Analyze Steady-State Community COBRA Models at Using SteadyCom

Author(s): Siu Hung Joshua Chan, Department of Chemical Engineering, The Pennsylvania State University

Reviewer(s):

INTRODUCTION

This tutorial demonstrates the use of SteadyCom to analyze a multi-organism COBRA model (e.g., for a microbial community) at a community steady-state [1]. Compared to the direct extension of flux balance analysis (FBA) which simply treats a community model as a multi-compartment model, SteadyCom explicitly introduces the biomass variables to describe the relationships between biomass, biomass production rate, growth rate and fluxes. SteadyCom also assumes the existence of a time-averaged population steady-state for a stable microbial community which in turn implies a time-averaged constant growth rate across all members. SteadyCom is equivalent to the reformulation of the earlier community flux balance analysis (cFBA) [2] with significant computational advantage. SteadyCom computes the maximum community growth rate by solving the follow optimization problem:

$$
\begin{aligned}
\max \quad & \mu \\
\text{s.t.} \quad & \sum_{j \in \mathbf{J}^k} S_{ij}^k V_j^k = 0, && \forall i \in \mathbf{I}^k, k \in \mathbf{K} \\
& LB_j^k X^k \le V_j^k \le UB_j^k X^k, && \forall j \in \mathbf{J}^k, k \in \mathbf{K} \\
& \sum_{k \in \mathbf{K}} V_{ex(i)}^k + u_i^{com} \ge 0, && \forall i \in \mathbf{I}^{com} \\
& V_{biomass}^k = X^k \mu, && \forall k \in \mathbf{K} \\
& \sum_{k \in \mathbf{K}} X^k = 1 && \\
& X^k, \quad \mu \ge 0, && \forall k \in \mathbf{K} \\
& V_j^k \in \Re, && \forall j \in \mathbf{J}^k, k \in \mathbf{K}
\end{aligned}
$$

where S_{ij}^k is the stoichiometry of metabolite *i* in reaction *j* for organism *k*, V_j^k, LB_j^k and UB_j^k are respectively the flux (in mmol/h), lower bound (in mmol/h/gdw) and upper bound (in mmol/h/gdw) for reaction *j* for organism *k*, u_i^{com} is the community uptake bound for metabolite *i*, X^k is the biomass (in gdw) of organism *k*, μ is the community growth rate, $\mathbf{I}^k$ is the set of metabolites of organism *k*, $\mathbf{I}^{com}$ is the set of community metabolites in the community exchange space, $\mathbf{J}^k$ is the set of reactions for organism k, $\mathbf{K}$ is the set of organisms in the community, and $ex(i) \in \mathbf{J}^k$ is the exchange reaction in organism *k* for extracellular metabolite *i*. See ref. [1] for the derivation and detailed explanation.

Throughout the tutorial, using a hypothetical model of four *E. coli* mutants auxotrophic for amino acids, we will demonstrate the three different functionalities of the module: (1) computing the maximum community growth rate using the function SteadyCom.m, (2) performing flux variability analysis under a given community growth rate using SteadyComFVA.m, and (3) analyzing the pairwise relationship between flux/biomass variables using a technique similar to Pareto-optimal analysis by calling the function SteadyComPOA.m

EQUIPMENT SETUP

If necessary, initialise the cobra toolbox and select a solver by running:

```
initCobraToolbox
```

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

 > Checking if git is installed ...  Done.
 > Checking if the repository is tracked using git ...  Done.
 > Checking if curl is installed ...  Done.
 > Checking if remote can be reached ...  Done.
 > Initializing and updating submodules ... Done.
 > Adding all the files of The COBRA Toolbox ...  Done.
 > Define CB map output... set to svg.
 > Retrieving models ...   Done.
 > TranslateSBML is installed and working properly.
 > Configuring solver environment variables ...
   - [-*--] ILOG_CPLEX_PATH: /Users/sxc554/Applications/IBM/ILOG/CPLEX_Studio1271/cplex/matlab/x86-64_os
   - [*---] GUROBI_PATH: /Library/gurobi700/mac64/matlab
   - [----] TOMLAB_PATH :  --> set this path manually after installing the solver ( see instructions )
   - [-*--] MOSEK_PATH: /Users/sxc554/mosek/7/toolbox/r2013aom
   Done.
 > Checking available solvers and solver interfaces ... Done.
 > Setting default solvers ... Done.
 > Saving the MATLAB path ... Done.
   - The MATLAB path was saved in the default location.

