Flux Balance Analysis (FBA)

Author(s): Vanja Vlasov, Marouen Ben Guebila, Systems Biochemistry Group, LCSB, University of Luxembourg,

Thomas Pfau, Systems Biology Group, LSRU, University of Luxembourg

Reviewer(s): Ines Thiele, Systems Biochemistry Group, LCSB, University of Luxembourg

Thomas Pfau, Systems Biology Group, LSRU, University of Luxembourg

INTRODUCTION

Flux balance analysis (FBA), one of the most used modelling approaches for metabolic systems, evaluates the metabolic flux distribution [1].

Applications of FBA for molecular systems biology include prediction of the growth rates, uptake rates, knockout lethality and product secretion. In FBA, the solution space is constrained by the assumption of a steady-state, under which each internal metabolite is consumed at the same rate as it is produced.

For the quantitative estimation of the metabolic fluxes, linear programming (LP) can be used to solve the stoichiometric matrix for a given objective function under different constraints. The constraints of the problem depict the space of all eligible possibilities from which an optimal solution can be selected.

$$\begin{array}{ll} \min\limits_{v} & c^T v \\ \text{s.t.} & Sv = b, \\ & l \leq v \leq u, \end{array}$$

where $c \in \Re^n$ is a parameter vector that linearly combines one or more reaction fluxes to form what is termed the objective function, and where a $b_i < 0$, or $b_i > 0$, represents some fixed output, or input, of the ith molecular species. $S \in \Re^{m \times n}$ is a stoichiometric matrix for m molecular species and n reactions, and b is a vector of known metabolic exchanges. The output of FBA is a particular flux distribution, v , which maximises or minimises the objective function and stands between upper and lower bounds, u and l, respectively.

There are multiple different variants of FBA which will be discussed here:

- **Standard FBA**
- **Sparse FBA**
- **Metabolite dilution FBA (mdFBA)**
- **Geometric FBA**
- **Parsimonious enzyme usage Flux Balance Analysis (pFBA)**
- **Dynamic FBA**
- **Relax FBA**
- **Flux enrichment analysis (FEA)**

EQUIPMENT SETUP

If necessary, initialise 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 :  --> set this path manually after installing the solver ( see instructions
   - [*---] GUROBI_PATH: /opt/gurobi702/linux64/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 as ~/pathdef.m.

 > 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              0      0      0      -      -
 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         -                 7      2      3      1      2

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


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

 > Checking for available updates ...
```

```
> The COBRA Toolbox is up-to-date.
```

For solving linear programming problems in FBA analysis, certain solvers are required:

```
changeCobraSolver ('gurobi', 'all', 1);
```

```
> CBT_LP_SOLVER has been set to gurobi.
> CBT_MILP_SOLVER has been set to gurobi.
> CBT_QP_SOLVER has been set to gurobi.
> CBT_MIQP_SOLVER has been set to gurobi.
> CBT_NLP_SOLVER has been set to gurobi.
```

The present tutorial can run with `'glpk'` package, which does not require additional installation and configuration. Although, for the analysis of large models is recommended to use the `'gurobi'` package. For detail information, refer to the solver installation guide: https://github.com/opencobra/cobratoolbox/blob/master/docs/source/installation/solvers.md

PROCEDURE

Before proceeding with the simulations, the path for the model needs to be set up:

```
pathModel = '~/work/sbgCloud/data/models/unpublished/Recon3D_models/';
filename = '2017_04_28_Recon3d.mat';
load([pathModel, filename])
model = modelRecon3model;
clear modelRecon3model
```

In this tutorial, the provided model is a generic model of the human cellular metabolism, Recon 3D [2]. Therefore, we assume, that the cellular objectives include energy production or optimisation of uptake rates and by-product secretion for various physiological functions of the human body.

Standard FBA

Standard FBA allows prediction of a cellular objective for a given set of constraints. These constraints can include e.g. uptake and release limits, or minimal and maximal reaction fluxes.

TIMING

The time to determine a FBA solution depends on the size of the genome-scale model and is commonly less than a second for a medium sized model.

- *Calculating maximal ATP energy production under aerobic condition*

For each new simulation, the original model will be copied to a new variable. This preserves the constraints of the original model and allows to perform simulations with new constraints. Additionally, this method of renaming the model avoids confusion while performing multiple simulations at the same time.

