Sparse Flux Balance Analysis

Author: Ronan Fleming, Hoai Minh Le, Systems Biochemistry Group, University of Luxembourg.

Reviewer:

INTRODUCTION

We consider a biochemical network of m molecular species and n biochemical reactions. The biochemical network is mathematically represented by a stoichiometric matrix $S \in \mathcal{Z}^{m \times n}$. In standard notation, flux balance analysis (FBA) is the linear optimisation problem

$$\begin{aligned} \min_{v} \quad & \rho(v) \equiv c^T v \\ \text{s.t.} \quad & Sv = b, \\ & l \leq v \leq u, \end{aligned}$$

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. A typical application of flux balance analysis is to predict an optimal non-equilibrium steady-state flux vector that optimises a linear objective function, such biomass production rate, subject to bounds on certain reaction rates. Herein we use sparse flux balance analysis to predict a minimal number of active reactions [melendez-hevia_game_1985], consistent with an optimal objective derived from the result of a standard flux balance analysis problem. In this context *sparse flux balance analysis* requires a solution to the following problem

$$\begin{aligned} \min_{v} \quad & \|v\|_0 \\ \text{s.t.} \quad & Sv = b \\ & l \leq v \leq u \\ & c^T v = \rho^\star \end{aligned}$$

where the last constraint is represents the requirement to satisfy an optimal objective value $\rho^\star$ derived from any solution to a flux balance analysis (FBA) problem.

EQUIPMENT SETUP

If necessary, initialise the cobra toolbox

```
global TUTORIAL_INIT_CB;
if ~isempty(TUTORIAL_INIT_CB) && TUTORIAL_INIT_CB==1
    initCobraToolbox
    changeCobraSolver('gurobi','all');
end
```

PROCEDURE

Set the tolerance to distinguish between zero and non-zero flux, based on the numerical tolerance of the currently installed optimisation solver.

```
feasTol = getCobraSolverParams('LP', 'feasTol');
```

Load Recon3.0model, unless it is already loaded into the workspace.

```
clear model
if ~exist('modelOrig','var')
    filename='Recon3.0model';
    directory='~/work/sbgCloud/programReconstruction/projects/recon2models/data/reconXComparis
    model = loadIdentifiedModel(filename,directory);
    model.csense(1:size(model.S,1),1)='E';
    modelOrig = model;
else
    model=modelOrig;
end
```

Display the constraints

```
minInf=-1000;
maxInf=1000;
printConstraints(model, minInf, maxInf);
```

```
MinConstraints:
maxConstraints:
```

Select the biomass reaction to optimise

```
model.biomassBool=strcmp(model.rxns,'biomass_reaction');
model.c(model.biomassBool)=1;
```

Display the biomass reaction

```
rxnAbbrList={'biomass_reaction'};
printFlag = 1;
formulas = printRxnFormula(model, rxnAbbrList, printFlag);
```

```
biomass_reaction 20.6508 h2o[c] + 20.7045 atp[c] + 0.385872 glu_L[c] + 0.352607 asp_L[c] + 0.036117 gtp[
```

Sparse flux balance analysis

We provide two options to run sparse flux balance analysis. A: directly in one step, no quality control, and B: two steps, all approximations, with a heuristic sparsity test**.**

TIMING

The time to compute a sparse flux balance analysis solution depends on the size of the genome-scale model and the option chosen to run sparse flux balance analysis. Option A: directly in one step, no quality control, can take anything from <0.1 seconds for a 1,000 reaction model, to 1,000 seconds for a model with 20,000 reactions. Option B: two steps, all approximations, with a sparsity test could take

hours for a model with >10,000 reactions because the length of time for the heuristic sparsity test is proportional to the number of active reactions in an approximate sparse solution.

A. Sparse flux balance analysis (directly in one step, no quality control)

This approach computes a sparse flux balance analysis solution, satisfing the FBA objection, with the default approach to approximate the solution to the cardinality minimisation problem underling sparse FBA. This approach does not check the quality of the solution, i.e., whether indeed it is the sparsest flux vector satisfing the optimality criterion $c^T v = \rho^\star$.

