Sparse Flux Balance Analysis

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

Reviewer: Stefania Magnusdottir, Molecular Systems Physiology Group, University of Luxembourg.

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

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

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. 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[1], 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

$$
\begin{array}{ll}
\min\limits_{v} & \|v\|_0 \\
\text{s.t.} & Sv = b \\
 & l \leq v \leq u \\
 & c^T v = \rho^\star
\end{array}
$$

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

EQUIPMENT SETUP

Initialize the COBRA Toolbox.

If necessary, initialize The Cobra Toolbox using the `initCobraToolbox` function.

```
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: C:\Program Files\IBM\ILOG\CPLEX_Studio1271\cplex\matlab\x64_win64
  - [----] GUROBI_PATH :  --> set this path manually after installing the solver ( see instructions )
  - [---*] TOMLAB_PATH: C:\Program Files\tomlab\
  - [----] MOSEK_PATH :  --> set this path manually after installing the solver ( see instructions )
  Done.
> Checking available solvers and solver interfaces ... Done.
> Setting default solvers ... Done.
> Saving the MATLAB path ... Done.
  - The MATLAB path was saved in the default location.

> Summary of available solvers and solver interfaces

    Support           LP   MILP     QP   MIQP    NLP
-----------------------------------------------------------------------
cplex_direct  full              0      0      0      0      -
dqqMinos      full              0      -      -      -      -
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              0      -      -      -      0
tomlab_cplex  full              1      1      1      1      -
qpng          experimental      -      -      1      -      -
tomlab_snopt  experimental      -      -      -      -      1
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      4      5      2      2

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


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

> Checking for available updates ...
--> You cannot update your fork using updateCobraToolbox(). [3d2698 @ Tutorial-sparseFBA].
    Please use the MATLAB.devTools (https://github.com/opencobra/MATLAB.devTools).
```

COBRA model.

In this tutorial, the model used is the generic reconstruction of human metabolism, the Recon 2.04[2], which is provided in the COBRA Toolbox. The Recon 2.04 model can also be downloaded from the

Virtual Metabolic Human webpage. You can also select your own model to work with. Before proceeding with the simulations, the path for the model needs to be set up:

```
if 0
    % Using own model, change "if 0" to "if 1" and change the filename and directory
    filename = 'Recon3.0model';
    directory = '~/work/sbgCloud/programReconstruction/projects/recon2models/data/reconXCompar
    model = loadIdentifiedModel(filename, directory);
    % model = convertOldStyleModel(model);%convert to new COBRA format style if needed.
else
    % Default use of Recon 2.04
    global CBTDIR
    load([CBTDIR filesep 'test' filesep 'models' filesep 'Recon2.v04.mat']);
    model = modelR204;
    clear modelR204;
```

Recon 2.04 is written in the "old style" COBRA format, and we thus use the function `convertOldStyleModel` to convert it to the new COBRA Toolbox format.

```
    model = convertOldStyleModel(model);
end
```

NOTE: The following text, code, and results are shown for the Recon 2.04 model

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');
```

Display the constraints

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

```
MinConstraints:
DM_T_antigen_g_ -1
EX_10fthf(e) -1
EX_10fthf5glu(e) -1
EX_10fthf6glu(e) -1
EX_10fthf7glu(e) -1
EX_11_cis_retfa(e) -1
EX_13_cis_retnglc(e) -1
EX_1glyc_hs(e) -1
EX_1mncam(e) -1
EX_2425dhvitd2(e) -1
EX_2425dhvitd3(e) -1
EX_24nph(e) -1
EX_25hvitd2(e) -1
EX_25hvitd3(e) -1
EX_2hb(e) -1
EX_2mcit(e) -1
EX_34dhoxpeg(e) -1
EX_34dhphe(e) -1
EX_35cgmp(e) -1
EX_3aib(e) -1
```

