unsteady-state Flux Balance Analysis (uFBA)

James T. Yurkovich

Department of Bioengineering and the Bioinformatics and Systems Biology Program, University of California, San Diego USA

Reviewed by Aarash Bordbar

INTRODUCTION

In this tutorial, we will use unsteady-state Flux Balance Analysis (uFBA) [1] to integrate exo- and endo-metabolomics data [2] into a constraint-based metabolic model for the human red blood cell [3]. The uFBA method allows for bypassing the steady-state assumption for intracellular metabolites that are measured.

We can model the flux through a metabolic network using a set of linear equations defined by

$$\mathbf{S} \cdot \mathbf{v} = \mathbf{b}$$

where **S** is the stoichiometric matrix, **v** is a vector of fluxes through the chemical reactions defined in **S**, and **b** represents constraints on the change of metabolite concentrations; at steady-state, **b** = 0. If the metabolomics measurements are non-linear (i.e., Fig. 1), then the first step of the uFBA workflow is to identify discrete time intervals which represent linearized metabolic states (Fig. 1). Once discrete states are identified (the raw data if linear), we proceed to estimating metabolite concentration rates of change. For each metabolic state, we can use linear regression to calculate the rate of change of each metabolite concentration. If the rate of change is significant, the model is updated by changing the steady-state constraint from 0 to

$$\mathbf{S} \cdot \mathbf{v} \geq \mathbf{b}_1$$
$$\mathbf{S} \cdot \mathbf{v} \leq \mathbf{b}_2$$

where $[\mathbf{b}_1, \mathbf{b}_2]$ represents the 95% confidence interval for each significantly changing metabolite. All unmeasured metabolites are assumed to be at steady-state (i.e., $\mathbf{b}_1 = \mathbf{b}_2 = 0$).

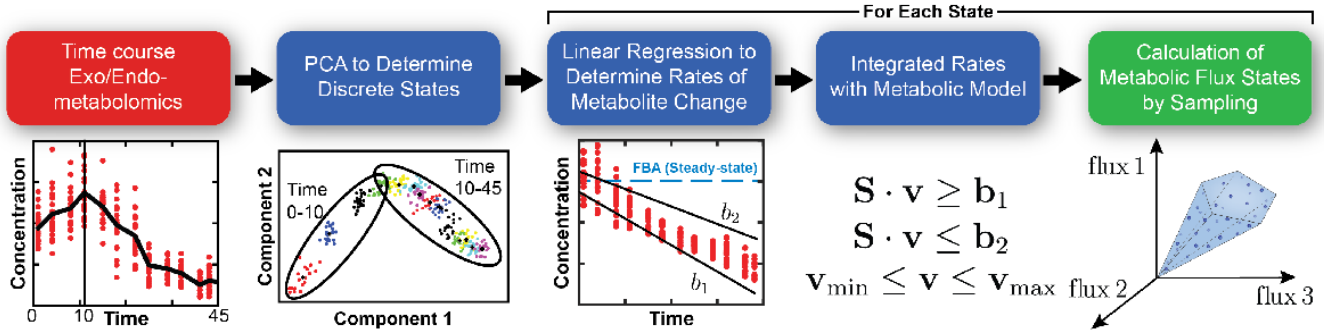

Fig. 1 | Overview of the uFBA workflow.

MATERIALS

Equipment Setup

Running uFBA requires the installation of a mixed-integer linear programming solver. We have used Gurobi 7.0.0 (http://www.gurobi.com/downloads/download-center) which is freely available for academic

use (this workflow has only been tested with Gurobi solvers; use other solvers at your own risk). This tutorial uses the Statistics Toolbox to perform linear regression (if the Statistics Toolbox is not installed, compute linear regression manually; see `testUFBA.m`).