First choose whether to maximize ('max') or minimize ('min') the FBA objective. Here we choose maximise

```
osenseStr='max';
```

Choose to minimize the zero norm of the optimal flux vector

```
minNorm='zero';
```

Run sparse flux balance analysis

```
sparseFBAsolution = optimizeCbModel(model, osenseStr, minNorm);
```

Obtain the vector of reaction rates from the solution structure

```
v = sparseFBAsolution.v;
```

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

```
nonZeroFlag = 1;
printFluxVector(model, v, nonZeroFlag);
```

```
4MOPt2im 774.74
5AOPtm 1000
ACETONEt2 436.469
ACONTm -166.447
ACt2m -884.726
ACTLMO 957.68
ACTNMO 436.469
ADEt -610.027
ADK3 -637.07
ADNt -27.0433
AKGMALtm 173.401
AKGt4_3 296.958
ALATA_L -386.265
ALCD21_D 237.152
ALR2 521.211
AMPDA 1000
AMY1e 1.93991
ARTFR208 111.111
ASAH1 978.661
ASCBOX 200.883
ASPCTr -265.632
ATPtm 1000
CATm 111.111
CBPPer 1000
CBPter 1000
CDIPTr -8.7824
CITtbm 1000
```

CLS_hs 8.7824
CO2ter -1000
CO2tm 166.447
CYTK10 349.71
CYTK11 1000
CYTK14 69.835
CYTK2n -992.887
CYTK4 609.862
CYTK9 -1000
DADNt4 -312.599
DAGt -766.729
DASCBR 200.883
DATPtn 9.93123
DCK1m 329.987
DCK1n -14.226
DCK2n -978.661
DCTPtn 1000
DCYTDn 492.887
DCYTt -170.013
DGTPtn 7.45652
DIDPtn 1000
DITPtn -1000
D_LACt2 -673.621
D_LACtm -673.621
DNDPt14m 9.86193
DNDPt19m 339.848
DOPAt4_2_r 918.523
DRPA 1000
DTDPtn 1000
DTTPtn -990.138
DUDPtn -1000
DURIt -492.887
DURItn -492.887
EX_acetone[e] -436.469
EX_atp[e] -1000
EX_chsterol[e] -154.258
EX_cmp[e] -9.69955
EX_dag_hs[e] 766.729
EX_dopa[e] -918.523
EX_gsn[e] 1000
EX_h2o2[e] -1000
EX_hco3[e] 522.582
EX_imp[e] 965.335
EX_inost[e] -771.121
EX_lac_D[e] 673.621
EX_o2s[e] -1000
EX_ocdca[e] -138.889
EX_pe_hs[e] -41.7153
EX_pglyc_hs[e] -10.9776
EX_ps_hs[e] -913.83
EX_Rtotal[e] 978.661
EX_strch1[e] -1.93991
EX_thmtp[e] 1000
EX_urate[e] 1000
EX_utp[e] -1000
EX_xolest_hs[e] 138.889
FALDtly 478.661
FATP3t -138.889
FOLt2 -1000
FUMm 941.873
FUMtm -58.1272
G6Pter -1000
GALSIDEtl 978.661
GGNG 1.93991
GLBRAN 1.93991
GLCter 1000
GLCtly 978.661
GLGNS1 1.93991