```
EX_3aib_D(e) -1
EX_3mlda(e) -1
EX_4abut(e) -1
EX_4hdebrisoquine(e) -1
EX_4hphac(e) -1
EX_4mptnl(e) -1
EX_4mtolbutamide(e) -1
EX_4nph(e) -1
EX_4nphsf(e) -1
EX_4pyrdx(e) -1
EX_5adtststerone(e) -1
EX_5adtststeroneglc(e) -1
EX_5adtststerones(e) -1
EX_5dhf(e) -1
EX_5fthf(e) -1
EX_5homeprazole(e) -1
EX_5htrp(e) -1
EX_5thf(e) -1
EX_6dhf(e) -1
EX_6htststerone(e) -1
EX_6thf(e) -1
EX_7dhf(e) -1
EX_7thf(e) -1
EX_9_cis_retfa(e) -1
EX_abt(e) -1
EX_ac(e) -1
EX_acac(e) -1
EX_acald(e) -1
EX_acetone(e) -1
EX_acgalfucgalacgalfuc12gal14acglcgalgluside_hs(e) -1
EX_acgalfucgalacgalfucgalacglcgal14acglcgalgluside_hs(e) -1
EX_acgam(e) -1
EX_ach(e) -1
EX_acn13acngalgbside_hs(e) -1
EX_acn23acngalgbside_hs(e) -1
EX_acnacngal14acglcgalgluside_hs(e) -1
EX_acnacngalgbside_hs(e) -1
EX_acngalacglcgal14acglcgalgluside_hs(e) -1
EX_ade(e) -1
EX_adp -1
EX_adprbp(e) -1
EX_adprib(e) -1
EX_adrn(e) -1
EX_adrnl(e) -1
EX_aflatoxin(e) -1
EX_ahandrostanglc(e) -1
EX_ak2lgchol_hs(e) -1
EX_akg(e) -1
EX_ala_B(e) -1
EX_ala_D(e) -1
EX_ala_L(e) -1
EX_aldstrn(e) -1
EX_amp(e) -1
EX_andrstrn(e) -1
EX_andrstrnglc(e) -1
EX_antipyrene(e) -1
EX_apnnox(e) -1
EX_appnn(e) -1
EX_aprgstrn(e) -1
EX_aqcobal(e) -1
EX_arab_L(e) -1
EX_arachd(e) -1
EX_arg_L(e) -1
EX_ascb_L(e) -100
EX_asn_L(e) -1
EX_asp_D(e) -1
EX_asp_L(e) -1
EX_atp(e) -1
```

```
EX_avite1(e) -1
EX_avite2(e) -1
EX_bhb(e) -1
EX_bildglcur(e) -1
EX_bilglcur(e) -1
EX_bilirub(e) -1
EX_biocyt(e) -1
EX_btn(e) -100
EX_but(e) -1
EX_bvite(e) -1
EX_bz(e) -1
EX_ca2(e) -1
EX_camp(e) -1
EX_caro(e) -1
EX_carveol(e) -1
EX_cca_d3(e) -1
EX_cgly(e) -1
EX_chsterol(e) -1
EX_chtn(e) -1
EX_cit(e) -1
EX_CLPND(e) -1
EX_cmp(e) -1
EX_co(e) -1
EX_co2(e) -100
EX_coumarin(e) -1
EX_creat(e) -1
EX_crmp_hs(e) -1
EX_crtsl(e) -1
EX_crtstrn(e) -1
EX_crvnc(e) -1
EX_csn(e) -1
EX_cspg_a(e) -1
EX_cspg_b(e) -1
EX_cspg_c(e) -1
EX_cspg_d(e) -1
EX_cspg_e(e) -1
EX_cyan(e) -1
EX_dcsptn1(e) -1
EX_debrisoquine(e) -1
EX_dgchol(e) -1
EX_dheas(e) -1
EX_dhf(e) -1
EX_digalsgalside_hs(e) -1
EX_dlnlcg(e) -1
EX_dmantipyrine(e) -1
EX_dmhptcrn(e) -1
EX_dopa(e) -1
EX_dopasf(e) -1
EX_drib(e) -1
EX_duri(e) -1
EX_eaflatoxin(e) -1
EX_ebastine(e) -1
EX_ebastineoh(e) -1
EX_eicostet(e) -1
EX_elaid(e) -1
EX_estradiol(e) -1
EX_estradiolglc(e) -1
EX_estriolglc(e) -1
EX_estroneglc(e) -1
EX_estrones(e) -1
EX_etoh(e) -1
EX_fe2(e) -1
EX_fe3(e) -1
EX_for(e) -1
EX_fru(e) -1
EX_fuc13galacglcgal14acglcgalgluside_hs(e) -1
EX_fuc14galacglcgalgluside_hs(e) -1
EX_fucacgalfucgalacglcgalgluside_hs(e) -1
```