PROCEDURE

Initialize

Running uFBA requires the use of several functions from the COBRA Toolbox.

```
initCobraToolbox
```

```
     _____   _____   _____   _____      _____      |
    /  ___| /  _  \ |  _  \ |  _  \    /  ___ \     |   COnstraint-Based Reconstruction and Analysis
    | |     | | | | | |_| | | |_| |    | |___| |    |   The COBRA Toolbox - 2017
    | |     | | | | |  _  { |  _  /    |  ___  |    |
    | |___  | |_| | | |_| | | | \ \    | |   | |    |   Documentation:
    \_____| \_____/ |_____/ |_|  \_\   |_|   |_|    |   http://opencobra.github.io/cobratoolbox
                                                     |

> Checking if git is installed ...  Done.
> Checking if the repository is tracked using git ...  Done.
> Checking if curl is installed ...  Done.
> Checking if remote can be reached ...  Done.
> Initializing and updating submodules ... Done.
> Adding all the files of The COBRA Toolbox ...  Done.
> Define CB map output... set to svg.
> Retrieving models ...   Done.
> TranslateSBML is installed and working properly.
> Configuring solver environment variables ...
  - [-*--] ILOG_CPLEX_PATH: /Users/syarra/Applications/IBM/ILOG/CPLEX_Studio1271/cplex/matlab
  - [*---] GUROBI_PATH: /Library/gurobi702/mac64/matlab
  - [----] TOMLAB_PATH :  --> set this path manually after installing the solver ( see instructions )
  - [----] MOSEK_PATH :  --> set this path manually after installing the solver ( see instructions )
  Done.
> Checking available solvers and solver interfaces ... Done.
> Setting default solvers ... Done.
> Saving the MATLAB path ... Done.
  - The MATLAB path was saved in the default location.

> Summary of available solvers and solver interfaces

  Support     LP   MILP     QP   MIQP    NLP
-----------------------------------------------------------------------
cplex_direct  full              0      0      0      0      -
dqqMinos      full              1      -      -      -      -
glpk          full              1      1      -      -      -
gurobi        full              1      1      1      1      -
ibm_cplex     full              1      1      1      -      -
matlab        full              1      -      -      -      1
mosek         full              0      0      0      -      -
pdco          full              1      -      1      -      -
quadMinos     full              1      -      -      -      1
tomlab_cplex  full              0      0      0      0      -
qpng          experimental      -      -      1      -      -
tomlab_snopt  experimental      -      -      -      -      0
gurobi_mex    legacy            0      0      0      0      -
lindo_old     legacy            0      -      -      -      -
lindo_legacy  legacy            0      -      -      -      -
lp_solve      legacy            1      -      -      -      -
opti          legacy            0      0      0      0      0
-----------------------------------------------------------------------
```

```
Total         -                 8      3      4      1      2

+ Legend: - = not applicable, 0 = solver not compatible or not installed, 1 = solver installed.


> You can solve LP problems using: 'dqqMinos' - 'glpk' - 'gurobi' - 'ibm_cplex' - 'matlab' - 'pdco' - '
> You can solve MILP problems using: 'glpk' - 'gurobi' - 'ibm_cplex'
> You can solve QP problems using: 'gurobi' - 'ibm_cplex' - 'pdco' - 'qpng'
> You can solve MIQP problems using: 'gurobi'
> You can solve NLP problems using: 'matlab' - 'quadMinos'

> Checking for available updates ...
--> You cannot update your fork using updateCobraToolbox(). [1085e9 @ tutorial_uFBA].
    Please use the MATLAB.devTools (https://github.com/opencobra/MATLAB.devTools).
```

We first load in sample data. This data is absolutely quantified and has already been volume adjusted such that intracellular and extracellular metabolite concentrations have compatible units.

```
global CBTDIR
load([CBTDIR filesep 'tutorials' filesep 'uFBA' filesep 'sample_data.mat']);
```

The `sample_data.mat` file contains the following variabels:

- `met_data`: a matrix containing the exo- and endo-metabolomics data
- `met_IDs`: a cell array containing the BiGG ID for each of the metabolites in `met_data`
- `model`: a modified version [3] of the iAB-RBC-283 COBRA model structure
- `time`: a vector of the time points (in days) at which the metabolite concentrations were measured
- uFBA`variables`: a struct containing the variables necessary for input into the uFBA algorithm

In this tutorial, the use of Gurobi is mandatory.

```
solverLPOk = changeCobraSolver('gurobi', 'LP');
```

```
> Gurobi interface added to MATLAB path.
```

```
solverMILPOk = changeCobraSolver('gurobi', 'MILP');
```

```
> Gurobi interface added to MATLAB path.
```

Estimate Metabolite Rates of Change (<1 sec.)