GLPASE1 835.562
GLUDxm -108.512
GLUt2m 666.228
GLYOXm 673.621
GLYtm -1000
GSNt -864.049
H2O2t 1000
H2O2tly 478.661
H2Otly 1000
H2Otm -1000
H2Otn 1000
HPYRRy 613.735
Htr 1000
ICDHyrm -166.447
IDPtn 1000
ITPtn -1000
LALDO 237.152
LEUt5m -774.74
LEUTAm -774.74
MALTe 1.93991
MALtm -1000
MDHm 115.274
MELATNOX 387.839
MEOHtly 478.661
MI14Ptn -507.113
MI1PS 1000
MMMm 1000
MMTSADm 1000
NAt5 -1000
NDPK10 -1000
NDPK10n 1000
NDPK1m 1000
NDPK4n 1000
NDPK6n -1000
NDPK7n -992.887
NDPK9n 1000
NH4tn -492.887
NMNATr -1000
O16G2e 1.93991
O2St -1000
O2Stm 444.444
PAIL_HStn 753.557
PAIL45P_HStn -246.443
PEPCKm 1000
PEt 41.7153
PGLYCt 10.9776
PI45PLC 246.443
PI4PLCn 507.113
PIter 1000
PItn -1000
PPItr -1000
PRDXl 478.661
PSSA1_hs -129.535
PSt3 913.83
PYNP2r 389.973
RTOTALt -978.661
Rtotaltl -978.661
SMS 13.1728
SOAT12 138.889
SPHINGStl -978.661
SPODMm 222.222
SUCCt2m -1000
SUCCt4_2 -1000
SUCD1m 333.333
SUCOAS1m -1000
SUCOASm -1000
THMTPt 1000
THYMDtm 9.86193

TMDK1m 9.86193
TRDR 1000
UMPK5 69.6701
UMPK6 -679.697
URAt -1000
URATEt 1000
URIDK2m -349.71
UTPtn 21.339
XOLESTte -138.889
EX_ahdt[e] 1000
EX_ctp[e] 9.69955
EX_dtmp[e] -1000
EX_dttp[e] 1000
EX_HC00250[e] -1000
EX_HC01361[e] -1000
r0139 -9.69955
r0149 -9.69955
r0160 613.735
r0193 -1000
r0196 -1000
r0276 34.6648
r0280 1000
r0391 389.973
r0407 -103.657
r0408 -103.657
r0413 1000
r0474 679.697
r0475 650.29
r0494 -1000
r0497 -1000
r0509 666.667
r0517 -1000
r0527 1000
r0531 -650.29
r0617 -1000
r0642 -381.907
r0643 -618.093
r0707 -1000
r0752 -388.669
r0753 388.669
r0801 -1000
r0818 349.71
r0838 108.512
r0853 339.848
r0885 -1000
r0892 -1000
r0940 -1000
r1050 154.258
r1109 884.726
r1116 -1000
r1156 -1000
r1384 -1000
r1423 1000
r2093 -703.042
r2374 281.721
r2420 -281.721
r2520 326.379
RE0124C -1000
RE0344C -138.889
RE0456N 1000
RE1233C 111.306
RE1447N -246.443
RE1448N -246.443
RE1530C -1000
RE1918C 918.523
RE2112C 843.982
RE2426C -387.839
RE2640C -884.726

RE2677N 978.661
RE3272N -507.113
RE3273C -779.903
RE3301C 779.903
RE3352C -1000
EX_crm_hs[e] -991.834
CITt4_4 718.279
INSTt4_2 771.121
PIt8 -1000
PIt9 -1000
biomass_reaction 753.336
3HCO3_NAt 159.139
DTMPKm 9.86193
G6PDH2c 896.343
GNDc 563.01
PGLc 896.343
RPEc 896.343
THMDt5le 9.86193
CBASPte 265.632
EX_cbasp[e] -265.632
GLYALDtr -613.735
PEPtr -1000
LKYNRtr 111.306
DCMPtr 1000
FUMtr -884.726
XTSNtr 1000
UDPGLCURtr 1000
IMPtr -965.335
NICRNtr -610.027
EX_dcmp[e] -1000
EX_glyald[e] 613.735
EX_Lkynr[e] -111.306
EX_pep[e] -1000
EX_xtsn[e] -1000
EX_udpglcur[e] -1000
EX_nicrnt[e] 610.027
FOLOAT1tc 1000
GSNt2r -135.951
NACSMCTte -389.973
HMCRNc -734.368
ALLTNt -1000
EX_HC00900[e] -1000
EX_hmcr[e] -734.368
EX_mi1p_D[e] 1000
HC00900t4 -1000
HMCRNt -734.368
MI1Pt -1000
EX_5aop[e] -1000
EX_alltn[e] -1000
EX_alahisala[e] 309.547
EX_glylyscys[e] 288.307
EX_hiscyscys[e] 185.441
EX_tyrcysgly[e] 305.727
ALAHISALAt -309.547
GLYLYSCYSt -288.307
HISCYSCYSt -185.441
TYRCYSGLYt -305.727
ALAHISALAr -309.547
GLYLYSCYSr -288.307
HISCYSCYSr -185.441
TYRCYSGLYr -305.727
2MOPtm 381.907
MMALtm -618.093
5AOPt2 1000
DM_mi145p[c] 246.443
DM_mi14p[c] 507.113
DM_C02712[c] 884.726
sink_Tyr_ggn[c] -1.93991

