Test physiologically relevant ATP yields from different carbon sources for a metabolic model

Author(s): Ines Thiele, Ronan M. T. Fleming, LCSB, University of Luxembourg.

Reviewer(s):

INTRODUCTION

In this tutorial, we show how to compute the ATP yield from different carbon sources under aerobic or anaerobic conditions. The theoretical values for the corresponding ATP yields are also provided. The tutorial can be adapted for Recon 3 derived condition- and cell-type specific models to test whether these models are still able to produce physiologically relevant ATP yields.

EQUIPMENT SETUP

If necessary, initialize 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: /opt/ibm/ILOG/CPLEX_Studio1271/cplex/matlab/x86-64_linux
  - [*---] 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                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 ...
> The COBRA Toolbox is up-to-date.
```

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

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

```
> Solver for LPproblems has been set to glpk.
> Solver for MILPproblems has been set to glpk.
> Solver glpk not supported for problems of type MIQP. Currently used: gurobi
> Solver glpk not supported for problems of type NLP. Currently used: matlab
> Solver glpk not supported for problems of type QP. Currently used: qpng
```

This tutorial can be run with glpk package as linear programming solver, which does not require additional installation and configuration. However, for the analysis of large models, such as Recon 3, it is not recommended to use glpk but rather industrial strength solvers, such as the GUROBI package. For detail information, refer to the solver installation guide.

PROCEDURE

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

```
% pathModel = '/Users/ines.thiele/Documents/MATLAB/files/';
% filename = '2017_04_28_Recon3d.mat';
pathModel = '~/work/sbgCloud/data/models/unpublished/Recon3D_models/';
filename = '2017_04_28_Recon3d.mat';
load([pathModel, filename])
model = modelRecon3model;
modelName = filename;
clear modelRecon3model Table_csources
tol = 1e-6;
```

In this tutorial, the used model is the generic model of human metabolism, the Recon 3[1].

The metabolites structures and reactions are from the Virtual Metabolic Human database (VMH, http://vmh.life).

Harmonization of abbreviation usage

First, we will harmonize different bracket types used in different model versions, e.g., different version of the human metabolic reconstruction.

```
model.rxns = regexprep(model.rxns, '\(', '\[');
model.rxns = regexprep(model.rxns, '\)', '\]');
model.mets = regexprep(model.mets, '\(', '\[');
model.mets = regexprep(model.mets, '\)', '\]');
```

Recon 3 uses ATPSm4mi instead of ATPS4m as an abbreviation for the ATP synthetase:

```
model.rxns = regexprep(model.rxns, 'ATPS4mi', 'ATPS4m');
```

Similarly, the glucose exchange reaction has been updated:

```
if length(strmatch('EX_glc[e]', model.rxns))>0
    model.rxns{find(ismember(model.rxns, 'EX_glc[e]'))} = 'EX_glc_D[e]';
end
```

Add ATP hydrolysis reaction to the model. If the reaction exist already, nothing will be added by the `rxnIDexists` variable will contain the index of the reaction that is present in the model. In this case, we will rename the reaction abbreviation to ensure that the tutorial works correctly.

```
[model, rxnIDexists] = addReaction(model, 'DM_atp_c_', 'h2o[c] + atp[c]  -> adp[c] + h[c] + pi
```

```
Warning: Reaction with the same name already exists in the model, updating the reaction
DM_atp_c_ h2o[c] + atp[c]  -> h[c] + adp[c] + pi[c]
```

```
if length(rxnIDexists)>0
    model.rxns{rxnIDexists} = 'DM_atp_c_'; % rename reaction in case that it exists already
end
```

Close model

Now, we will set the lower bound ('`model.lb`') of all exchange and sink (`siphon`) reactions to ensure that only those metabolites that are supposed to be taken up are indded supplied to the model.

First, we will find all reactions based on their abbreviation ('`model.rxns`')

```
modelClosed = model;
modelexchanges1 = strmatch('Ex_', modelClosed.rxns);
modelexchanges4 = strmatch('EX_', modelClosed.rxns);
modelexchanges2 = strmatch('DM_', modelClosed.rxns);
modelexchanges3 = strmatch('sink_', modelClosed.rxns);
```

Grab also the biomass reaction(s) based on the reaction abbreviation.