Next, we run linear regression to find the rate of change for each metabolite concentration.

```
changeSlopes = zeros(length(met_IDs), 1);
changeIntervals = zeros(length(met_IDs), 1);
for i = 1:length(met_IDs)
    [tmp1, tmp2] = regress(met_data(:, i), [time ones(length(time), 1)], 0.05);
    changeSlopes(i, 1) = tmp1(1);
    changeIntervals(i, 1) = abs(changeSlopes(i, 1) - tmp2(1));
end
```

The variables `changeSlopes` and `changeIntervals` contain the metabolite rates of change and 95% confidence intervals, respectively. We will create a new vector, `ignoreSlopes`, which contains a 0 for the metabolites whose slopes change significantly and a 1 otherwise:

```
tmp1 = changeSlopes - changeIntervals;
tmp2 = changeSlopes + changeIntervals;
ignoreSlopes = double(tmp1 < 0 & tmp2 > 0);
```

Integration of Metabolomics Data (<10 min.)

Finally, we need to input the data into the uFBA algorithm which is encapsulated in the function `buildUFBAmodel`. This function takes as input a COBRA model structure and a struct containing the required input variables (see Table 1).

Ideally, all metabolites in the model would be measured, resulting in a feasible model. However, experimental limitations limit the number of metabolites that can measured. Thus, when the metabolite constraints are added, the model will most likely not simulate. The uFBA algorithm reconciles the measured metabolomics data and the network structure by parsimoniously allowing unmeasured metabolites concentrations to deviate from steady-state (i.e., $\mathbf{S \cdot v = b}$) in order to build a computable model. We refer to the method for deviating unmeasured metabolites from steady-state as "metabolite node relaxation." As part of this procedure, free exchange of extracellular metabolites out of the system is only allowed if (1) the metabolite concentration is measured to be increasing or (2) if the relaxation of a particular extracellular metabolite is required for model feasibility.

There are five different techniques built into the uFBA method to perform the node relaxation. The technique used in this tutorial is an MILP optimization that minimizes the number of unmeasured metabolites relaxaed from steady-state; this choice effectively minimizes the changes made to the model in order to achieve feasibilitiy. Full details for this and all other node relaxation techniques can be found in [1]. Sinks are added for each of the relaxed metabolite nodes, and the flux through each of these sinks is minimized while still allowing the model to simulate. The minimimum value is then multiplied by a relaxation factor `lambda` (Table 1) and used as the bound for the sink reaction.

Full details for the algorithm are provided in the original publication [1].

Required Inputs	Description
model	A COBRA model structure containing (at minimum) the following fields: S, b, lb, ub, mets, rxns
metNames	A cell array containing the model IDs of the measured metabolites that will have bounds set by the algorithm. These metabolites should correspond to model.mets. Note: measured metabolites that were not significantly changed over time should also be included.
changeSlopes	A vector (length(metNames) x 1) that contains the mean rate of change (the slope from linear regression) for each metabolite in metNames.
changeIntervals	A vector (length(metNames) x 1) that contains the difference between the mean slope of change and the upper bound of the 95% confidence interval for each slope in changeSlopes.
ignoreSlopes	A binary vector (length(metNames) x 1) that instructs which changeSlopes to be ignored (ignore if 1). Metabolites were ignored if the values of the slopes were not significant based on linear regression (i.e., if slope value +/- the interval crossed zero).
Optional Inputs	**Description**
objRxn	The objective reaction (corresponding to model.rxns) for the new uFBA model. Default is the objective reaction from the original model.
metNoSink	A cell array of metabolites (corresponding to model.mets) that should not have a sink added, typically for metabolites where the concentration is known to be zero. Default is an empty cell array.
metNoSinkUp	A cell array of metabolites (corresponding to model.mets) that should not have a sink added in the up direction (which would allow metabolite accumulation). Default is an empty cell array.
metNoSinkDown	A cell array of metabolites (corresponding to model.mets) that should not have a sink added in the down direction (which would allow metabolite depletion). Default is an empty cell array.
conflictingMets	A cell array of intracellular metabolites (corresponding to model.mets) where the intracellular rates conflict with extracellular rates, and the model cannot compensate through biosynthesis of the metabolite or use of the flux in other pathways. Typically only necessary for very simple cell types (e.g., RBCs). The intracellular rate is adjusted to the extracellular to allow the model to simulate. Default is an empty cell array.
solvingStrategy	One of {'case1','case2','case3','case4','case5'} which correspond to the 5 node relaxation techniques discussed in the methods section of [1]. Default value is the first LP technique, 'case2'.
lambda	A multiplicative relaxation away from the minimum allowed deviation from the steady-state model. Default value is 1.5.
numIterations	The number of iterations for the integer cut optimization method. Default value is 100.
timeLimit	The time limit for the solver during the numIterations optimization loop. Default value is 30 seconds.
eWeight	A weighting factor for preferential selection of extracellular sinks over intracellular during node relaxation. Default value is 1e4. If no weigthing is preferred, eWeight should be set to a value of 1.
Outputs	**Description**
model	The final uFBA model.
metsToUse	Metabolites for which metabolomics data was integrated.
relaxedNodes	A cell array which contains three columns: (1) which metabolites were relaxed from steady-state; (2) the direction of the relaxation (accumulation/depletion); and (3) the upper bound of the added sink.