 > Summary of available solvers and solver interfaces

   Support     LP   MILP     QP   MIQP    NLP
 ----------------------------------------------------------------------
 cplex_direct  full              0      0      0      0      -
 dqqMinos      full              1      -      -      -      -
 glpk          full              1      1      -      -      -
 gurobi        full              1      1      1      1      -
 ibm_cplex     full              1      1      1      -      -
 matlab        full              1      -      -      -      1
 mosek         full              1      1      1      -      -
 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         -                 9      4      5      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' - 'mosek' -
```

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

> Checking for available updates ...
> There are 0 new commit(s) on <master> and 32 new commit(s) on <develop> [16260f @ develop]
> You can update The COBRA Toolbox by running updateCobraToolbox() (from within MATLAB).
```

All SteadyCom functions involve only solving linear programming problems. Any solvers supported by the COBRA toolbox will work. But SteadyCom contains specialized codes for IBM ILOG Cplex which was tested to run significantly faster for SteadyComFVA and SteadyComPOA for larger problems through calling the Cplex object in Matlab directly.

```
changeCobraSolver('ibm_cplex', 'LP');
```

PROCEDURE

Model Construction

Load the *E. coli* iAF1260 model in the COBRA toolbox:

```
load('iAF1260.mat', 'iAF1260');
```

Polish the model a little bit:

```
% convert the compartment format from e.g., '_c' to '[c]'
iAF1260.mets = regexprep(iAF1260.mets, '_([^_]+)$', '\[$1\]');
% make all empty cells in cell arrays to be empty string
fieldToBeCellStr = {'metFormulas'; 'genes'; 'grRules'; 'metNames'; 'rxnNames'; 'subSystems'};
for j = 1:numel(fieldToBeCellStr)
    iAF1260.(fieldToBeCellStr{j})(cellfun(@isempty, iAF1260.(fieldToBeCellStr{j}))) = {''};
end
```

Add a methionine export reaction to allow the export of methionine.

```
iAF1260 = addReaction(iAF1260,{'METt3pp',''},'met__L[c] + h[c] => met__L[p] + h[p]');
```

```
METt3pp h[c] + met__L[c]  -> h[p] + met__L[p]
```

Reactions essential for amino acid autotrophy:

```
argH = {'ARGSL'};  % essential for arginine biosynthesis
lysA = {'DAPDC'};  % essential for lysine biosynthesis
metA = {'HSST'};  % essential for methionine biosynthesis
ilvE = {'PPNDH'};  % essential for phenylalanine biosynthesis
```

Reactions essential for exporting amino acids:

```
argO = {'ARGt3pp'};  % Evidence for an arginine exporter encoded by yggA (argO) that is regula
lysO = {'LYSt3pp'};  % Distinct paths for basic amino acid export in Escherichia coli: YbjE (L
yjeH = {'METt3pp'};  % YjeH is a novel L-methionine and branched chain amino acids exporter in
```

```
yddG = {'PHEt2rpp'};  % YddG from Escherichia coli promotes export of aromatic amino acids.
```

Now make four copies of the model with auxotrophy for different amino acids and inability to export amino acids:

```
% auxotrophic for Lys and Met, not exporting Phe
Ec1 = iAF1260;
Ec1 = changeRxnBounds(Ec1, [lysA; metA; yddG], 0, 'b');
% auxotrophic for Arg and Phe, not exporting Met
Ec2 = iAF1260;
Ec2 = changeRxnBounds(Ec2, [argH; yjeH; ilvE], 0, 'b');
% Auxotrophic for Arg and Phe, not exporting Lys
Ec3 = iAF1260;
Ec3 = changeRxnBounds(Ec3, [argH; lysO; ilvE], 0, 'b');
% Auxotrophic for Lys and Met, not exporting Arg
Ec4 = iAF1260;
Ec4 = changeRxnBounds(Ec4, [argO; lysA; metA], 0, 'b');
```

Now none of the four organisms can grow alone and they must cross feed each other to survive. See Figure 1 in ref. [1] for the visualization of the community.