```
BM= (find(~cellfun(@isempty,strfind(lower(modelClosed.mets), 'bioma'))));
```

As these measures may not identify all exchange and sink reactions in a model, depending on the used nomencalture, we will also grab all reactions based on stoichiometry. Here, we will identify all reactions that contain only one non-zero entry in the S matrix (column).

```
selExc = (find(full((sum(abs(modelClosed.S)==1, 1)==1) & (sum(modelClosed.S~=0) == 1))))';
```

We will now put all these identified reactions together into one variable '`modelexchanges`' and set the lower bound for these reactions to 0.

```
modelexchanges = unique([modelexchanges1; modelexchanges2; modelexchanges3; modelexchanges4; s
modelClosed.lb(find(ismember(modelClosed.rxns, modelClosed.rxns(modelexchanges))))=0;
```

Also, set all upper bounds t 1000 (representing infinity). This may be important if other constraints had been applied to the model, which may interfere with the newly set lower bound of lb=0 for all exchange reactions. Note that this may affect any constraints that had been applied, e.g., condition-specific constraints based on measured uptake or secretion rates.

```
modelClosed.ub(selExc) = 1000;
```

Define the ATP hydrolysis reactioDefinen DM_atp_c to be the objective reaction, for which we will maximize for in the following sections.

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

Store the original closed model setup for consequent use in the variable `modelClosedOri`.

```
modelClosedOri = modelClosed;
```

Test for ATP yield from different carbon sources

Now, we re ready to thest for the different individual carbon sources under aerobic and anaerobic conditions for their ATP yield. Therefore, we will provide 1 mol/gdw/hr of a carbon source and maximize the flux through the DM_atp_c_.

The results will be stored in the table 'Table_csources'. The table will also contain the theoretical ATP yield, as given by[2]. The table also provides the information of how much flux is going throught he ATP syntheatse. Note that the computed flux distribution is not garantied to be unique, although we use the option 'zero', which approximates the sparsest possible flux distribution with an maximal ATP yield.

Carbon source: Glucose (VMH ID: glc_D), Oxygen: Yes

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = -1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_glc_D[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_glc_D[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
% build table
Table_csources{2, 1} = strcat(modelName, ': ATP yield');
```

```
Table_csources{3, 1} = strcat(modelName, ': ATPS4m yield');
Table_csources{4, 1} = 'Theoretical';
% fill in results
k = 2;
Table_csources{1, k} = 'glc - aerobic';

Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x) > 0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = '31';
```