Table 1 | Inputs and outputs of the `buildUFBAmodel` function.

```
uFBAvariables.metNames = met_IDs;
uFBAvariables.changeSlopes = changeSlopes;
uFBAvariables.changeIntervals = changeIntervals;
uFBAvariables.ignoreSlopes = ignoreSlopes;

uFBAoutput = buildUFBAmodel(model, uFBAvariables);
```

```
sink_ascb-L[c] ascb-L[c]  <=>
```

```
sink_gthrd[e] gthrd[e]  <=>
sink_urate[e] urate[e]  <=>
sink_10fthf[c]_up 10fthf[c]_G + 10fthf[c]_L  ->
sink_13dpg[c]_up 13dpg[c]_G + 13dpg[c]_L  ->
sink_2kmb[c]_up 2kmb[c]_G + 2kmb[c]_L  ->
sink_35cgmp[c]_up 35cgmp[c]_G + 35cgmp[c]_L  ->
sink_35cgmp[e]_up 35cgmp[e]_G + 35cgmp[e]_L  ->
sink_5mdr1p[c]_up 5mdr1p[c]_G + 5mdr1p[c]_L  ->
sink_5mdru1p[c]_up 5mdru1p[c]_G + 5mdru1p[c]_L  ->
sink_6pgl[c]_up 6pgl[c]_G + 6pgl[c]_L  ->
sink_ac[c]_up ac[c]_G + ac[c]_L  ->
sink_accoa[c]_up accoa[c]_G + accoa[c]_L  ->
sink_adn[c]_up adn[c]_G + adn[c]_L  ->
sink_adn[e]_up adn[e]_G + adn[e]_L  ->
sink_akg[c]_up akg[c]_G + akg[c]_L  ->
sink_akg[e]_up akg[e]_G + akg[e]_L  ->
sink_ala-L[e]_up ala-L[e]_G + ala-L[e]_L  ->
sink_ametam[c]_up ametam[c]_G + ametam[c]_L  ->
sink_arg-L[e]_up arg-L[e]_G + arg-L[e]_L  ->
sink_asn-L[e]_up asn-L[e]_G + asn-L[e]_L  ->
sink_band[c]_up band[c]_G + band[c]_L  ->
sink_bandmt[c]_up bandmt[c]_G + bandmt[c]_L  ->
sink_ca2[c]_up ca2[c]_G + ca2[c]_L  ->
sink_ca2[e]_up ca2[e]_G + ca2[e]_L  ->
sink_camp[c]_up camp[c]_G + camp[c]_L  ->
sink_camp[e]_up camp[e]_G + camp[e]_L  ->
sink_cl[c]_up cl[c]_G + cl[c]_L  ->
sink_co2[c]_up co2[c]_G + co2[c]_L  ->
sink_co2[e]_up co2[e]_G + co2[e]_L  ->
sink_coa[c]_up coa[c]_G + coa[c]_L  ->
sink_cys-L[c]_up cys-L[c]_G + cys-L[c]_L  ->
sink_cys-L[e]_up cys-L[e]_G + cys-L[e]_L  ->
sink_cytd[c]_up cytd[c]_G + cytd[c]_L  ->
sink_cytd[e]_up cytd[e]_G + cytd[e]_L  ->
sink_dhap[c]_up dhap[c]_G + dhap[c]_L  ->
sink_dhdascb[c]_up dhdascb[c]_G + dhdascb[c]_L  ->
sink_dhdascb[e]_up dhdascb[e]_G + dhdascb[e]_L  ->
sink_dhmtp[c]_up dhmtp[c]_G + dhmtp[c]_L  ->
sink_dkmpp[c]_up dkmpp[c]_G + dkmpp[c]_L  ->
sink_e4p[c]_up e4p[c]_G + e4p[c]_L  ->
sink_for[c]_up for[c]_G + for[c]_L  ->
sink_fru[c]_up fru[c]_G + fru[c]_L  ->
sink_fum[c]_up fum[c]_G + fum[c]_L  ->
sink_g1p[c]_up g1p[c]_G + g1p[c]_L  ->
sink_g3p[c]_up g3p[c]_G + g3p[c]_L  ->
sink_gdp[c]_up gdp[c]_G + gdp[c]_L  ->
sink_gln-L[e]_up gln-L[e]_G + gln-L[e]_L  ->
sink_glucys[c]_up glucys[c]_G + glucys[c]_L  ->
sink_gly[c]_up gly[c]_G + gly[c]_L  ->
sink_gly[e]_up gly[e]_G + gly[e]_L  ->
sink_gsn[c]_up gsn[c]_G + gsn[c]_L  ->
sink_gsn[e]_up gsn[e]_G + gsn[e]_L  ->