```
modelaerobic = model;
```

The ATP demand reaction, i.e., `DM_atp_c_` within the model is a reaction that involves hydrolysis of ATP to ADP, Pi and proton in the cytosol.

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

```
DM_atp_c_ h2o[c] + atp[c]  -> h[c] + adp[c] + pi[c]
```

We will set this reaction as our objective with the `'changeObjective'` command. Maximising the flux through this reaction will result in the remaining network producing a maximal amount of ATP (up to the limit of the reaction).

```
modelaerobic = changeObjective (modelaerobic, 'DM_atp_c_');
```

The glucose and oxygen, in this case, are provided in high amounts for calculating the flux through ATP demand.

The `'changeRxnBounds'` command changes the lower ('`l`'), upper ('u'), or both the bounds ('b') for the specified reaction. Here we fix the glucose uptake to 20 μmol/min/g of proteins and allow a maximal uptake of oxygen of 1000 μmol/min/g of proteins. The oxygen uptake flux is basically unconstrained.

```
modelaerobic = changeRxnBounds (modelaerobic, 'EX_glc_D[e]', -20, 'l');
modelaerobic = changeRxnBounds (modelaerobic, 'EX_o2[e]', -1000, 'l');
```

The `optimizeCbModel()` calculates one of the optimal solutions within the defined solution space, wherein, the output can be either maximum or minimum flux through the defined objective, whichever is desired. In the above example, the maximal flux through the `DM_atp_c_` is desired.

```
FBAaerobic = optimizeCbModel (modelaerobic, 'max')
```

```
FBAaerobic = 
           x: [10600×1 double]
           f: 1000
           y: [5835×1 double]
           w: [10600×1 double]
        stat: 1
    origStat: 'OPTIMAL'
      solver: 'gurobi'
        time: 0.3890
           v: [10600×1 double]
```

ANTICIPATED RESULTS

When oxygen and all external and internal carbon sources are provided and open in the model of the human cell metabolism, the flux through ATP demand reaction is high 1000 μmol/min/g of proteins.

TROUBLESHOOTING

If there are multiple energy sources available in the model; Specifying more constraints is necessary. If we do not do that, we will have additional carbon and oxygen energy sources available in the cell and the maximal ATP production.

To avoid this issue, all external carbon sources need to be closed.

```
%Closing the uptake of all energy and oxygen sources
idx=strmatch('Exchange/demand reaction',model.subSystems);
c=0;
for i=1:length(idx)
```

```
    if model.lb(idx(i))~=0
        c=c+1;
        uptakes{c}=model.rxns{idx(i)};
    end
end

modelalter = model;
modelalter = changeRxnBounds(modelalter, uptakes, 0, 'b');
modelalter = changeRxnBounds(modelalter, 'EX_HC00250[e]', -1000, 'l');

% The alternative way to do that, in case you were using another large model,
% that does not contain defined Subsystem is
% to find uptake exchange reactions with following codes:
% [selExc, selUpt] = findExcRxns(model);
% uptakes = model.rxns(selUpt);
% Selecting from the exchange uptake reactions those
% which contain at least 1 carbon in the metabolites included in the reaction:
% subuptakeModel = extractSubNetwork(model, uptakes);
% hiCarbonRxns = findCarbonRxns(subuptakeModel,1);
% Closing the uptake of all the carbon sources
% modelalter = model;
% modelalter = changeRxnBounds(modelalter, hiCarbonRxns, 0, 'l');
```

- *Calculating maximum ATP energy production under anaerobic condition*

```
modelanaerobic = modelalter;
modelanaerobic = changeRxnBounds(modelanaerobic, 'EX_glc_D[e]',-20,'l');
modelanaerobic = changeRxnBounds (modelanaerobic, 'EX_o2[e]', 0, 'l');
modelanaerobic = changeObjective(modelanaerobic,'DM_atp_c_');
FBAanaerob = optimizeCbModel(modelanaerobic,'max')
```