For this carbon source (glucose), we will also print all reactions that are non-zero in the sparse flux distribution and thus contribute to the maximale ATP yield.

```
ReactionsInSparseSolution = modelClosed.rxns(find(FBA.x))
```

```
ReactionsInSparseSolution = 
    '10FTHFtm'
    '2HBt2'
    '2HCO3_NAt'
    '3HKYNAKGAT'
    '4HGLSDm'
    '4HOXPACDOX_NADP_'
    '4MOPt2im'
    '5MTHFt'
    '5MTHFt2'
    'ABUTt4_2_r'
    'ACCOAtn'
    'ACCOAtr'
    'ACHtn'
    'ACONTm'
    'ACRNtm'
    'ACt2r'
    'ADK1m'
    'ADK3'
    'ADK3m'
    'ADKd'
    'ADNt'
    'ADPtx'
    'ADRNCRNt'
    'ADRNt'
    'AKGDm'
    'AKGMALtm'
    'AKGt4_3'
    'AKGtp'
    'ALAASNNaEx'
    'ALADGLYexR'
    'ALAGLNexR'
    'ALAGLYexR'
    'ALATA_L'
    'ALAtN1'
    'ALCD21_D'
    'ALCD21_L'
    'ALCD22_D'
    'ALR2'
    'ALR3'
    'AMETt2m'
    'AMPtp'
    'ARACHCRNt'
```

```
'ARACHDCOAtx'
'ARACHDtr'
'ARACHt'
'ARGt4'
'ASCBt'
'ASCBt4'
'ASNtN1'
'ATPtm'
'BHBt'
'BILDGLCURt'
'BILGLCURt'
'BILIRUBt2'
'BILIRUBtr'
'BTNt2'
'BTNt4i'
'C160CPT1'
'C160CRNt'
'C161CRN2t'
'C161CRNt'
'C180CRNt'
'C181CRNt'
'C204CPT1'
'C204CRNt'
'C226CRNt'
'CAt7r'
'CHAT'
'CHATn'
'CHOLATEt'
'CHOLt4'
'CHOLtn'
'CHOLtu'
'CHSTEROLt2'
'CHSTEROLt3'
'CITRtm'
'CITt4_2'
'CITtam'
'CLOXAtex2'
'CLPNDCRNt'
'CO2tm'
'COAtm'
'COAtn'
'COAtp'
'CREATt4_2_r'
'CRNtim'
'CRNtuNa'
'CRVNCtr'
'CSm'
'CSNAT2x'
'CSNAT3x'
'CSNATer'
'CSNATr'
'CYSGLTH'
'CYSGLUexR'
'CYSGLYexR'
'CYStec'
'CYTDt'
'CYTK10'
'CYTK10n'
'CYTK11'
'CYTK11n'
'CYTK12'
'CYTK12n'
'CYTK13n'
'CYTK14'
'CYTK14n'
'CYTK1n'
'CYTK2n'
'CYTK3'
```

```
'CYTK4'
'CYTK4n'
'CYTK5'
'CYTK5n'
'CYTK6'
'CYTK6n'
'CYTK7'
'CYTK7n'
'CYTK8'
'CYTK9'
'CYTK9n'
'D3AIBTm'
'DALAt2r'
'DCK1m'
'DCK1n'
'DCK2n'
'DCSPTN1CRNt'
'DCYTt'
'DGCHOLte'
'DHEASt'
'DHFtm'
'DIDPtn'
'DITPtn'
'DLNLCGCPT1'
'DLNLCGCRNt'
'DMHPTCRNt'
'DMNONCOACRNCPT1'
'DMNONCRNt'
'DOPAt4_2_r'
'DOPAtu'
'DUDPtn'
'DURIK1'
'EICOSTETCRNt'
'ELAIDCRNt'
'ESTRADIOLtr'
'ESTRONESt'
'ESTRONEtr'
'EX_hco3[e]'
'EX_o2s[e]'
'F1Atg'
'FALDH'
'FATP1t'
'FATP2t'
'FATP3t'
'FATP4t'
'FATP5t'
'FATP6t'
'FATP7t'
'FATP8t'
'FATP9t'
'FOLt2'
'FRDPtc'
'FRDPtr'
'FRUt1r'
'FUMm'
'FUMSO3tm'
'FUMSO4tm'
'FUMtm'
'FUMTSULtm'
'G6Pter'
'GALGLUSIDEtl'
'GALt1r'
'GCC2cm'
'GCHOLAt'
'GCHOLAte'
'GGT6'
'GHMT2rm'
'GLACter'
```

```
'GLCt1r'
'GLCt2_2'
'GLCURter'
'GLNASNNaEx'
'GLNLASEer'
'GLNTHRNaEx'
'GLNtN1'
'GLRASE'
'GLUt2m'
'GLYBt4_2_r'
'GLYC3Ptm'
'GLYt7_211_r'
'GLYtm'
'GSNt'
'GUAD'
'GULLACter'
'GULNter'
'H2O2tly'
'H2Otly'
'H2Otm'
'H2Otp'
'HCO3_CLt'
'HDCAtr'
'HDCEAtr'
'HISt4'
'HISTAtu'
'HIStN1'
'HPDCACRNt'
'HPYRRy'
'HPYRtp'
'HSD11B1r'
'HSD11B2r'
'HSD17B1'