sink_gtp[c]_up gtp[c]_G + gtp[c]_L  ->
sink_gua[c]_up gua[c]_G + gua[c]_L  ->
sink_h2o2[c]_up h2o2[c]_G + h2o2[c]_L  ->
sink_h2o2[e]_up h2o2[e]_G + h2o2[e]_L  ->
sink_h2o[c]_up h2o[c]_G + h2o[c]_L  ->
sink_h2o[e]_up h2o[e]_G + h2o[e]_L  ->
sink_h[c]_up h[c]_G + h[c]_L  ->
sink_h[e]_up h[e]_G + h[e]_L  ->
sink_hco3[c]_up hco3[c]_G + hco3[c]_L  ->
sink_hco3[e]_up hco3[e]_G + hco3[e]_L  ->
sink_hcys-L[c]_up hcys-L[c]_G + hcys-L[c]_L  ->
sink_hcys-L[e]_up hcys-L[e]_G + hcys-L[e]_L  ->
sink_his-L[e]_up his-L[e]_G + his-L[e]_L  ->
sink_icit[c]_up icit[c]_G + icit[c]_L  ->
sink_ile-L[e]_up ile-L[e]_G + ile-L[e]_L  ->
sink_k[c]_up k[c]_G + k[c]_L  ->
```

```
sink_leu-L[c]_up leu-L[c]_G + leu-L[c]_L  ->
sink_leu-L[e]_up leu-L[e]_G + leu-L[e]_L  ->
sink_lys-L[e]_up lys-L[e]_G + lys-L[e]_L  ->
sink_man6p[c]_up man6p[c]_G + man6p[c]_L  ->
sink_met-L[c]_up met-L[c]_G + met-L[c]_L  ->
sink_met-L[e]_up met-L[e]_G + met-L[e]_L  ->
sink_methf[c]_up methf[c]_G + methf[c]_L  ->
sink_mlthf[c]_up mlthf[c]_G + mlthf[c]_L  ->
sink_na1[c]_up na1[c]_G + na1[c]_L  ->
sink_nad[c]_up nad[c]_G + nad[c]_L  ->
sink_nadh[c]_up nadh[c]_G + nadh[c]_L  ->
sink_nadp[c]_up nadp[c]_G + nadp[c]_L  ->
sink_nadph[c]_up nadph[c]_G + nadph[c]_L  ->
sink_nh3[c]_up nh3[c]_G + nh3[c]_L  ->
sink_nh3[e]_up nh3[e]_G + nh3[e]_L  ->
sink_nh4[c]_up nh4[c]_G + nh4[c]_L  ->
sink_nh4[e]_up nh4[e]_G + nh4[e]_L  ->
sink_o2[c]_up o2[c]_G + o2[c]_L  ->
sink_o2[e]_up o2[e]_G + o2[e]_L  ->
sink_o2s[c]_up o2s[c]_G + o2s[c]_L  ->
sink_oaa[c]_up oaa[c]_G + oaa[c]_L  ->
sink_orn[c]_up orn[c]_G + orn[c]_L  ->
sink_phe-L[e]_up phe-L[e]_G + phe-L[e]_L  ->
sink_phpyr[c]_up phpyr[c]_G + phpyr[c]_L  ->
sink_pi[c]_up pi[c]_G + pi[c]_L  ->
sink_pi[e]_up pi[e]_G + pi[e]_L  ->
sink_ppi[c]_up ppi[c]_G + ppi[c]_L  ->
sink_prpp[c]_up prpp[c]_G + prpp[c]_L  ->
sink_ptrc[c]_up ptrc[c]_G + ptrc[c]_L  ->
sink_ptrc[e]_up ptrc[e]_G + ptrc[e]_L  ->
sink_pyr[e]_up pyr[e]_G + pyr[e]_L  ->
sink_r1p[c]_up r1p[c]_G + r1p[c]_L  ->
sink_s7p[c]_up s7p[c]_G + s7p[c]_L  ->
sink_sbt-D[c]_up sbt-D[c]_G + sbt-D[c]_L  ->
sink_ser-L[e]_up ser-L[e]_G + ser-L[e]_L  ->
sink_spmd[c]_up spmd[c]_G + spmd[c]_L  ->
sink_spmd[e]_up spmd[e]_G + spmd[e]_L  ->
sink_sprm[c]_up sprm[c]_G + sprm[c]_L  ->
sink_sprm[e]_up sprm[e]_G + sprm[e]_L  ->
sink_thf[c]_up thf[c]_G + thf[c]_L  ->
sink_thr-L[e]_up thr-L[e]_G + thr-L[e]_L  ->
sink_trp-L[e]_up trp-L[e]_G + trp-L[e]_L  ->
sink_tyr-L[e]_up tyr-L[e]_G + tyr-L[e]_L  ->
sink_udpgal[c]_up udpgal[c]_G + udpgal[c]_L  ->
sink_urea[c]_up urea[c]_G + urea[c]_L  ->