```
EX_fucacngal14acglcgalgluside_hs(e) -1
EX_fucacngalacglcgalgluside_hs(e) -1
EX_fucfuc12gal14acglcgalgluside_hs(e) -1
EX_fucfuc132galacglcgal14acglcgalgluside_hs(e) -1
EX_fucfucfucgalacglc13galacglcgal14acglcgalgluside_hs(e) -1
EX_fucfucfucgalacglcgal14acglcgalgluside_hs(e) -1
EX_fucfucgalacglcgalgluside_hs(e) -1
EX_fucgal14acglcgalgluside_hs(e) -1
EX_fucgalfucgalacglcgalgluside_hs(e) -1
EX_fucgalgbside_hs(e) -1
EX_fuc_L(e) -1
EX_galacglcgalgbside_hs(e) -1
EX_galfuc12gal14acglcgalgluside_hs(e) -1
EX_galfucgalacglcgal14acglcgalgluside_hs(e) -1
EX_galgalfucfucgalacglcgalacglcgal14acglcgalgluside_hs(e) -1
EX_galgalgalthcrm_hs(e) -1
EX_gbside_hs(e) -1
EX_gchola(e) -1
EX_gd1b2_hs(e) -1
EX_gd1c_hs(e) -1
EX_gdp(e) -1
EX_glc(e) -1
EX_gln_L(e) -1
EX_gluala(e) -1
EX_glu_L(e) -1
EX_glyb(e) -1
EX_glyc_S(e) -1
EX_glygn2(e) -1
EX_glygn4(e) -1
EX_glygn5(e) -1
EX_gmp(e) -1
EX_gp1c_hs(e) -1
EX_gp1calpha_hs(e) -1
EX_gq1b_hs(e) -1
EX_gq1balpha_hs(e) -1
EX_gt1a_hs(e) -1
EX_gthox(e) -1
EX_gthrd(e) -1
EX_gtp(e) -1
EX_gua(e) -1
EX_h(e) -100
EX_h2o(e) -100
EX_h2o2(e) -1
EX_ha(e) -1
EX_ha_pre1(e) -1
EX_hco3(e) -100
EX_hcoumarin(e) -1
EX_hdca(e) -1
EX_hestratriol(e) -1
EX_hexc(e) -1
EX_hista(e) -1
EX_hom_L(e) -1
EX_hpdca(e) -1
EX_hspg(e) -1
EX_htaxol(e) -1
EX_hxan(e) -1
EX_i(e) -1
EX_idp(e) -1
EX_ile_L(e) -1
EX_imp(e) -1
EX_inost(e) -1
EX_k(e) -1
EX_ksi(e) -1
EX_ksi_deg1(e) -1
EX_ksii_core2(e) -1
EX_ksii_core4(e) -1
EX_lac_D(e) -1
EX_lac_L(e) -1
```

EX_lcts(e) -1
EX_Lcystin(e) -1
EX_leuktrA4(e) -1
EX_leuktrB4(e) -1
EX_leuktrC4(e) -1
EX_leuktrD4(e) -1
EX_leuktrE4(e) -1
EX_leuktrF4(e) -1
EX_leu_L(e) -1
EX_lgnc(e) -1
EX_limnen(e) -1
EX_lipoate(e) -1
EX_lneldc(e) -1
EX_lnlc(e) -1
EX_lnlnca(e) -1
EX_lnlncg(e) -1
EX_lys_L(e) -1
EX_malttr(e) -1
EX_meoh(e) -1
EX_mepi(e) -1
EX_mercplaccys(e) -1
EX_met_L(e) -1
EX_mthgxl(e) -1
EX_n2m2nmasn(e) -1
EX_nac(e) -1
EX_nad(e) -1
EX_nadp(e) -1
EX_ncam(e) -1
EX_nh4(e) -100
EX_nifedipine(e) -1
EX_no(e) -1