sink_chol[c] -129.535
sink_cholate[c] -843.982
sink_glygn2[c] -833.622
sink_nad[c] -1000
sink_odecoa[c] 138.889
sink_thmtp[c] -1000
sink_tmndnccoa[c] -111.111
DXTRNt 835.562
EX_dxtrn[e] 835.562
sink_dchac[c] -156.018
EX_glcn[e] 1000
ADK1 -1000
CO2t -1000
CYTDt2r -650.29
DURIPP 1000
ENO 1000
EX_adn[e] 27.0433
EX_co2[e] 1000
EX_cytd[e] 650.29
EX_dad_2[e] 312.599
EX_dcyt[e] 170.013
EX_duri[e] 492.887
EX_fum[e] 884.726
EX_o2[e] -1000
EX_pi[e] 980.601
EX_succ[e] 1000
EX_thymd[e] -9.86193
EX_uri[e] 610.027
FBA -344.515
FUM -826.599
GALU 1000
GAPD 1000
GK1 34.6648
H2CO3D 522.582
H2Ot 1000
LEUTA 774.74
NDPK1 -1000
NDPK2 -518.853
NDPK3 -650.29
NDPK4 -990.138
NDPK5 -642.833
NDPK6 -1000
NDPK8 -920.399
NDPK9 -1000
O2t 1000
PGI -1000
PGK -493.314
PGM -1000
PGMT 1000
PIt6b -1000
PPM 1000
PUNP1 610.027
RNDR1 322.53
RNDR2 7.45652
RNDR4 1000
RPI 103.657
SPODM 76.8946
TALA 1000
TKT1 896.343
TKT2 896.343
TPI -448.172
UDPG4E 978.661
URIDK3 -1000
URIt2r -610.027
r0345 -312.599
r0570 1000
NTD6 312.599
NTD7 637.07

EX_nac[e] 389.973
EX_nh4[e] 1000
ALCD1 -478.661
EX_cit[e] -718.279
EX_ura[e] 1000
LGTHL 436.469
PIt7 -1000
PYK 1000
r0392 -613.735
EX_pyr[e] 1000
PYRt2r -296.958
DCMPDA -349.71
EX_ade[e] 610.027
EX_acald[e] 81.4766
ACALDt -81.4766
C09642te 918.523
EX_C09642[e] 918.523
12PPDRte -237.152
EX_12ppd_R[e] -237.152
GLCNte 1000
SPHGNSte 978.661
EX_sphings[e] 978.661
FDPte 344.515
EX_fdp[e] 344.515
CRMte 991.834
NH4tr -1000
UDPGALt2n 978.661
GALSIDEtn -978.661
CERT1tn 978.661
HMR_0793 978.661
HMR_2294 138.889
HMR_4343 389.973
HMR_4782 1000
HMR_6611 -1000
HMR_6617 1000
HMR_6619 -1000
HMR_7748 896.343
HMR_7749 -896.343
HMR_8475 329.987
HMR_8476 -500
HMR_8510 673.621
HMR_8585 506.686
HMR_8884 21.339
HMR_9187 506.686
HMR_9674 507.113
DCA3GSc 1000
DM_dca3g[c] 1000
EX_M01966[e] 506.686
sink_his_L[c] -590.214
sink_ile_L[c] -215.513
sink_leu_L[c] -410.978
sink_met_L[c] -1000
sink_phe_L[c] -195.465
sink_thr_L[c] -235.561
sink_trp_L[c] -10.0239
sink_val_L[c] -265.632
sink_arg_L[c] -270.644
sink_asn_L[c] -210.501
sink_gln_L[c] -245.585
sink_glu_L[c] -457.139
sink_pro_L[c] -310.739
sink_tyr_L[c] -426.013
DM_kynate[c] 111.306
DCMPtm 329.987
ATPS4mi 1000
CYOR_u10mi 666.667
CYOOm2i 333.333