'HSD17B7r'
'HSD3A1r'
'HSD3A2r'
'HSD3B7P'
'Htx'
'ICDHxm'
'ICDHyrm'
'ILEtec'
'INSt'
'INSTt4'
'It'
'KCC2t'
'KCCt'
'KHK'
'KYNAKGAT'
'KYNATESYN'
'LALDO'
'LDH_Lm'
'LEUKTRA4t'
'LEUKTRB4t'
'LEUKTRC4t'
'LEUKTRD4t'
'LEUKTRE4t'
'LEUKTRF4t'
'LEUt5m'
'LEUTAm'
'LEUtec'
'LGNCt'
'L_LACtcm'
'LNELDCCRNt'
'LNLCCRNt'
'LNLNCACPT1'
'LNLNCACPT2'
'LNLNCACRNt'
'LNLNCGCPT1'
```

```
'LNLNCGCRNt'
'LTC4CP'
'LTD4DP'
'MALSO3tm'
'MALSO4tm'
'MALtm'
'MALTSULtm'
'MANt1r'
'MDHm'
'METtec'
'MMEm'
'MMTSADm'
'MTHFCm'
'MTHFDm'
'NACUP'
'NADHtpu'
'NADHtru'
'NADtru'
'NAHCO3_HCLt'
'NAIt'
'NAt'
'NAt5'
'NAtx'
'NCKt'
'NCNt'
'NDPK10'
'NDPK10n'
'NDPK1m'
'NDPK1n'
'NDPK3n'
'NDPK4n'
'NDPK5n'
'NDPK6n'
'NDPK7n'
'NDPK8n'
'NH4t3r'
'NKCC2t'
'NKCCt'
'NMNATr'
'NRPPHRt4_2_r'
'O2St'
'O2Stm'
'O2tm'
'O2tp'
'OCDCAtr'
'OCDCEAtr'
'ORNt3m'
'ORNt4m'
'OXAHCOtex'
'PAIL45P_HStn'
'PAIL4P_HStn'
'PCRNtc'
'PCRNtm'
'PE_HSter'
'PGCD'
'PGDIr'
'PGESr'
'PGISr'
'PHCHGSm'
'PHEt4'
'PI45P5P'
'PIter'
'PItn'
'PPAt'
'PPItr'
'PRGNLONEtr'
'PRODt2r'
'PROSTGD2t'
```

```
'PROSTGE2t'
'PROSTGE2t2'
'PROSTGF2t'
'PROSTGH2t'
'PROSTGI2t'
'PROSTGI2tr'
'PTDCACRNt'
'PVD3'
'PYRt2p'
'RDH1'
'RDH1a'
'RDH2'
'RDH2a'
'RDH3'
'RDH3a'
'RETNt'
'SBTD_D2'
'SCP22x'
'SERDGLYexR'
'SERGLNexR'
'SERGLYexR'
'SERTHRNaEx'
'SERtN1'
'SERtp'
'SO4CLtex2'
'SO4HCOtex'
'SO4t4_2'
'SPTix'
'SRTNt6_2_r'
'SRTNtu'
'STRDNCCRNt'
'SUCCt2m'
'SUCOAS1m'
'SUCOASm'
'TAURt4_2_r'
'TAURtcx'
'TCHOLAt'
'TCHOLAte'
'TDCHOLAte'
'TDCHOLAtx'
'TETPENT3CPT1'
'TETPENT3CPT2'
'TETPENT3CRNt'
'TETPENT6CPT1'
'TETPENT6CPT2'
'TETPENT6CRNt'
'TETTET6CPT1'
'TETTET6CPT2'
'TETTET6CRNt'
'THCHOLSTOICtm'
'THMMPt4'
'THRGLNexR'
'THRGLYexR'
'THYMDt1'
'THYOXt2'
'TMNDNCCRNt'
'TRIODTHYt'
'TRIODTHYt2'
'TRIOK'
'TSULt4_3'
'TTDCPT2'
'TTDCRNt'
'TXA2tr'
'TYRt'
'UDPGLCAter'
'UDPGLCter'
'UMPK2'
'UMPK3'
```

```
'UMPK3n'
'UMPK4'
'UMPK4n'
'UMPK5'
'UMPK5n'
'UMPK6'
'UMPK6n'
'UMPK7'
'UMPK7n'
'UMPKn'
'URATEt'
'UREAt5'
'URIt'
'UROLACer'
'UTPtn'
'VALtec'
'VD3'
'XOLEST2te'
'XYLUR'
'r0002'
'r0021'
'r0027'
'r0081'
'r0083'
'r0202'
'r0226'
'r0249'
'r0280'
'r0317'
'r0377'
'r0383'
'r0408'
'r0409'
'r0425'
'r0426'
'r0434'
'r0443'
'r0483'
'r0509'
'r0527'
'r0552'
'r0553'
'r0555'
'r0571'
'r0584'
'r0615'
'r0617'
'r0641'
'r0643'
'r0647'
'r0686'
'r0688'
'r0698'
'r0754'
'r0755'
'r0756'
'r0757'
'r0784'
'r0801'
'r0809'
'r0812'
'r0817'
'r0819'
'r0821'
'r0822'
'r0829'
'r0830'
'r0834'
```

```
'r0835'
'r0836'
'r0840'
'r0841'
'r0842'
'r0860'
'r0879'
'r0881'
'r0885'
'r0913'
'r0915'
'r0917'
'r0931'
'r0932'
'r0936'
'r0937'
'r0942'
'r0960'
'r0983'
'r0984'
'r0995'
'r0998'
'r1001'
'r1008'
'r1019'
'r1051'
'r1052'
'r1088'