```
FBAanaerob = 
           x: [10600×1 double]
           f: 40.0000
           y: [5835×1 double]
           w: [10600×1 double]
        stat: 1
    origStat: 'OPTIMAL'
      solver: 'gurobi'
        time: 0.1144
           v: [10600×1 double]
```

ANTICIPATED RESULTS

Comparing to the aerobic condition, anaerobic condition with only glucose as an energy source has reduced flux through ATP demand (40 μmol/min/g of proteins), signifying the need to oxygen to run the oxidative phosphorylation.

Sparse FBA

Sparse modelling finds the relatively small number of most predictive variables in high-dimensional data sets. Sparse FBA minimises the number of reactions by keeping same maximal objective.

$$\begin{array}{ll}\min\limits_{v} & \\ \text{s.t.} & Sv = b, \\ & l \leq v \leq u, \\ & c^T v = \rho^* \end{array}$$

where the last constraint is optional and represents the requirement to satisfy an optimal objective value ρ^* derived from any solution to a FBA problem. This approach is used to check for minimal sets of reactions that either should be active or should not be active in a flux balance model that is representative of a biochemical network.

```
% [vSparse, sparseRxnBool, essentialRxnBool]  = sparseFBA(model, osenseStr,...
%  checkMinimalSet, checkEssentialSet, zeroNormApprox)

% As an optional input there are different appoximation types
% of zero-norm (only available when minNorm = 'zero')
% (default = 'cappedL1').
%
%  * 'cappedL1' : Capped-L1 norm
%  * 'exp'      : Exponential function
%  * 'log'      : Logarithmic function
%  * 'SCAD'     : SCAD function
%  * 'lp-'      : :math:`L_p` norm with :math:`p < 0`
%  * 'lp+'      : :math:`L_p` norm with :math:`0 < p < 1`
%  * 'l1'       : L1 norm
%  * 'all'      : try all approximations and return the best result
```

TIMING

The time to determine a sparseFBA solution depends on the size of the genome-scale model and is taking from < 1 second for a 1,000 reaction model, to < 2 seconds for a model with more than 10,000 reactions.

```
modelspar = modelalter;
modelspar = changeRxnBounds(modelspar, 'EX_glc_D[e]',-20,'l');
modelspar = changeRxnBounds (modelspar, 'EX_o2[e]', 0, 'l');
modelspar = changeObjective(modelspar, 'DM_atp_c_');
[vSparse, sparseRxnBool, essentialRxnBool] = sparseFBA(modelspar, 'max');
```

```
csense is not defined. We assume that all constraints are equalities.
---FBA
Obj =40
|vFBA|_0 =1212
Comp. time =0.45
---Non-convex approximation---
40 = sparse LP objective.
Error ||c^T*v - f*||^2=0
83 of these are heuristically minimal rxns.
83 of these are essential rxns.
```

ANTICIPATED RESULTS

Commonly, a sparse FBA solution will have much smaller number of active reactions comparing standard FBA on the same model with same objective function.

Additional outputs of the `sparseFBA` function are `sparseRxnBool` and `essentialRxnBool`, which return vectors with 1 and 0's, with sparse and essential reactions respectively.

Display the sparse flux solution, but only the non-zero fluxes.

```
for i=1:length(vSparse)
    if vSparse(i)~=0
        fprintf('%10d \t %s\n', vSparse(i), modelspar.rxns{i})
    end
end
```