EX_npthl(e) -1
EX_nrpphr(e) -1
EX_nrpphrsf(e) -1
EX_nrvnc(e) -1
EX_o2s(e) -1
EX_oagd3_hs(e) -1
EX_oagt3_hs(e) -1
EX_ocdcea(e) -1
EX_omeprazole(e) -1
EX_onpthl(e) -1
EX_orn(e) -1
EX_oxa(e) -1
EX_paf_hs(e) -1
EX_pchol_hs(e) -1
EX_peplys(e) -1
EX_perillyl(e) -1
EX_pglyc_hs(e) -1
EX_pheacgln(e) -1
EX_phyQ(e) -1
EX_phyt(e) -1
EX_pi(e) -100
EX_pnto_R(e) -100
EX_ppa(e) -1
EX_prgstrn(e) -1
EX_pro_D(e) -1
EX_pro_L(e) -1
EX_prostgd2(e) -1
EX_prostge1(e) -1
EX_prostge2(e) -1
EX_prostgf2(e) -1
EX_ps_hs(e) -1
EX_ptdca(e) -1
EX_pyr(e) -1
EX_rbt(e) -1
EX_retfa(e) -1
EX_retinol(e) -100
EX_retinol_9_cis(e) -1

EX_retinol_cis_11(e) -1
EX_retn(e) -100
EX_retnglc(e) -1
EX_Rtotal(e) -1
EX_Rtotal2(e) -1
EX_Rtotal3(e) -1
EX_s2l2fn2m2masn(e) -1
EX_s2l2n2m2masn(e) -1
EX_sarcs(e) -1
EX_sel(e) -1
EX_ser_D(e) -1
EX_ser_L(e) -1
EX_sl_L(e) -1
EX_so4(e) -100
EX_spc_hs(e) -1
EX_sph1p(e) -1
EX_sphs1p(e) -1
EX_srtn(e) -1
EX_strch1(e) -1
EX_strch2(e) -1
EX_strdnc(e) -1
EX_succ(e) -1
EX_sucr(e) -1
EX_tag_hs(e) -1
EX_tagat_D(e) -1
EX_taur(e) -1
EX_taxol(e) -1
EX_tchola(e) -1
EX_tcynt(e) -1
EX_tdchola(e) -1
EX_tethex3(e) -1
EX_tetpent3(e) -1
EX_tetpent6(e) -1
EX_tettet6(e) -1
EX_thf(e) -1
EX_thmmp(e) -1
EX_thmtp(e) -1
EX_thr_L(e) -1
EX_thym(e) -1
EX_thymd(e) -1
EX_thyox_L(e) -1
EX_tmndnc(e) -1
EX_tolbutamide(e) -1
EX_tre(e) -1
EX_triodthy(e) -1
EX_triodthysuf(e) -1
EX_trp_L(e) -1
EX_tststerone(e) -1
EX_tststeroneglc(e) -1
EX_tststerones(e) -1
EX_tsul(e) -1
EX_txa2(e) -1
EX_tymsf(e) -1
EX_Tyr_ggn(e) -1
EX_tyr_L(e) -1
EX_udp(e) -1
EX_ump(e) -1
EX_ura(e) -1
EX_urate(e) -1
EX_urea(e) -1
EX_uri(e) -1
EX_utp(e) -1
EX_vacc(e) -1
EX_val_L(e) -1
EX_vitd2(e) -100
EX_whddca(e) -1
EX_whhdca(e) -1
EX_whtststerone(e) -1

EX_whttdca(e) -1
EX_xolest_hs(e) -1
EX_xolest2_hs(e) -1
EX_xoltri24(e) -1
EX_xoltri25(e) -1
EX_xoltri27(e) -1
EX_xyl_D(e) -1
EX_yvite(e) -1
sink_pre_prot(r) -1
EX_4abutn(e) -1
EX_acmana(e) -1
EX_ahdt(e) -1
EX_ctp(e) -1
EX_dgmp(e) -1
EX_dgtp(e) -1