sink_urea[e]_up urea[e]_G + urea[e]_L  ->
sink_val-L[e]_up val-L[e]_G + val-L[e]_L  ->
sink_xmp[c]_up xmp[c]_G + xmp[c]_L  ->
sink_10fthf[c]_down  -> 10fthf[c]_G + 10fthf[c]_L
sink_13dpg[c]_down  -> 13dpg[c]_G + 13dpg[c]_L
sink_2kmb[c]_down  -> 2kmb[c]_G + 2kmb[c]_L
sink_35cgmp[c]_down  -> 35cgmp[c]_G + 35cgmp[c]_L
sink_35cgmp[e]_down  -> 35cgmp[e]_G + 35cgmp[e]_L
sink_5mdr1p[c]_down  -> 5mdr1p[c]_G + 5mdr1p[c]_L
sink_5mdru1p[c]_down  -> 5mdru1p[c]_G + 5mdru1p[c]_L
sink_6pgl[c]_down  -> 6pgl[c]_G + 6pgl[c]_L
sink_ac[c]_down  -> ac[c]_G + ac[c]_L
sink_accoa[c]_down  -> accoa[c]_G + accoa[c]_L
sink_adn[c]_down  -> adn[c]_G + adn[c]_L
sink_akg[c]_down  -> akg[c]_G + akg[c]_L
sink_ala-L[e]_down  -> ala-L[e]_G + ala-L[e]_L
sink_ametam[c]_down  -> ametam[c]_G + ametam[c]_L
sink_arg-L[e]_down  -> arg-L[e]_G + arg-L[e]_L
sink_asn-L[e]_down  -> asn-L[e]_G + asn-L[e]_L
sink_band[c]_down  -> band[c]_G + band[c]_L
sink_bandmt[c]_down  -> bandmt[c]_G + bandmt[c]_L
sink_ca2[c]_down  -> ca2[c]_G + ca2[c]_L
sink_ca2[e]_down  -> ca2[e]_G + ca2[e]_L
```

```
sink_camp[c]_down  -> camp[c]_G + camp[c]_L
sink_camp[e]_down  -> camp[e]_G + camp[e]_L
sink_cl[c]_down  -> cl[c]_G + cl[c]_L
sink_co2[c]_down  -> co2[c]_G + co2[c]_L
sink_co2[e]_down  -> co2[e]_G + co2[e]_L
sink_coa[c]_down  -> coa[c]_G + coa[c]_L
sink_cys-L[c]_down  -> cys-L[c]_G + cys-L[c]_L
sink_cys-L[e]_down  -> cys-L[e]_G + cys-L[e]_L
sink_cytd[c]_down  -> cytd[c]_G + cytd[c]_L
sink_cytd[e]_down  -> cytd[e]_G + cytd[e]_L
sink_dhap[c]_down  -> dhap[c]_G + dhap[c]_L
sink_dhdascb[c]_down  -> dhdascb[c]_G + dhdascb[c]_L
sink_dhdascb[e]_down  -> dhdascb[e]_G + dhdascb[e]_L
sink_dhmtp[c]_down  -> dhmtp[c]_G + dhmtp[c]_L
sink_dkmpp[c]_down  -> dkmpp[c]_G + dkmpp[c]_L
sink_e4p[c]_down  -> e4p[c]_G + e4p[c]_L
sink_for[c]_down  -> for[c]_G + for[c]_L
sink_fru[c]_down  -> fru[c]_G + fru[c]_L
sink_fum[c]_down  -> fum[c]_G + fum[c]_L
sink_g1p[c]_down  -> g1p[c]_G + g1p[c]_L
sink_g3p[c]_down  -> g3p[c]_G + g3p[c]_L
sink_gdp[c]_down  -> gdp[c]_G + gdp[c]_L
sink_gln-L[e]_down  -> gln-L[e]_G + gln-L[e]_L
sink_glucys[c]_down  -> glucys[c]_G + glucys[c]_L
sink_gly[c]_down  -> gly[c]_G + gly[c]_L
sink_gly[e]_down  -> gly[e]_G + gly[e]_L
sink_gsn[c]_down  -> gsn[c]_G + gsn[c]_L
sink_gtp[c]_down  -> gtp[c]_G + gtp[c]_L
sink_gua[c]_down  -> gua[c]_G + gua[c]_L
sink_h2o2[c]_down  -> h2o2[c]_G + h2o2[c]_L
sink_h2o2[e]_down  -> h2o2[e]_G + h2o2[e]_L
sink_h2o[c]_down  -> h2o[c]_G + h2o[c]_L
sink_h2o[e]_down  -> h2o[e]_G + h2o[e]_L
sink_h[c]_down  -> h[c]_G + h[c]_L