Display the number of active reactions

```
fprintf('%u%s\n',nnz(v),' active reactions in the sparse flux balance analysis solution.');
```

```
435 active reactions in the sparse flux balance analysis solution.
```

ANTICIPATED RESULTS

Typically, a sparse flux balance analysis solution will have a small fraction of the number of the number of reactions active than in a flux balance analysis solution, e.g., Recon3.0model has 10,600 reactions. When maximising biomass production, a typical flux balance analysis solution might have approximately 3,000 active reactions (this is LP solver dependent) whereas for the same problem there are 435 active reactions in the sparse flux balance analysis solution from optimizeCbModel (using the default capped L1 norm approximate step function, see below).

B. Sparse flux balance analysis (two steps, all approximations, with a sparsity test)

This approach computes a sparse flux balance analysis solution, satisfing the FBA objection, with the default approach to approximate the solution to the cardinality minimisation problem underling sparse FBA. This approach does not check the quality of the solution, i.e., whether indeed it is the sparsest flux vector satisfing the optimality criterion $c^T v = \rho^\star$.

Solve a flux balance analysis problem

Build a linear programming problem structure (LPproblem) that is compatible with the interfacefunction (solveCobraLP) to any installed linear optimisation solver.

```
[c,S,b,lb,ub,csense] = deal(model.c,model.S,model.b,model.lb,model.ub,model.csense);
[m,n] = size(S);

LPproblem = struct('c',c,'osense',-1,'A',S,'csense',csense,'b',b,'lb',lb,'ub',ub);
```

Now solve the flux balance analysis problem

```
LPsolution = solveCobraLP(LPproblem);
if LPsolution.stat == 1
    vFBA = LPsolution.full(1:n);
else
    vFBA = [];
    error('FBA problem error!')
end
```

Display the number of active reactions

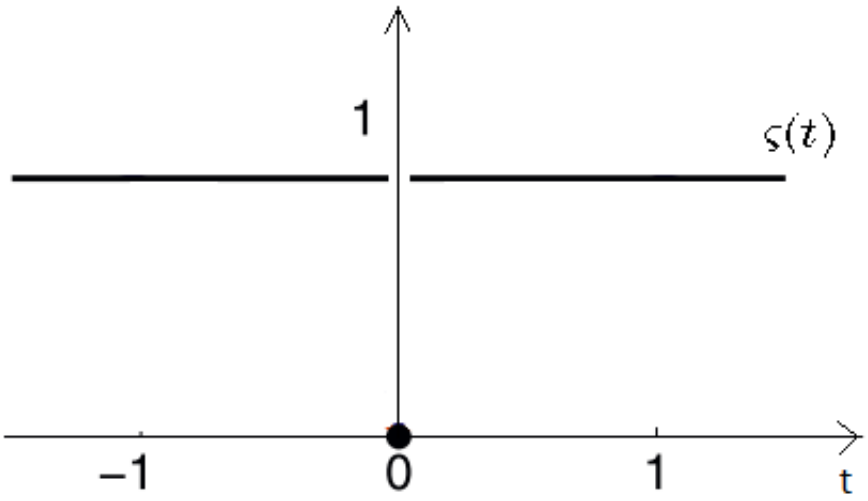
```
fprintf('%u%s\n',nnz(vFBA),' active reactions in the flux balance analysis solution.');
```

```
2997 active reactions in the flux balance analysis solution.
```