'r1090'
'r1147'
'r1148'
'r1156'
'r1291'
'r1292'
'r1377'
'r1378'
'r1384'
'r1400'
'r1401'
'r1418'
'r1428'
'r1429'
'r1435'
'r1493'
'r1498'
'r1512'
'r1544'
'r1546'
'r1547'
'r1548'
'r1549'
'r1551'
'r1552'
'r1553'
'r1554'
'r1556'
'r1557'
'r1559'
'r1560'
'r1561'
'r1562'
'r1563'
'r1564'
'r1565'
'r1566'
'r1567'
'r1568'
'r1569'
```

```
'r1570'
'r1571'
'r1573'
'r1574'
'r1576'
'r1578'
'r1579'
'r1580'
'r1581'
'r1583'
'r1584'
'r1585'
'r1586'
'r1587'
'r1588'
'r1589'
'r1590'
'r1591'
'r1592'
'r1593'
'r1594'
'r1595'
'r1596'
'r1597'
'r1598'
'r1599'
'r1600'
'r1602'
'r1603'
'r1604'
'r1605'
'r1606'
'r1607'
'r1608'
'r1609'
'r1610'
'r1611'
'r1612'
'r1613'
'r1614'
'r1615'
'r1616'
'r1617'
'r1618'
'r1619'
'r1620'
'r1621'
'r1622'
'r1623'
'r1624'
'r1625'
'r1626'
'r1627'
'r1628'
'r1629'
'r1630'
'r1631'
'r1632'
'r1633'
'r1634'
'r1635'
'r1636'
'r1637'
'r1638'
'r1640'
'r1641'
'r1642'
'r1643'
```

```
'r1644'
'r1645'
'r1646'
'r1647'
'r1648'
'r1649'
'r1650'
'r1651'
'r1652'
'r1653'
'r1654'
'r1655'
'r1656'
'r1657'
'r1658'
'r1659'
'r1660'
'r1661'
'r1662'
'r1664'
'r1665'
'r1666'
'r1668'
'r1669'
'r1670'
'r1671'
'r1680'
'r1682'
'r1685'
'r1686'
'r1690'
'r1695'
'r1698'
'r1700'
'r1701'
'r1715'
'r1716'
'r1723'
'r1725'
'r1726'
'r1728'
'r1730'
'r1731'
'r1738'
'r1742'
'r1743'
'r1744'
'r1745'
'r1746'
'r1753'
'r1758'
'r1759'
'r1760'
'r1772'
'r1775'
'r1787'
'r1789'
'r1791'
'r1806'
'r1811'
'r1926'
'r1931'
'r1936'
'r1946'
'r1954'
'r1992'
'r2006'
'r2007'
```

```
'r2014'
'r2079'
'r2081'
'r2082'
'r2083'
'r2084'
'r2085'
'r2086'
'r2087'
'r2088'
'r2089'
'r2091'
'r2092'
'r2093'
'r2094'
'r2095'
'r2096'
'r2097'
'r2099'
'r2101'
'r2102'
'r2103'
'r2104'
'r2105'
'r2106'
'r2107'
'r2108'
'r2109'
'r2110'
'r2111'
'r2112'
'r2113'
'r2115'
'r2116'
'r2117'
'r2118'
'r2119'
'r2120'
'r2121'
'r2122'
'r2123'
'r2124'
'r2125'
'r2126'
'r2127'
'r2128'
'r2129'
'r2130'
'r2131'
'r2132'
'r2133'
'r2136'
'r2139'
'r2142'
'r2143'
'r2194'
'r2196'
'r2198'
'r2199'
'r2200'
'r2201'
'r2202'
'r2308'
'r2309'
'r2310'
'r2311'
'r2312'
'r2313'
```

```
    'r2314'
    'r2315'
    'r2316'
    'r2317'
    'r2318'
    'r2319'
    'r2320'
    'r2321'
    'r2322'
    'r2323'
    'r2324'
    'r2325'
    'r2327'
    'r2328'
    'r2329'
    'r2330'
    'r2331'
    'r2332'
    'r2333'
    'r2338'
    'r2342'
    'r2343'
    'r2346'
    'r2347'
    'r2352'
    'r2354'
    'r2355'
    'r2356'
    'r2357'
    'r2358'
    'r2359'
    'r2360'
    'r2361'
    'r2362'
    'r2363'
    'r2365'
    'r2366'
    'r2367'
    'r2368'
    'r2369'
    'r2371'
    'r2372'
    'r2373'
    'r2374'
    'r2375'
    'r2376'
```