Get the extracellular metabolites, the corresponding exchange reactions and the uptake rates for the *E. coli* model, which are used later to constrain the community model:

```
% extracellular metabolites (met[e])
metEx = strcmp(getCompartment(iAF1260.mets),'e');
% the corresponding exchange reactions
rxnExAll = find(sum(iAF1260.S ~= 0, 1) == 1);
[rxnEx, ~] = find(iAF1260.S(metEx, rxnExAll)');  % need to be in the same order as metEx
rxnEx = rxnExAll(rxnEx);
% exchange rate
lbEx = iAF1260.lb(rxnEx);
```

Create a community model with the four *E. coli* tagged as 'Ec1', 'Ec2', 'Ec3', 'Ec4' respectively by calling `createMultipleSpeciesModel`.

```
nameTagsModel = {'Ec1'; 'Ec2'; 'Ec3'; 'Ec4'};
EcCom = createMultipleSpeciesModel({Ec1; Ec2; Ec3; Ec4}, nameTagsModel);
```

```
The following fields are missing in several models, they will not be merged:
Ec1IEX_12ppd__R[u]tr Ec112ppd__R[e]  <=> 12ppd__R[u]
Ec1IEX_12ppd__S[u]tr Ec112ppd__S[e]  <=> 12ppd__S[u]
Ec1IEX_14glucan[u]tr Ec114glucan[e]  <=> 14glucan[u]
Ec1IEX_15dap[u]tr Ec115dap[e]  <=> 15dap[u]
Ec1IEX_23camp[u]tr Ec123camp[e]  <=> 23camp[u]
Ec1IEX_23ccmp[u]tr Ec123ccmp[e]  <=> 23ccmp[u]
Ec1IEX_23cgmp[u]tr Ec123cgmp[e]  <=> 23cgmp[u]
Ec1IEX_23cump[u]tr Ec123cump[e]  <=> 23cump[u]
Ec1IEX_23dappa[u]tr Ec123dappa[e]  <=> 23dappa[u]
Ec1IEX_26dap__M[u]tr Ec126dap__M[e]  <=> 26dap__M[u]
Ec1IEX_2ddglcn[u]tr Ec12ddglcn[e]  <=> 2ddglcn[u]
Ec1IEX_34dhpac[u]tr Ec134dhpac[e]  <=> 34dhpac[u]
Ec1IEX_3amp[u]tr Ec13amp[e]  <=> 3amp[u]
Ec1IEX_3cmp[u]tr Ec13cmp[e]  <=> 3cmp[u]
Ec1IEX_3gmp[u]tr Ec13gmp[e]  <=> 3gmp[u]
Ec1IEX_3hcinnm[u]tr Ec13hcinnm[e]  <=> 3hcinnm[u]
Ec1IEX_3hpppn[u]tr Ec13hpppn[e]  <=> 3hpppn[u]
Ec1IEX_3ump[u]tr Ec13ump[e]  <=> 3ump[u]
Ec1IEX_4abut[u]tr Ec14abut[e]  <=> 4abut[u]
Ec1IEX_4hoxpacd[u]tr Ec14hoxpacd[e]  <=> 4hoxpacd[u]
```

```