EX_dha(e) -1
EX_dhap(e) -1
EX_dtmp(e) -1
EX_dttp(e) -1
EX_fad(e) -1
EX_fald(e) -1
EX_g1p(e) -1
EX_HC00229(e) -1
EX_HC00250(e) -1
EX_HC01104(e) -1
EX_HC01361(e) -1
EX_HC01440(e) -1
EX_HC01441(e) -1
EX_HC01444(e) -1
EX_HC01446(e) -1
EX_HC01577(e) -1
EX_HC01609(e) -1
EX_HC01610(e) -1
EX_HC01700(e) -1
EX_HC02160(e) -1
EX_HC02161(e) -1
EX_itp(e) -1
EX_orot(e) -1
EX_prpp(e) -1
EX_ptrc(e) -1
EX_pydx5p(e) -1
EX_spmd(e) -1
EX_udpg(e) -1
EX_no2(e) -1
EX_so3(e) -1
EX_sprm(e) -1
EX_prostgh2(e) -1
EX_prostgi2(e) -1
EX_ppi(e) -1
EX_cdp(e) -1
EX_dtdp(e) -1
EX_HC00955(e) -1
EX_HC00001(e) -1
EX_HC00002(e) -1
EX_HC00003(e) -1
EX_HC00004(e) -1
EX_citr_L(e) -1
EX_HC01787(e) -1
EX_C02470(e) -1
EX_HC01852(e) -1
EX_HC01939(e) -1
EX_HC01942(e) -1
EX_HC01943(e) -1
EX_HC01944(e) -1
EX_HC00822(e) -1
EX_C02528(e) -1
EX_HC02192(e) -1
EX_HC02193(e) -1

EX_HC02195(e) -1
EX_HC02196(e) -1
EX_HC02220(e) -1
EX_HC02154(e) -1
EX_HC02175(e) -1
EX_HC02176(e) -1
EX_HC02199(e) -1
EX_HC02200(e) -1
EX_HC02201(e) -1
EX_HC02172(e) -1
EX_HC02191(e) -1
EX_HC02194(e) -1
EX_HC02197(e) -1
EX_HC02198(e) -1
EX_HC02187(e) -1
EX_HC02180(e) -1
EX_HC02179(e) -1
EX_HC02202(e) -1
EX_HC02203(e) -1
EX_HC02204(e) -1
EX_HC02205(e) -1
EX_HC02206(e) -1
EX_HC02207(e) -1
EX_HC02208(e) -1
EX_HC02210(e) -1
EX_HC02213(e) -1
EX_HC02214(e) -1
EX_HC02216(e) -1
EX_HC02217(e) -1
EX_malcoa(e) -1
EX_arachcoa(e) -1
EX_coa(e) -1
EX_CE2250(e) -1
EX_CE1935(e) -1
EX_CE1940(e) -1
EX_CE1943(e) -1
EX_CE2011(e) -1
EX_CE1936(e) -1
EX_CE1939(e) -1
EX_maltttr(e) -1
EX_maltpt(e) -1
EX_malthx(e) -1
EX_CE2915(e) -1
EX_CE4722(e) -1
EX_CE2916(e) -1
EX_CE4723(e) -1
EX_CE2917(e) -1
EX_CE4724(e) -1
EX_malthp(e) -1
EX_CE2839(e) -1
EX_CE2838(e) -1
EX_CE1950(e) -1
EX_cynt(e) -1
EX_23cump(e) -1
EX_3ump(e) -1
EX_CE5786(e) -1
EX_CE5788(e) -1
EX_CE5789(e) -1
EX_CE5797(e) -1
EX_CE5798(e) -1
EX_CE5787(e) -1
EX_CE5791(e) -1
EX_CE5867(e) -1
EX_CE5868(e) -1
EX_CE5869(e) -1
EX_CE4633(e) -1
EX_CE4881(e) -1
EX_CE5854(e) -1