```
-2.666667e+01    4ABUTtm
-2.666667e+01    ABTArm
4.444444e+00    AKGDm
-1.111111e+01    AKGt4_3
-4.444444e+00    ALATA_L
4.444444e+00    CITtam
8.888889e+00    DNDPt18m
8.888889e+00    FADH2tx
-8.888889e+00    FADtx
1.111111e+01    FOLt2
4.444444e+00    FUMtm
2.222222e+01    G3PD2m
1.777778e+01    GCC2am
1.777778e+01    GCC2bim
1.777778e+01    GCC2cm
2.000000e+01    GLCt1r
4.444444e+00    GLUt2m
1.777778e+01    GLYtm
2.222222e+01    H2CO3Dm
-4.444444e+00    H2Otm
1.777778e+01    HPYRRy
-1.777778e+01    Htx
2.222222e+01    PCm
-2.222222e+01    SUCD1m
-4.444444e+00    SUCOASm
1.777778e+01    THFtm
1.777778e+01    TRIOK
-8.888889e+00    URIDK2m
-3.555556e+01    EX_HC00250[e]
1.777778e+01    r0027
2.222222e+01    r0081
1.777778e+01    r0160
-2.666667e+01    r0178
-3.555556e+01    r0193
2.000000e+01    r0280
2.000000e+01    r0355
8.888889e+00    r0818
1.777778e+01    r0822
-1.777778e+01    r0838
8.888889e+00    r0853
-3.555556e+01    r0940
2.666667e+01    r1144
2.222222e+01    r1434
```

```
4.444444e+00   r2472
-8.888889e+00   RE1519X
8.888889e+00   RE3347C
1.777778e+01   GLYSNAT5tc
4.444444e+00   DHAPtc
1.777778e+01   DM_Lcystin
-1.111111e+01   FOLOAT1tc
4.444444e+00   r0202m
-2.222222e+01   MALOAtm
-1.777778e+01   MLTHFtm
2.222222e+01   DM_akg[c]
-1.777778e+01   ASPTA
4.000000e+01   DM_atp_c_
5.777778e+01   ENO
2.000000e+01   FBA
-4.444444e+00   FUM
-2.666667e+01   G3PD1
5.777778e+01   GAPD
-1.777778e+01   GHMT2r
-5.111111e+01   H2Ot
2.000000e+01   PGI
-5.777778e+01   PGK
-5.777778e+01   PGM
2.000000e+01   TPI
8.888889e+00   URIDK3
1.777778e+01   GLYt2r
-4.444444e+00   MDH
2.000000e+01   PFK
3.777778e+01   PYK
-1.777778e+01   r0392
2.666667e+01   GGTe_1
3.555556e+01   CYSGLYPTASEe_1
-2.666667e+01   EX_CE5026[e]
-8.888889e+00   EX_5cysgly34dhphe[e]
3.555556e+01   EX_CE1261[e]
8.888889e+00   HMR_3321
1.777778e+01   HMR_3831
4.444444e+00   HMR_4957
-2.000000e+01   EX_glc_D[e]
2.666667e+01   DM_4abut[c]
```

Metabolite dilution flux balance analysis (mdFBA)

This is a variant of FBA for predicting metabolic flux distributions by accounting for growth-associated dilution of all metabolites in a context-dependent manner [3]. A solution from mdFBA guarantees, that all metabolites used in any reaction of the solution can either be produced by the network or taken up from the surrounding medium.

TIMING

Since this is a MIXED Integer Problem it can take a long time to solve.

- *Calculating ATP energy production under aerobic condition with the mdFBA*

In this function, there is an optional output `newActives`, that represent reactions that are only active in this analysis.

```
modelmd = model;
modelmd = changeRxnBounds(modelmd, 'EX_glc_D[e]',-20,'l');
modelmd = changeRxnBounds (modelmd, 'EX_o2[e]', 0, 'l');
modelmd = changeObjective(modelmd, 'DM_atp_c_');

[sol, newActives] = mdFBA(modelmd)
```

```
sol = 
          obj: 1000
       solver: 'gurobi'
         stat: 1
     origStat: 'OPTIMAL'
         time: 61.3177
         full: [10600×1 double]
```

TROUBLESHOOTING

When a model does not have a feasible solution, we are adding an input: `'getInvalidSolution', true`.

```
modelnosol = modelalter;
modelnosol = changeObjective(modelnosol, 'DM_atp_c_');
[sol, newActives] = mdFBA(modelnosol,  'getInvalidSolution', true)
```

```
sol = 
         cont: []
          int: []
          obj: []
       solver: 'gurobi'
         stat: 0
     origStat: 'INFEASIBLE'
         time: 0.3063
         full: []

newActives =

  0×0 empty cell array
```

Geometric FBA

The constraint-based models contain often internal flux loops that are not biologically relevant, though that provides flux distributions that have correct mathematical objectives.

Above mentioned issue can be solved with the geometric FBA approach, which resolves one single solution of the flux distribution. It finds the smallest frame that contains all sets of optimal FBA solutions and posts a set of multiple linear programming problems [4].

Geometric FBA solves this problem in a way that with each applied iteration, the allowable solution space is reduced by the algorithm. After a finite number of iterations, the bounds converge to a single solution, which is free of internal loops.

TIMING

The time to determine a geometricFBA solution depends on the size of the genome-scale model and is taking more ≥ 30 minutes for a model with more than 10,000 reactions.