Ec1IEX_5dglcn[u]tr Ec15dglcn[e]  <=> 5dglcn[u]
Ec1IEX_LalaDgluMdapDala[u]tr Ec1LalaDgluMdapDala[e]  <=> LalaDgluMdapDala[u]
Ec1IEX_LalaDgluMdap[u]tr Ec1LalaDgluMdap[e]  <=> LalaDgluMdap[u]
Ec1IEX_ac[u]tr Ec1ac[e]  <=> ac[u]
Ec1IEX_acac[u]tr Ec1acac[e]  <=> acac[u]
Ec1IEX_acald[u]tr Ec1acald[e]  <=> acald[u]
Ec1IEX_acgal1p[u]tr Ec1acgal1p[e]  <=> acgal1p[u]
Ec1IEX_acgal[u]tr Ec1acgal[e]  <=> acgal[u]
Ec1IEX_acgam1p[u]tr Ec1acgam1p[e]  <=> acgam1p[u]
Ec1IEX_acgam[u]tr Ec1acgam[e]  <=> acgam[u]
Ec1IEX_acmana[u]tr Ec1acmana[e]  <=> acmana[u]
Ec1IEX_acmum[u]tr Ec1acmum[e]  <=> acmum[u]
Ec1IEX_acnam[u]tr Ec1acnam[e]  <=> acnam[u]
Ec1IEX_acolipa[u]tr Ec1acolipa[e]  <=> acolipa[u]
Ec1IEX_acser[u]tr Ec1acser[e]  <=> acser[u]
Ec1IEX_ade[u]tr Ec1ade[e]  <=> ade[u]
Ec1IEX_adn[u]tr Ec1adn[e]  <=> adn[u]
Ec1IEX_adocbl[u]tr Ec1adocbl[e]  <=> adocbl[u]
Ec1IEX_ag[u]tr Ec1ag[e]  <=> ag[u]
Ec1IEX_agm[u]tr Ec1agm[e]  <=> agm[u]
Ec1IEX_akg[u]tr Ec1akg[e]  <=> akg[u]
Ec1IEX_ala_B[u]tr Ec1ala_B[e]  <=> ala_B[u]
Ec1IEX_ala__D[u]tr Ec1ala__D[e]  <=> ala__D[u]
Ec1IEX_ala__L[u]tr Ec1ala__L[e]  <=> ala__L[u]
Ec1IEX_alaala[u]tr Ec1alaala[e]  <=> alaala[u]
Ec1IEX_all__D[u]tr Ec1all__D[e]  <=> all__D[u]
Ec1IEX_alltn[u]tr Ec1alltn[e]  <=> alltn[u]
Ec1IEX_amp[u]tr Ec1amp[e]  <=> amp[u]
Ec1IEX_anhgm[u]tr Ec1anhgm[e]  <=> anhgm[u]
Ec1IEX_arab__L[u]tr Ec1arab__L[e]  <=> arab__L[u]
Ec1IEX_arbtn_fe3[u]tr Ec1arbtn_fe3[e]  <=> arbtn_fe3[u]
Ec1IEX_arbtn[u]tr Ec1arbtn[e]  <=> arbtn[u]
Ec1IEX_arg__L[u]tr Ec1arg__L[e]  <=> arg__L[u]
Ec1IEX_ascb__L[u]tr Ec1ascb__L[e]  <=> ascb__L[u]
Ec1IEX_asn__L[u]tr Ec1asn__L[e]  <=> asn__L[u]
Ec1IEX_aso3[u]tr Ec1aso3[e]  <=> aso3[u]
Ec1IEX_asp__L[u]tr Ec1asp__L[e]  <=> asp__L[u]
Ec1IEX_but[u]tr Ec1but[e]  <=> but[u]
Ec1IEX_butso3[u]tr Ec1butso3[e]  <=> butso3[u]
Ec1IEX_ca2[u]tr Ec1ca2[e]  <=> ca2[u]