```
EX_glcur(e) -1
EX_CE1926(e) -1
EX_udpgal(e) -1
EX_crm_hs(e) -1
EX_galside_hs(e) -1
EX_CE0074(e) -1
EX_cdpea(e) -1
EX_12dgr120(e) -1
EX_CE5853(e) -1
EX_CE1925(e) -1
EX_C05965(e) -1
EX_C04849(e) -1
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, satisfying the FBA objection, with the default approach to approximate the solution to the cardinality minimisation problem[3] 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);
```

```
3MOBt2im 0.127657
3MOPt2im 49.4471
4ABUTtm 0.0439253
ABTArm 0.0439253
ABUTt2r 0.0439253
ADEt -3.17842
ADK1 -3.03312
ADK1m -0.0837315
ADK3 -0.27054
ADNtm -0.0837315
ALAt2r 1
ALATA_L -0.137857
AMPDA 0.147159
ARGtiDF 1
R_group_phosphotase_1 0.0559212
ASNt4 0.893617
ASPGLUm 0.127657
ASPTAm -0.127657
BTNDe 0.893613
CATm 0.024882
CDIPTr -0.0372829
CHOLt4 0.493981
CLS_hs 0.0372829
CYOR_u10m 9.95278
CYSTA 0.150649
CYTK4 0.155035
DAGK_hs 0.0559212
DATPtn 0.04216
DCMPDA -0.155035
DCTPtn 0.030196
DESAT18_4 0.11773
DESAT18_7 0.11773
DGTPtn 0.0316544
DNDPt9m 0.0418657
DOPACHRMISO 2.29026
DSAT 0.0559212
DTTPtn 0.0418657
DURIK1 0.0932265
ENO 5.22241
EX_4abut(e) -0.0439253
EX_ade(e) 2.17842
EX_ala_L(e) -1
EX_amp(e) 1
EX_arg_L(e) -1
EX_asn_L(e) -0.893617
EX_asp_L(e) -1
EX_atp(e) -1
```

EX_biocyt(e) -0.893613
EX_btn(e) 0.893613
EX_chol(e) -0.493981
EX_chsterol(e) -0.0652435
EX_duri(e) -0.0932265
EX_gln_L(e) -1
EX_gly(e) -0.974083
EX_h2o2(e) -1
EX_hco3(e) -0.0837315
EX_his_L(e) -0.404253
EX_ile_L(e) -0.914893
EX_inost(e) -0.0745627
EX_leu_L(e) -1
EX_lneldc(e) 0.11773
EX_lpchol_hs(e) -0.0559212
EX_lys_L(e) -1
EX_mercplaccys(e) 0.150649
EX_met_L(e) -0.48936
EX_no(e) -0.148933
EX_o2(e) -6.68552
EX_ocdca(e) -0.11773
EX_orn(e) -0.84642
EX_pe_hs(e) -0.177089
EX_pglyc_hs(e) -0.0466021
EX_phe_L(e) -0.829787
EX_pi(e) 7.60762
EX_pro_L(e) -1
EX_ps_hs(e) -0.624468
EX_pyr(e) 4.20007
EX_ser_L(e) -1
EX_sph1p(e) -0.0559212
EX_thr_L(e) -1
EX_thymd(e) -0.0418657
EX_trp_L(e) -0.0425533
EX_tyr_L(e) -0.510637
EX_ura(e) 0.767268
EX_utp(e) -1
EX_val_L(e) -1
FACOAL1822 -0.11773
FATP3t -0.11773
FBA 0.435143
FUMm 0.0439253
G3PD2m 0.0559212
GAPD 5.22241
GK1 0.147159
GLNS 0.0425533
GLNt4 1
GLUDxm 0.647824
GLUt2m 48.4859
GLYt2r 0.974083
GPDDA1 0.0559212
GTHRDt -49.6274
H2O2t 1
H2Ot 1.56714
H2Otm -7.19359
HISt4 0.404253
HPYRRy 0.35051
ILEt4 0.914893
ILEt5m -49.4471
ILETA 49.4471
ILETAm -49.4471
INSTt2r 0.0745627
LEUt4 1
LNELDCt -0.11773
LPASE 0.0559212
LPCHOLt 0.0559212
LYSt4 1.89361
MCLACCYSR 0.150649

MCLOR -0.150649
MDHm 0.0439253
MERCPLACCYSt 0.150649
METtec 0.48936
NADH2_u10m 6.18955
NDPK6 -0.705459
NTD7 3.09469
O2t 6.68552
O2tm 4.90175
ORNt3m -0.84642
ORNTArm 0.84642
ORNtiDF 0.84642
P5CDm 0.430173
P5CRm 0.0837315
PCm 0.0837315
PEt 0.177089
PGI -0.880086
PGK -5.22241
PGLYCt 0.0466021
PGM -5.22241
PHEtec 0.829787
PPM 3.94569
PRO1xm 4.17729
PROt2r 1
PSSA1_hs -0.549902
PSt3 0.624468
PUNP1 3.17842
PYK 5.06486
PYNP2r 0.767268
PYRt2m 0.0837315
PYRt2r -4.20007
RNDR1 0.125891
RNDR2 0.0316544
RNDR4 0.0618088
RPI 2.63046
SMS 0.0559212
SPH1Pte -0.0559212
SPODMm 0.0497639
THMDt4 0.0418657
THRt4 1
TKT1 1.31523
TKT2 1.31523
TMDK1 0.0418657
TPI 1.80629
TRDR 0.0945153
TRIOK 0.35051
TRPt 0.0425533
TYRt 0.510637
UMPK -0.612233
UMPK6 -0.155035
URAt -0.767268
VALt4 1
VALt5m -0.127657
VALTAm -0.127657
EX_ahdt(e) 1
EX_dgmp(e) 0.539243
EX_dgtp(e) -0.539243
EX_HC00250(e) -0.450235
EX_HC01361(e) -1
EX_prpp(e) -1
r0010 0.5
r0047 0.0837315
r0051 -1
r0074 0.84642
r0145 -0.148933
r0160 0.35051
r0178 0.0439253
r0191 0.435143