```
modelgeo = model;
modelgeo = changeRxnBounds(modelgeo, 'EX_glc_D[e]',-20,'l');
modelgeo = changeRxnBounds (modelgeo, 'EX_o2[e]', 0, 'l');
modelgeo = changeObjective(modelgeo, 'DM_atp_c_');
FBAgeo = geometricFBA (modelgeo);
```

Parsimonious enzyme usage Flux Balance Analysis (pFBA)

The pFBA method was developed to achieve higher flux levels when more enzymes are needed [5]. The pFBA method first completes the FBA to find the optimal value for the objective function. Thereupon, satisfying a constraint that the original FBA objective function equals the optimal value, get the answer of an another linear program to discover the flux distribution that minimises the total flux through all metabolic reactions in the model.

Following example tests the basic solution for minimising the flux of all reactions, while growing on glucose or lactose minimal media.

```
% Loading models and expected results
load('testpFBAData.mat', 'model_glc', 'model_lac');
objGenes = load('testpFBAData.mat', 'GeneClasses_glc2', 'GeneClasses_glc1',...
    'GeneClasses_glc0', 'GeneClasses_lac2', 'GeneClasses_lac1', 'GeneClasses_lac0');
objRxns = load('testpFBAData.mat', 'RxnClasses_glc2', 'RxnClasses_glc1',...
    'RxnClasses_glc0', 'RxnClasses_lac2', 'RxnClasses_lac1', 'RxnClasses_lac0');
objModel = load('testpFBAData.mat', 'modelIrrev_glc2', 'modelIrrev_glc1',...
    'modelIrrev_glc0', 'modelIrrev_lac2', 'modelIrrev_lac1', 'modelIrrev_lac0');
```

TIMING

The time to determine a pFBA solution depends on the size of the genome-scale model and is taking from < 1 minute for a 1,000 reaction model, to 5 minutes for a model with more than 10,000 reactions.

- Minimise all flux for glucose

```
[t_objGenes.GeneClasses_glc0 t_objRxns.RxnClasses_glc0...
    t_objModel.modelIrrev_glc0] = pFBA(model_glc, 'geneoption', 0);
```

```
Single gene deletion analysis in progress ...
```

- Minimise all flux for lactate

```
[t_objGenes.GeneClasses_lac0 t_objRxns.RxnClasses_lac0...
    t_objModel.modelIrrev_lac0] = pFBA(model_lac, 'geneoption', 0);
```

```
Single gene deletion analysis in progress ...
```

```
t_objRxnsf = fieldnames(t_objRxns);
t_objModelf = fieldnames(t_objModel);
disp (t_objRxnsf)
```

