling results can be plotted for the first nine reactions in the model (Figure 14);

```
[minFlux,maxFlux] = FluxVariability(model,0);

figure;
for i = 1 : 9
subplot(3,3,i)
hist(samples(i,:),10);
hold on
plot([minFlux(i) maxFlux(i)], [0 1],'*r');
title(model.rxns{i});
end
```

The error of Sv = 0 for the sampled flux vectors can be calculated, and these errors are plotted in Figure 15. The errors are all very small, indicating that these sampling results are valid:

```
errors = max(abs(model.S*samples));
plot(errors)
```

The correlation between the first ten network reactions can be calculated and plotted (Figure 16) with:

```
sampleScatterMatrix(model.rxns(1:10),model,samples);
```

This scatter plot allows us to visualize the interaction between two network reactions. For example, the aconitase reactions ACONTa and ACONTb are perfectly correlated in the tested condition (sample points between the two reactions align on a line). In contrast, the flux through the adenylate kinase reaction (ADK1) seems mostly independent from the flux through the acetaldehyde dehydrogenase reaction (ACALD). Considering the locations of these reactions on the metabolic map of the E. coli core model (Figure 1), this observation is not astonishing.

Which network reactions are perfectly correlated? Once the samples points have been calculated they can readily used to determine the correlation between any two reactions in the network. To obtain the list of perfectly correlated reactions (e.g., as in the example above, ACONTa and ACONTb), the following commands can be used:

```
[setsSorted,setNoSorted,setSize] = identifyCorrelSets(model,samples);
setNames = [];
setNumbers = [];
for i = 1 : length(setsSorted)
setNames = [setNames; setsSorted{i}.names];
setNumbers = [setNumbers; i*ones(length(setsSorted{i}.names),1)];
end
```

Two vectors will be returned (setNames, setNumbers) which can be copied into a spreadsheet to obtain a table view (see Table 6). Interestingly, only 37 of the 95 network reactions are grouped into 17 perfectly correlated reaction sets. A main reason for this is that we did not require the growth rate to have a minimum value. Hence, many functional states are possible, some of which support growth while others don´t.

Figure 14  Histograms of sampled fluxes for nine reactions.

Figure 15  Error of Sv = 0 for the sampled fluxes.

Figure 16  Scatter matrix of sample points in the E. coli core model, simulated in minimal medium glucose under aerobic conditions.

Table 6: Correlated reaction sets in the E. coli core model, when grown on glucose minimal medium, aerobic conditions. Reactions that are not part of a correlated reaction set with more than one member are not listed.

| Correlated Set # | Reaction Name | Correlated Set # | Reaction Name | Correlated Set # | Reaction Name |
|---|---|---|---|---|---|
| 1 | CYTBD | 6 | GND | 12 | EX_for(e) |
| 1 | EX_o2(e) | 6 | PGL | 12 | PFL |

1 O2t 7 GAPD 13 ENO

2 ACONTa 7 PGK 13 PGM

2 ACONTb 8 FBA 14 D_LACt2

2 CS 8 TPI 14 LDH_D

3 ACKr 9 EX_pi(e) 15 CO2t

3 ACt2r 9 PIt2r 15 EX_co2(e)

3 PTAr 10 EX_nh4(e) 16 ALCD2x

4 TALA 10 NH4t 16 ETOHt2r

4 TKT1 11 EX_h2o(e) 17 ADK1

5 ICL 11 H2Ot 17 PPS

5 MALS