```
r0193 -0.450235
r0276 0.147159
r0280 0.0840257
r0392 -0.35051
r0407 1.31523
r0408 0.921296
r0409 0.393933
r0410 0.539243
r0413 0.0316544
r0474 0.124839
r0509 0.0439253
r0707 -1
r0787 0.0559212
r0817 -0.148933
r0838 -0.647824
r0885 0.167463
r0892 -1
r0911 0.430173
r0940 -0.450235
r0941 0.0837315
r1050 0.0652435
r1116 -3.53924
r1117 0.0418657
r1143 1
r1156 -0.767268
r1423 5.66708
r1431 0.0837315
r1433 0.0837315
r1453 -3.66338
r2447 0.0932265
r2471 1
r2520 49.4599
RE0344C -0.11773
RE0452M 0.0418657
RE2675C 0.0559212
RE2954C 0.0418657
RE3198C 4.58051
RE3273C -0.111846
RE3301C 0.0559244
EX_ppi(e) -1
EX_citr_L(e) -0.148933
PIt9 -4.94055
CYOOm3 4.97639
biomass_reaction 3.19806
ALAALACNc 0.0643263
ALAALAPEPT1tc 0.0643263
LEULEULAPc 0.37234
LEULEUPEPT1tc 0.37234
PROGLYPEPT1tc 0.74932
PROGLYPRO1c 0.74932
3HCO3_NAt 0.0279105
DATPtm 0.0418657
DUTPDP 0.0618088
EX_alaala(e) -0.0643263
EX_leuleu(e) -0.37234
EX_progly(e) -0.74932
EX_dpcoa(e) -2.53924
EX_pan4p(e) 2.53924
FADH2ETC 0.0559212
NODe -0.148933
PTPATe -2.53924
RPEc 1.31523
3MOBte 0.127657
EX_3mob(e) -0.127657
```

Display the number of active reactions

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