sink_h[e]_down  -> h[e]_G + h[e]_L
sink_hco3[c]_down  -> hco3[c]_G + hco3[c]_L
sink_hco3[e]_down  -> hco3[e]_G + hco3[e]_L
sink_hcys-L[c]_down  -> hcys-L[c]_G + hcys-L[c]_L
sink_hcys-L[e]_down  -> hcys-L[e]_G + hcys-L[e]_L
sink_his-L[e]_down  -> his-L[e]_G + his-L[e]_L
sink_icit[c]_down  -> icit[c]_G + icit[c]_L
sink_ile-L[e]_down  -> ile-L[e]_G + ile-L[e]_L
sink_k[c]_down  -> k[c]_G + k[c]_L
sink_leu-L[c]_down  -> leu-L[c]_G + leu-L[c]_L
sink_leu-L[e]_down  -> leu-L[e]_G + leu-L[e]_L
sink_lys-L[e]_down  -> lys-L[e]_G + lys-L[e]_L
sink_man6p[c]_down  -> man6p[c]_G + man6p[c]_L
sink_met-L[c]_down  -> met-L[c]_G + met-L[c]_L
sink_met-L[e]_down  -> met-L[e]_G + met-L[e]_L
sink_methf[c]_down  -> methf[c]_G + methf[c]_L
sink_mlthf[c]_down  -> mlthf[c]_G + mlthf[c]_L
sink_na1[c]_down  -> na1[c]_G + na1[c]_L
sink_nad[c]_down  -> nad[c]_G + nad[c]_L
sink_nadh[c]_down  -> nadh[c]_G + nadh[c]_L
sink_nadp[c]_down  -> nadp[c]_G + nadp[c]_L
sink_nadph[c]_down  -> nadph[c]_G + nadph[c]_L
sink_nh3[c]_down  -> nh3[c]_G + nh3[c]_L
sink_nh3[e]_down  -> nh3[e]_G + nh3[e]_L
sink_nh4[c]_down  -> nh4[c]_G + nh4[c]_L
sink_nh4[e]_down  -> nh4[e]_G + nh4[e]_L
sink_o2[c]_down  -> o2[c]_G + o2[c]_L
sink_o2[e]_down  -> o2[e]_G + o2[e]_L
sink_o2s[c]_down  -> o2s[c]_G + o2s[c]_L
sink_oaa[c]_down  -> oaa[c]_G + oaa[c]_L
sink_orn[c]_down  -> orn[c]_G + orn[c]_L
sink_phe-L[e]_down  -> phe-L[e]_G + phe-L[e]_L
sink_phpyr[c]_down  -> phpyr[c]_G + phpyr[c]_L
sink_pi[c]_down  -> pi[c]_G + pi[c]_L
```

```
sink_pi[e]_down  -> pi[e]_G + pi[e]_L
sink_ppi[c]_down  -> ppi[c]_G + ppi[c]_L
sink_prpp[c]_down  -> prpp[c]_G + prpp[c]_L
sink_ptrc[c]_down  -> ptrc[c]_G + ptrc[c]_L
sink_ptrc[e]_down  -> ptrc[e]_G + ptrc[e]_L
sink_pyr[e]_down  -> pyr[e]_G + pyr[e]_L
sink_r1p[c]_down  -> r1p[c]_G + r1p[c]_L
sink_s7p[c]_down  -> s7p[c]_G + s7p[c]_L
sink_sbt-D[c]_down  -> sbt-D[c]_G + sbt-D[c]_L
sink_ser-L[e]_down  -> ser-L[e]_G + ser-L[e]_L
sink_spmd[c]_down  -> spmd[c]_G + spmd[c]_L
sink_spmd[e]_down  -> spmd[e]_G + spmd[e]_L
sink_sprm[c]_down  -> sprm[c]_G + sprm[c]_L
sink_sprm[e]_down  -> sprm[e]_G + sprm[e]_L
sink_thf[c]_down  -> thf[c]_G + thf[c]_L
sink_thr-L[e]_down  -> thr-L[e]_G + thr-L[e]_L
sink_trp-L[e]_down  -> trp-L[e]_G + trp-L[e]_L
sink_tyr-L[e]_down  -> tyr-L[e]_G + tyr-L[e]_L
sink_udpgal[c]_down  -> udpgal[c]_G + udpgal[c]_L
sink_urea[c]_down  -> urea[c]_G + urea[c]_L
sink_urea[e]_down  -> urea[e]_G + urea[e]_L
sink_val-L[e]_down  -> val-L[e]_G + val-L[e]_L
sink_xmp[c]_down  -> xmp[c]_G + xmp[c]_L
```