```
    'RxnClasses_glc0'
    'RxnClasses_lac0'
```

```
disp (t_objModelf)
```

```
    'modelIrrev_glc0'
    'modelIrrev_lac0'
```

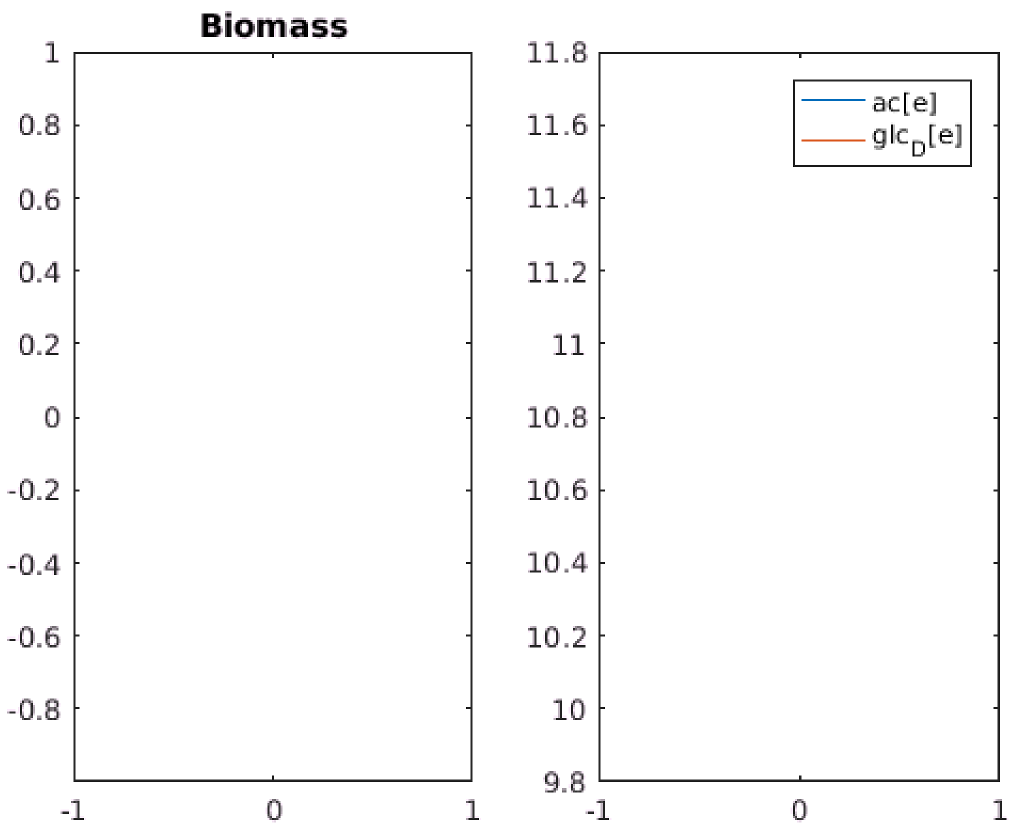
Dynamic FBA

The dynamic FBA is an extension of standard FBA that accounts for cell culture dynamics, implementing both dynamic (nonlinear programming) and static (LP) optimisation of an objective function and applying constraints to the rates of change of flux in addition to the standard FBA constraints [6].

The dynamic FBA method implemented in this function is essentially the same as the method described by Varma A. and B. O. Palsson [7].

```
modeldinamic = model;
modeldinamic = changeRxnBounds (modeldinamic, 'EX_glc_D[e]', -20, 'b');
modeldinamic = changeRxnBounds (modeldinamic, 'EX_o2[e]', -1000, 'l');
modeldinamic = changeRxnBounds (modeldinamic, 'EX_ac[e]', -1000, 'l');
smi = {'EX_glc_D[e]' 'EX_ac[e]'};
% exchange reaction for substrate in environment

smc = [10.8]; % Glucose, Acetate concentration (all in mM)

Xec = 0.001; % initial biomass
dt = 1.0/1000000000000.0; % time steps
time = 1.0/dt; % simulation time

[concentrationMatrix, excRxnNames, timeVec,...
    biomassVec] = dynamicFBA(modeldinamic, smi, smc, Xec, dt, time, smi );
```

```
ans = 1806
Step number Biomass
Dynamic FBA analysis in progress ...

No feasible solution - nutrients exhausted. Biomass:  0.001000
```



Relax FBA

Find the minimal set of relaxations on bounds and steady-state constraint to make the FBA problem feasible.

```
modelrelax = modelalter;
FBArel = relaxFBA(modelrelax)
```

```
csense is not defined. We assume that all constraints are equalities.
FBArel = 
    stat: 1
       v: [10600×1 double]
       r: [5835×1 double]
       p: [10600×1 double]
       q: [10600×1 double]
```

Flux enrichment analysis (FEA)

The flux enrichment analysis calculates the likelihood that a set of fluxes would belong to a subsystem or pathway.

TIMING

The time to calculate the FEA is < 1 second for any size of a model.