Ec1IEX_cbi[u]tr Ec1cbi[e]  <=> cbi[u]
Ec1IEX_cbl1[u]tr Ec1cbl1[e]  <=> cbl1[u]
Ec1IEX_cd2[u]tr Ec1cd2[e]  <=> cd2[u]
Ec1IEX_cgly[u]tr Ec1cgly[e]  <=> cgly[u]
Ec1IEX_chol[u]tr Ec1chol[e]  <=> chol[u]
Ec1IEX_cit[u]tr Ec1cit[e]  <=> cit[u]
Ec1IEX_cl[u]tr Ec1cl[e]  <=> cl[u]
Ec1IEX_cmp[u]tr Ec1cmp[e]  <=> cmp[u]
Ec1IEX_co2[u]tr Ec1co2[e]  <=> co2[u]
Ec1IEX_cobalt2[u]tr Ec1cobalt2[e]  <=> cobalt2[u]
Ec1IEX_colipa[u]tr Ec1colipa[e]  <=> colipa[u]
Ec1IEX_cpgn_un[u]tr Ec1cpgn_un[e]  <=> cpgn_un[u]
Ec1IEX_cpgn[u]tr Ec1cpgn[e]  <=> cpgn[u]
Ec1IEX_crn[u]tr Ec1crn[e]  <=> crn[u]
Ec1IEX_csn[u]tr Ec1csn[e]  <=> csn[u]
Ec1IEX_cu2[u]tr Ec1cu2[e]  <=> cu2[u]
Ec1IEX_cu[u]tr Ec1cu[e]  <=> cu[u]
Ec1IEX_cyan[u]tr Ec1cyan[e]  <=> cyan[u]
Ec1IEX_cynt[u]tr Ec1cynt[e]  <=> cynt[u]
Ec1IEX_cys__D[u]tr Ec1cys__D[e]  <=> cys__D[u]
Ec1IEX_cys__L[u]tr Ec1cys__L[e]  <=> cys__L[u]
Ec1IEX_cytd[u]tr Ec1cytd[e]  <=> cytd[u]
Ec1IEX_dad_2[u]tr Ec1dad_2[e]  <=> dad_2[u]
Ec1IEX_damp[u]tr Ec1damp[e]  <=> damp[u]
Ec1IEX_dca[u]tr Ec1dca[e]  <=> dca[u]
Ec1IEX_dcmp[u]tr Ec1dcmp[e]  <=> dcmp[u]
Ec1IEX_dcyt[u]tr Ec1dcyt[e]  <=> dcyt[u]
Ec1IEX_ddca[u]tr Ec1ddca[e]  <=> ddca[u]
```

```
Ec1IEX_dgmp[u]tr Ec1dgmp[e]  <=> dgmp[u]
Ec1IEX_dgsn[u]tr Ec1dgsn[e]  <=> dgsn[u]
Ec1IEX_dha[u]tr Ec1dha[e]  <=> dha[u]
Ec1IEX_dimp[u]tr Ec1dimp[e]  <=> dimp[u]
Ec1IEX_din[u]tr Ec1din[e]  <=> din[u]
Ec1IEX_dms[u]tr Ec1dms[e]  <=> dms[u]
Ec1IEX_dmso[u]tr Ec1dmso[e]  <=> dmso[u]
Ec1IEX_dopa[u]tr Ec1dopa[e]  <=> dopa[u]
Ec1IEX_dtmp[u]tr Ec1dtmp[e]  <=> dtmp[u]
Ec1IEX_dump[u]tr Ec1dump[e]  <=> dump[u]
Ec1IEX_duri[u]tr Ec1duri[e]  <=> duri[u]
Ec1IEX_eca4colipa[u]tr Ec1eca4colipa[e]  <=> eca4colipa[u]
Ec1IEX_enlipa[u]tr Ec1enlipa[e]  <=> enlipa[u]
Ec1IEX_enter[u]tr Ec1enter[e]  <=> enter[u]
Ec1IEX_etha[u]tr Ec1etha[e]  <=> etha[u]
Ec1IEX_ethso3[u]tr Ec1ethso3[e]  <=> ethso3[u]