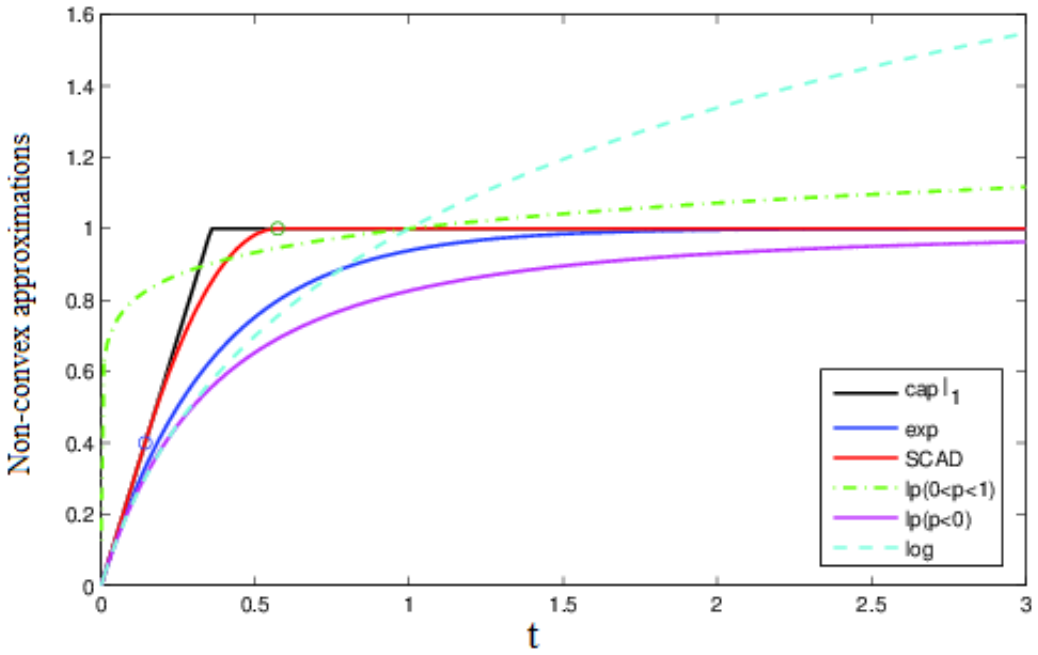
Approimations underlying sparse flux balance analysis

Due to its combinatorial nature, minimising the zero norm explicitly is an NP-hard problem. Therefore we approximately solve the problem. The approach is to replace the zero norm with a separable sum of

step functions, which are each approximated by anther function. Consider the step function ς(t): R → R where ς (t)=1 if t ≠ 0 and ς (t)=0 otherwise, illustrated in the Figure below:



There are then many different approximate step functions that can be minimised. The figure below illustrates the many different approximate step functions that can be chosen to be minimised instead of an explicit step function.



Depending on the application, and the biochemical network, one or other approximation may outperform the rest, therefore a pragmatic strategy is to try each and select the most sparse flux vector. The step set of function approximations available are

* 'cappedL1' : Capped-L1 norm

* 'exp' : Exponential function

* 'log' : Logarithmic function

* 'SCAD' : SCAD function

* 'lp-' : L_p norm with p<0

* 'lp+' : L_p norm with 0<p<1

Here we prepare a cell array of strings which indicate the set of step function approximations we wish to compare.

```
approximations = {'cappedL1','exp','log','SCAD','lp-','lp+'};
```

Run the sparse linear optimisation solver

First we must build a problem structure to pass to the sparse solver, by adding an additional constraint requiring that the sparse flux solution also statisfy the optimal objective value from flux balance analysis

```
constraint.A = [S ; c'];
constraint.b = [b ; c'*vFBA];
constraint.csense = [csense;'E'];
constraint.lb = lb;
constraint.ub = ub;
```

Now we call the sparse linear step function approximations

```
bestResult = n;
bestAprox = '';
for i=1:length(approximations)
    solution = sparseLP(char(approximations(i)),constraint);
    if solution.stat == 1
        nnzSol=nnz(abs(solution.x)>feasTol);
        fprintf('%u%s%s',nnzSol,' active reactions in the sparseFBA solution with ', char(appr
        if bestResult > nnzSol
            bestResult=nnzSol;
            bestAprox = char(approximations(i));
            solutionL0 = solution;
        end
    end
end
```

```
 434 active reactions in the sparseFBA solution with cappedL1
 433 active reactions in the sparseFBA solution with exp
 433 active reactions in the sparseFBA solution with log
 433 active reactions in the sparseFBA solution with SCAD
 433 active reactions in the sparseFBA solution with lp-
 433 active reactions in the sparseFBA solution with lp+
```