We now initiate the next test and delete the variable 'FBA'.

```
k = k+1; clear FBA
```

Carbon source: Glucose (VMH ID: glc_D), Oxygen: No

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns,'EX_o2[e]'))) = 0;
modelClosed.ub(find(ismember(modelClosed.rxns,'EX_o2[e]'))) = 0;
modelClosed.lb(find(ismember(modelClosed.rxns,'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns,'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns,'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns,'EX_glc_D[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns,'EX_glc_D[e]'))) = -1;
```

```
FBA = optimizeCbModel(modelClosed,'max');
% fill in results
Table_csources{1, k} = 'glc - anaerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x)>=0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns,'ATPS4m'))));
end
Table_csources{4, k} = '2';
k = k+1; clear FBA
```

Carbon source: Glutamine (VMH ID: gln_L), Oxygen: Yes

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = -1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_gln_L[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_gln_L[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
% fill in results
Table_csources{1, k} = 'gln_L - aerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x) >= 0
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = 'NA';
k = k+1; clear FBA
```

Carbon source: Glutamine (VMH ID: gln_L), Oxygen: No

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_gln_L[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_gln_L[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
% fill in results
Table_csources{1, k} = 'gln_L - anaerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x) >= 0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = 'NA';
k = k+1; clear FBA
```

Carbon source: Fructose (VMH ID: fru), Oxygen: Yes

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = -1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_fru[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_fru[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
% fill in results
Table_csources{1, k} = 'fru - aerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x)>=0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3 ,k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = '31';
k = k+1; clear FBA
```

Carbon source: Fructose (VMH ID: fru), Oxygen: No

```
modelClosed = changeObjective(modelClosed, 'DM_atp_c_');
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_fru[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_fru[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
% fill in results
Table_csources{1, k} = 'fru - anaerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x)>=0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = '2';
k = k+1; clear FBA
```

Carbon source: Butyrate (VMH ID: but), Oxygen: Yes

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = -1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_but[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_but[e]'))) = -1;
```

```
FBA = optimizeCbModel(modelClosed, 'max');
% fill in results
Table_csources{1, k} = 'but - aerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x) >= 0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = '21.5';
k = k+1; clear FBA
```

Carbon source: Butyrate (VMH ID: but), Oxygen: No

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_but[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_but[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
Table_csources{1, k} = 'but - anaerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x) > 0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = '0';
k = k+1; clear FBA
```

Carbon source: Caproic acid (VMH ID: caproic), Oxygen: Yes

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = -1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_caproic[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_caproic[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
Table_csources{1, k} = 'caproic - aerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x) > 0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = '35.25';
k = k+1; clear FBA
```

Carbon source: Caproic acid (VMH ID: caproic), Oxygen: No

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_caproic[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_caproic[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
Table_csources{1, k} = 'caproic - anaerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x) > 0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = '0';
k = k+1; clear FBA
```

Carbon source: Octanoate (VMH ID: octa), Oxygen: Yes

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = -1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_octa[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_octa[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
Table_csources{1, k} = 'octa - aerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x) > 0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = '49';
k = k+1; clear FBA
```

Carbon source: Octanoate (VMH ID: octa), Oxygen: No

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, modelClosed.rxns(modelexchanges)))) = 0;
modelClosed.c = zeros(length(modelClosed.rxns), 1);
modelClosed = changeObjective(modelClosed, 'DM_atp_c_');
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_octa[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_octa[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
Table_csources{1, k} = 'octa - anaerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x) > 0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = '0';
k = k+1; clear FBA
```