The output contains the resulting model (`uFBAoutput.model`):

```
model_ufba = optimizeCbModel(uFBAoutput.model)
```

```
model_ufba =
           x: [231×1 double]
           f: 0.2296
           y: [432×1 double]
           w: [231×1 double]
        stat: 1
    origStat: 'OPTIMAL'
      solver: 'gurobi'
        time: 0.0092
           v: [231×1 double]
```

References

[1] A Bordbar*, JT Yurkovich*, G Paglia, O Rolfsson, O Sigurjonsson, and BO Palsson. "Elucidating dynamic metabolic physiology through network integration of quantitative time-course metabolomics." *Sci. Rep.* (2017). doi:10.1038/srep46249. (* denotes equal contribution)

[2] A Bordbar, PI Johansson, G Paglia, SJ Harrison, K Wichuk, M Magnusdottir, S Valgeirsdottir, M Gybel-Brask, SR Ostrowski, S Palsson, O Rolfsson, OE Sigurjonsson, MB Hansen, S Gudmundsson, and BO Palsson. "Identified metabolic signature for assessing red blood cell unit quality is associated with endothelial damage markers and clinical outcomes." *Transfusion* (2016). doi:10.1111/trf.13460.

[3] A Bordbar, D McCloskey, DC Zielinski, N Sonnenschein, N Jamshidi, and BO Palsson. "Personalized Whole-Cell Kinetic Models of Metabolism for Discovery in Genomics and Pharmacodynamics." *Cell Systems* (2015). doi:10.1016/j.cels.2015.10.003.