Select the most sparse flux vector, unless there is a numerical problem.

```
if ~isequal(bestAprox,'')
    vBest = solutionL0.x;
else
    vBest = [];
    error('Min L0 problem error !!!!')
end
```

Report the best approximation

```
display(strcat('Best step function approximation: ',bestAprox));
```

```
Best step function approximation:exp
```

Report the number of active reactions in the most sparse flux vector

```
fprintf('%u%s',nnz(abs(vBest)>feasTol),' active reactions in the best sparse flux balance anal
```

```
433 active reactions in the best sparse flux balance analysis solution.
```

Warn if there might be a numerical issue with the solution

```
feasError=norm(constraint.A * solutionL0.x - constraint.b,2);
if feasError>feasTol
    fprintf('%g\t%s\n',feasError, ' feasibily error.')
    warning('Numerical issue with the sparseLP solution')
end
```

Heuristically check if the selected set of reactions is minimal

Each step function approximation minimises a different problem than minimising the zero norm explicitly. Therefore it is wise to test, at least heuristically, if the most sparse approximate solution to minimising the zero norm is at least locally optimal, in the sense that the set of preicted reactions cannot be reduced by omitting, one by one, an active reaction. If it is locally optimal in this sense, one can be more confident that the most sparse approximate solution is the most sparse solution, but still there is no global guaruntee, as it is a combinatorial issue.

Identify the set of predicted active reactions

```
activeRxnBool = abs(vBest)>feasTol;
nActiveRnxs = nnz(activeRxnBool);
activeRxns = false(n,1);
activeRxns(activeRxnBool) = true;
minimalActiveRxns=activeRxns;
```

Close all predicted non-active reactions by setting their lb = ub = 0

```
lbSub = model.lb;
ubSub = model.ub;
lbSub(~activeRxns) = 0;
lbSub(~activeRxns) = 0;
```

Generate an LP problem to be reduced

```
% Check if one still can achieve the same objective
LPproblem = struct('c',-c,'osense',-1,'A',S,'csense',csense,'b',b,'lb',lbSub,'ub',ubSub);
```

For each active reaction in the most sparse approximate flux vector, one by one, set the reaction bounds to zero, then test if the optimal flux balance analysis objective value is still attained. If it is, then that reaction is not part of the minimal set. If it is not, then it is probably part of the minimal set.

```
for i=1:n
    if activeRxnBool(i)
        LPproblem.lb = model.lb;
        LPproblem.ub = model.ub;
```

```
        %close bounds on this reaction
        LPproblem.lb(i) = 0;% Close the reaction
        LPproblem.ub(i) = 0;% Close the reaction
        %solve the LP problem
        LPsolution = solveCobraLP(LPproblem);
        %check if the optimal FBA objective is attained
        if LPsolution.stat == 1 && abs(LPsolution.obj + c'*vFBA)<1e-8
            minimalActiveRxns(i) = 0;
            vBestTested = LPsolution.full(1:n);
        else
            %relax those bounds if reaction appears to be part of the minimal set
            LPproblem.lb(i) = model.lb(i);
            LPproblem.ub(i) = model.ub(i);
        end
    end
end
```

Report the number of active reactions in the approximately most sparse flux vector, or the reduced approximately most sparse flux vector, if it is more sparse.

```
if nnz(minimalActiveRxns)<nnz(activeRxns)
    fprintf('%u%s',nnz(abs(vBestTested)>feasTol),' active reactions in the best sparseFBA solu
    nonZeroFlag = 1;
    printFluxVector(model, vBestTested, nonZeroFlag);
else
    fprintf('%u%s',nnz(abs(vBest)>feasTol),' active reactions in the best sparseFBA solution (
end
```

```
433 active reactions in the best sparseFBA solution (tested).
```