Carbon source: Decanoate (VMH ID: dca), Oxygen: Yes

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, modelClosed.rxns(modelexchanges))))=0;
modelClosed.c = zeros(length(modelClosed.rxns),1);
modelClosed = changeObjective(modelClosed, 'DM_atp_c_');
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = -1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_dca[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_dca[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
Table_csources{1, k} = 'dca - aerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x) > 0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = '62.75';
k = k+1; clear FBA
```

Carbon source: Decanoate (VMH ID: dca), Oxygen: No

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_dca[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_dca[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');

Table_csources{1, k} = 'dca - anaerobic';
% fill in only when the LP problem was feasible
Table_csources(2, k) = num2cell(FBA.f);
if length(FBA.x) > 0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = '0';
k = k+1; clear FBA
```

Carbon source: Laureate (VMH ID: ddca), Oxygen: Yes

```
modelClosed = modelClosedOri;
modelClosed.c = zeros(length(modelClosed.rxns), 1);
modelClosed = changeObjective(modelClosed, 'DM_atp_c_');
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = -1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_ddca[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_ddca[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
Table_csources{1, k} = 'ddca - aerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x) > 0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = '76.5';
k = k+1; clear FBA
```

Carbon source: Laureate (VMH ID: ddca), Oxygen: No

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_ddca[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_ddca[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
Table_csources{1, k} = 'ddca - anaerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x)>0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3,k) = num2cell(FBA.x(find(ismember(modelClosed.rxns,'ATPS4m'))));
end
Table_csources{4,k} = '0';
k = k+1; clear FBA
```

Carbon source: Tetradecanoate (VMH ID: ttdca), Oxygen: Yes

```
modelClosed = modelClosedOri;
```

```
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = -1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_ttdca[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_ttdca[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
Table_csources{1, k} = 'ttdca - aerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x) > 0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = '90.25';
k = k+1; clear FBA
```

Carbon source: Tetradecanoate (VMH ID: ttdca), Oxygen: No

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_ttdca[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_ttdca[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
Table_csources{1, k} = 'ttdca - anaerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x)>0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns,'ATPS4m'))));
end
Table_csources{4, k} = '0';
k = k+1; clear FBA
```

Carbon source: Hexadecanoate (VMH ID: hdca), Oxygen: Yes

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = -1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_hdca[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_hdca[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
Table_csources{1, k} = 'hdca - aerobic';
Table_csources(2, k) = num2cell(FBA.f);
if length(FBA.x) > 0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = '104';
k = k+1; clear FBA
```

Carbon source: Hexadecanoate (VMH ID: hdca), Oxygen: No

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_hdca[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_hdca[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
Table_csources{1, k} = 'hdca - anaerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x) > 0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = '0';
k = k+1; clear FBA
```

Carbon source: Octadecanoate (VMH ID: ocdca), Oxygen: Yes

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = -1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_ocdca[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_ocdca[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
Table_csources{1, k} = 'ocdca - aerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x) > 0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = '117.75';
k = k+1; clear FBA
```

Carbon source: Octadecanoate (VMH ID: ocdca), Oxygen: No

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_ocdca[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_ocdca[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
Table_csources{1, k} = 'ocdca - anaerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x) > 0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = '0';
k = k+1; clear FBA
```

Carbon source: Arachidate (VMH ID: arach), Oxygen: Yes

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = -1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_arach[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_arach[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
Table_csources{1, k} = 'arach - aerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x) > 0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = '131.5';
k = k+1; clear FBA
```

Carbon source: Arachidate (VMH ID: arach), Oxygen: No

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_arach[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_arach[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
Table_csources{1, k} = 'arach - anaerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x) > 0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
```

```
Table_csources{4, k} = '0';
k = k+1; clear FBA
```

Carbon source: Behenic acid (VMH ID: docosac), Oxygen: Yes

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = -1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_docosac[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_docosac[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
Table_csources{1, k} = 'docosac - aerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x) > 0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = '145.25';
k = k+1; clear FBA
```

Carbon source: Behenic acid (VMH ID: docosac), Oxygen: No

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_docosac[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_docosac[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
Table_csources{1, k} = 'docosac - anaerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x) > 0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = '0';
k = k+1; clear FBA
```