```
modelfea = model;
```

```
res = optimizeCbModel(modelfea,'max');
% say you are interested in enriching the active reactions
activeReactions = find(res.x)
```

```
activeReactions =

    17
    31
    34
    35
    36
    37
    39
    40
    41
    42
```

```
% You can also look for e.g. positive/negative/zeros flux reactions,
% that depends pretty much on the question.
% Now you look for the enrichement of reactions per subsystems
resultCell = FEA(modelfea, activeReactions, 'subSystems')
```

```
nonZeroInd =

    60
    97
    61
    70
    21
    38
    62
     9
    40
    13
```

```
resultCell =
    'P-value'         'Adjusted P-value'     'Group'                                                      'Enri
    [4.1074e-114]     [     3.6556e-112]     'Fatty acid oxidation'                                       [
    [ 6.7727e-89]     [      3.0138e-87]     'Peptide metabolism'                                         [
    [ 5.9624e-46]     [      1.4300e-44]     'Xenobiotics metabolism'                                     [
    [ 6.4271e-46]     [      1.4300e-44]     'Fatty acid synthesis'                                       [
    [ 5.1714e-40]     [      9.2051e-39]     'Cholesterol metabolism'                                     [
    [ 2.1540e-20]     [      3.1950e-19]     'Sphingolipid metabolism'                                    [
    [ 3.2852e-19]     [      4.1769e-18]     'Glycerophospholipid metabolism'                             [
    [ 7.1944e-15]     [      8.0037e-14]     'Exchange/demand reaction'                                   [
    [ 1.1620e-14]     [      1.1491e-13]     'Bile acid synthesis'                                        [
    [ 1.7554e-09]     [      1.5623e-08]     'Inositol phosphate metabolism'                              [
    [ 3.4088e-09]     [      2.7580e-08]     'Glycosphingolipid metabolism'                               [
    [ 4.3982e-08]     [      3.2620e-07]     'Blood group synthesis'                                      [
    [ 4.8635e-08]     [      3.3296e-07]     'Transport, golgi apparatus'                                 [
    [ 3.4854e-07]     [      2.1530e-06]     'Transport, mitochondrial'                                   [
    [ 3.6287e-07]     [      2.1530e-06]     'Nucleotide interconversion'                                 [
    [ 3.2185e-06]     [      1.7903e-05]     'Steroid metabolism'                                         [
    [ 4.4752e-05]     [      2.3429e-04]     'Folate metabolism'                                          [
    [ 1.2070e-04]     [      5.9681e-04]     'Arachidonic acid metabolism'                                [
    [ 1.2902e-04]     [      6.0434e-04]     'Chondroitin sulfate degradation'                            [
    [ 2.5743e-04]     [          0.0011]     'Phenylalanine metabolism'                                   [
    [ 2.9236e-04]     [          0.0012]     'Tryptophan metabolism'                                      [
    [ 3.1272e-04]     [          0.0012]     'O-glycan metabolism'                                        [
    [ 3.1272e-04]     [          0.0012]     'Ubiquinone synthesis'                                       [
    [ 7.6508e-04]     [          0.0028]     'NAD metabolism'                                             [
    [ 8.9717e-04]     [          0.0032]     'Starch and sucrose metabolism'                              [
    [     0.0013]     [          0.0044]     'Pyruvate metabolism'                                        [
    [     0.0013]     [          0.0044]     'Heme synthesis'                                             [
    [     0.0014]    [          0.0044]    'Transport, lysosomal'                                   [
    [     0.0021]    [          0.0064]    'Tyrosine metabolism'                                    [
    [     0.0027]    [          0.0077]    'CoA synthesis'                                          [
    [     0.0027]    [          0.0077]    'Glyoxylate and dicarboxylate metabolism'                [
    [     0.0040]    [          0.0110]    'Methionine and cysteine metabolism'                     [
    [     0.0047]    [          0.0126]    'Arginine and proline metabolism'                        [
    [     0.0054]    [          0.0140]    'Valine, leucine, and isoleucine metabolism'             [
    [     0.0076]    [          0.0193]    'Biotin metabolism'                                      [
    [     0.0127]    [          0.0312]    'Vitamin B6 metabolism'                                  [