```
805 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 reactions active than in a flux balance analysis solution, e.g., Recon 2.04 model has 7,440 reactions. When maximising biomass production, a typical flux balance analysis solution might have approximately 2,000 active reactions (this is LP solver dependent) whereas for the same problem there are 247 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, satisfying the FBA objection, with the default approach to approximate the solution to the cardinality minimisation problem[3] underlying 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 interface function (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

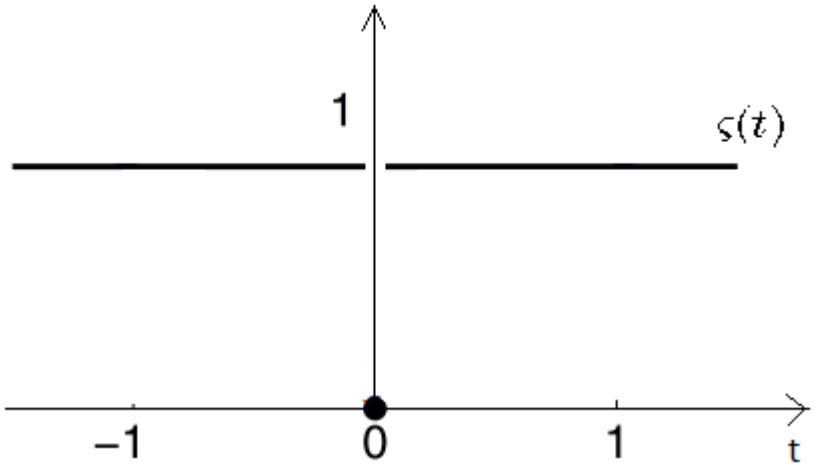
```
fprintf('%u%s\n',nnz(vFBA),' active reactions in the flux balance analysis solution.');
```

```
1876 active reactions in the flux balance analysis solution.
```

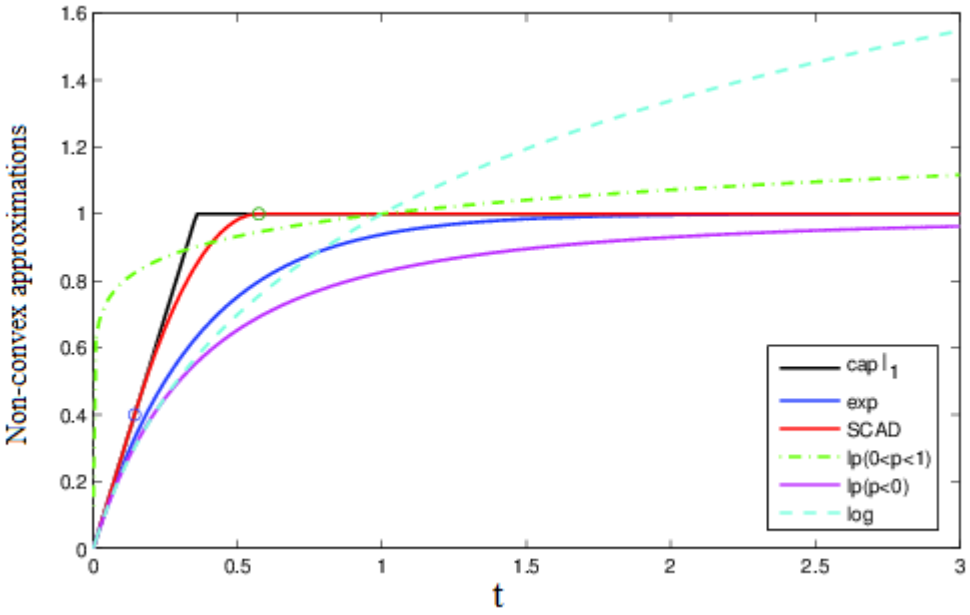
Approimations underlying sparse flux balance analysis

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[4] 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 satisfies 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
```

```
247 active reactions in the sparseFBA solution with cappedL1
247 active reactions in the sparseFBA solution with exp
247 active reactions in the sparseFBA solution with log
247 active reactions in the sparseFBA solution with SCAD
247 active reactions in the sparseFBA solution with lp-
247 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:cappedL1
```

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
```

```
247 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 predicted 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 guarantee, 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
```

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

REFERENCES

[1] Meléndez-Hevia, E., Isidoro, A. (1085). The game of the pentose phosphate cycle. Journal of Theoretical Biology 117, 251-263.

[2] Thiele, I., Swainston, N., Fleming, R.M., Hoppe, A., Sahoo, S., Aurich, M.K., Haraldsdottir, H., Mo, M.L., Rolfsson, O., Stobbe, M.D., et al. (2013). A community-driven global reconstruction of human metabolism. Nat Biotechnol 31, 419-425.

[3] Fleming, R.M.T., et al. (*submitted*, 2017). Cardinality optimisation in constraint-based modelling: illustration with Recon 3D.

[4] Le Thi, H.A., Pham Dinh, T., Le, H.M., and Vo, X.T. (2015). DC approximation approaches for sparse optimization. European Journal of Operational Research 244, 26-46.