Carbon source: Lignocerate (VMH ID: lgnc), Oxygen: Yes

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = -1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_lgnc[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_lgnc[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
Table_csources{1, k} = 'lgnc - aerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x) > 0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3 ,k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = '159';
k = k+1; clear FBA
```

Carbon source: Lignocerate (VMH ID: lgnc), Oxygen: No

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_lgnc[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_lgnc[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
Table_csources{1, k} = 'lgnc - anaerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x) > 0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = '0';
k = k+1; clear FBA
```

Carbon source: Cerotate (VMH ID: hexc), Oxygen: Yes

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = -1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_hexc[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_hexc[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
Table_csources{1, k} = 'hexc - aerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x) > 0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3, k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = '172.75';
k = k+1; clear FBA
```

Carbon source: Cerotate (VMH ID: hexc), Oxygen: No

```
modelClosed = modelClosedOri;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_o2[e]'))) = 0;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = -1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_h2o[e]'))) = 1000;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_co2[e]'))) = 1000;
modelClosed.lb(find(ismember(modelClosed.rxns, 'EX_hexc[e]'))) = -1;
modelClosed.ub(find(ismember(modelClosed.rxns, 'EX_hexc[e]'))) = -1;

FBA = optimizeCbModel(modelClosed, 'max');
Table_csources{1, k} = 'hexc - anaerobic';
Table_csources(2, k) = num2cell(FBA.f);
% fill in only when the LP problem was feasible
if length(FBA.x) > 0
    % set all flux values less than tol to 0
    FBA.x(find(abs(FBA.x)<=tol)) = 0;
    Table_csources(3 ,k) = num2cell(FBA.x(find(ismember(modelClosed.rxns, 'ATPS4m'))));
end
Table_csources{4, k} = '0';
k = k+1; clear FBA
```

The table contains all computed ATP yields.

```
Table_csources = Table_csources'
```

```
Table_csources =
                        []    '2017_04_28_Recon3d.mat: ATP yield'    '2017_04_28_Recon3d.mat: ATPS4m yiel
    'glc - aerobic'         [                          32.0000]    [                             28.000
    'glc - anaerobic'       [                           2.0000]    [
    'gln_L - aerobic'       [                          22.5000]    [                             20.500
    'gln_L - anaerobic'     [                           1.0000]    [
    'fru - aerobic'         [                          32.0000]    [                             28.000
    'fru - anaerobic'       [                           2.0000]    [
    'but - aerobic'         [                          22.0000]    [                             22.000
    'but - anaerobic'       [                                0]
    'caproic - aerobic'     [                          34.5000]    [                             33.500
    'caproic - anaerobic'   [                                0]
    'octa - aerobic'        [                          52.0000]    [                             50.000
    'octa - anaerobic'      [                                0]
    'dca - aerobic'         [                          67.0000]    [                             64.000
    'dca - anaerobic'       [                                0]
    'ddca - aerobic'        [                          82.0000]    [                             78.000
    'ddca - anaerobic'      [                                0]
    'ttdca - aerobic'       [                          97.0000]    [                             92.000
    'ttdca - anaerobic'     [                                0]
    'hdca - aerobic'        [                         113.0000]    [                            106.000
    'hdca - anaerobic'      [                      -4.3763e-12]    [
    'ocdca - aerobic'       [                         129.0000]    [                            120.000
    'ocdca - anaerobic'     [                      -1.8350e-11]    [
    'arach - aerobic'       [                         143.0000]    [                            133.000
    'arach - anaerobic'     [                                0]
    'docosac - aerobic'     [                         157.0000]    [                            146.000
    'docosac - anaerobic'   [                                0]
    'lgnc - aerobic'        [                         168.0000]    [                            156.000
    'lgnc - anaerobic'      [                                0]
    'hexc - aerobic'        [                         178.5000]    [                            167.500
    'hexc - anaerobic'      [                                0]
```

TIMING

This tutorial takes only a few minutes.

REFERENCES

[1] Brunk, E. et al. Recon 3D: A Three-Dimensional View of Human Metabolism and Disease. Submited