    [     0.0130]    [          0.0312]    'Glycine, serine, alanine, and threonine metabolism'     [
    [     0.0143]    [          0.0328]    'Histidine metabolism'                                   [
    [     0.0144]    [          0.0328]    'Citric acid cycle'                                      [
    [     0.0181]    [          0.0403]    'Vitamin E metabolism'                                   [
    [     0.0209]    [          0.0454]    'Urea cycle'                                             [
    [     0.0222]    [          0.0471]    'Vitamin D metabolism'                                   [
    [     0.0233]    [          0.0482]    'Transport, endoplasmic reticular'                       [
    [     0.0265]    [          0.0526]    'Pyrimidine catabolism'                                  [
    [     0.0266]    [          0.0526]    'Pentose phosphate pathway'                              [
    [     0.0294]    [          0.0569]    'N-glycan synthesis'                                     [
    [     0.0321]    [          0.0608]    'Purine synthesis'                                       [
    [     0.0342]    [          0.0610]    'N-glycan degradation'                                   [
    [     0.0343]    [          0.0610]    'Triacylglycerol synthesis'                              [
    [     0.0343]    [          0.0610]    'Vitamin B2 metabolism'                                  [
    [     0.0360]    [          0.0628]    'Keratan sulfate degradation'                            [
    [     0.0399]    [          0.0683]    'Transport, peroxisomal'                                 [
    [     0.0553]    [          0.0929]    'Taurine and hypotaurine metabolism'                     [
    [     0.0581]    [          0.0958]    'Miscellaneous'                                          [
    [     0.0667]    [          0.1080]    'Alanine and aspartate metabolism'                       [
    [     0.0708]    [          0.1125]    'Aminosugar metabolism'                                  [
    [     0.0771]    [          0.1189]    'Beta-Alanine metabolism'                                [
    [     0.0787]    [          0.1189]    'Vitamin A metabolism'                                   [
    [     0.0788]    [          0.1189]    'Propanoate metabolism'                                  [
    [     0.0829]    [          0.1210]    'Phosphatidylinositol phosphate metabolism'              [
    [     0.0838]    [          0.1210]    'Transport, nuclear'                                     [
    [     0.0843]    [          0.1210]    'Eicosanoid metabolism'                                  [
    [     0.0924]    [          0.1305]    'Androgen and estrogen synthesis and metabolism'         [
    [     0.1150]    [          0.1599]    'Lysine metabolism'                                      [
    [     0.1238]    [          0.1695]    'Glycolysis/gluconeogenesis'                             [
    [     0.1369]    [          0.1794]    'N-glycan metabolism'                                    [
    [     0.1369]    [          0.1794]    'R group synthesis'                                      [
    [     0.1371]    [          0.1794]    'Purine catabolism'                                      [
    [     0.1582]    [          0.2040]    'C5-branched dibasic acid metabolism'                    [
    [     0.1652]    [          0.2090]    'Fructose and mannose metabolism'                        [
```

REFERENCES

[1] Orth, J. D., Thiele I., and Palsson, B. Ø. What is flux balance analysis? *Nat. Biotechnol., 28*(3), 245–248 (2010).

[2] Noronha A., et al. ReconMap: an interactive visualization of human metabolism. *Bioinformatics*., 33 (4): 605-607 (2017).

[3] Benyamini T, et al. Flux balance analysis accounting for metabolite dilution.*Genome Biology*., 11(4):R43 (2010).

[4] Smallbone K., and Simeonidis E. Flux balance analysis: A geometric perspective. *J Theor Biol*., 258: 311-315 (2009).

[5] Lewis N.E, et al. Omic data from evolved E. coli are consistent with computed optimal growth from genome-scale models. 10.1038 (2010).

[6] Mahadevan R., et al. Dynamic Flux Balance Analysis of Diauxic Growth in Escherichia coli. *Biophys J.,* 83(3):1331-40 (2002).

[7] Varma A. and Palsson, B. Ø. Stoichiometric flux balance models quantitatively predict growth and metabolic by-product secretion in wild-type Escherichia coli W3110. *App Environ Microbiol*., 60(10):3724-31 (1994).