Ec1IEX_etoh[u]tr Ec1etoh[e]  <=> etoh[u]
Ec1IEX_f6p[u]tr Ec1f6p[e]  <=> f6p[u]
Ec1IEX_fald[u]tr Ec1fald[e]  <=> fald[u]
Ec1IEX_fe2[u]tr Ec1fe2[e]  <=> fe2[u]
Ec1IEX_fe3[u]tr Ec1fe3[e]  <=> fe3[u]
Ec1IEX_fe3dcit[u]tr Ec1fe3dcit[e]  <=> fe3dcit[u]
Ec1IEX_fe3dhbzs[u]tr Ec1fe3dhbzs[e]  <=> fe3dhbzs[u]
Ec1IEX_fe3hox_un[u]tr Ec1fe3hox_un[e]  <=> fe3hox_un[u]
Ec1IEX_fe3hox[u]tr Ec1fe3hox[e]  <=> fe3hox[u]
Ec1IEX_fecrm_un[u]tr Ec1fecrm_un[e]  <=> fecrm_un[u]
Ec1IEX_fecrm[u]tr Ec1fecrm[e]  <=> fecrm[u]
Ec1IEX_feenter[u]tr Ec1feenter[e]  <=> feenter[u]
Ec1IEX_feoxam_un[u]tr Ec1feoxam_un[e]  <=> feoxam_un[u]
Ec1IEX_feoxam[u]tr Ec1feoxam[e]  <=> feoxam[u]
Ec1IEX_for[u]tr Ec1for[e]  <=> for[u]
Ec1IEX_fru[u]tr Ec1fru[e]  <=> fru[u]
Ec1IEX_frulys[u]tr Ec1frulys[e]  <=> frulys[u]
Ec1IEX_fruur[u]tr Ec1fruur[e]  <=> fruur[u]
Ec1IEX_fuc__L[u]tr Ec1fuc__L[e]  <=> fuc__L[u]
Ec1IEX_fum[u]tr Ec1fum[e]  <=> fum[u]
Ec1IEX_g1p[u]tr Ec1g1p[e]  <=> g1p[u]
Ec1IEX_g3pc[u]tr Ec1g3pc[e]  <=> g3pc[u]
Ec1IEX_g3pe[u]tr Ec1g3pe[e]  <=> g3pe[u]
Ec1IEX_g3pg[u]tr Ec1g3pg[e]  <=> g3pg[u]
Ec1IEX_g3pi[u]tr Ec1g3pi[e]  <=> g3pi[u]
Ec1IEX_g3ps[u]tr Ec1g3ps[e]  <=> g3ps[u]
Ec1IEX_g6p[u]tr Ec1g6p[e]  <=> g6p[u]
Ec1IEX_gal1p[u]tr Ec1gal1p[e]  <=> gal1p[u]
Ec1IEX_gal_bD[u]tr Ec1gal_bD[e]  <=> gal_bD[u]
Ec1IEX_gal[u]tr Ec1gal[e]  <=> gal[u]
Ec1IEX_galct__D[u]tr Ec1galct__D[e]  <=> galct__D[u]
Ec1IEX_galctn__D[u]tr Ec1galctn__D[e]  <=> galctn__D[u]
Ec1IEX_galctn__L[u]tr Ec1galctn__L[e]  <=> galctn__L[u]
Ec1IEX_galt[u]tr Ec1galt[e]  <=> galt[u]
Ec1IEX_galur[u]tr Ec1galur[e]  <=> galur[u]
Ec1IEX_gam6p[u]tr Ec1gam6p[e]  <=> gam6p[u]
Ec1IEX_gam[u]tr Ec1gam[e]  <=> gam[u]
Ec1IEX_gbbtn[u]tr Ec1gbbtn[e]  <=> gbbtn[u]
Ec1IEX_gdp[u]tr Ec1gdp[e]  <=> gdp[u]
Ec1IEX_glc__D[u]tr Ec1glc__D[e]  <=> glc__D[u]
Ec1IEX_glcn[u]tr Ec1glcn[e]  <=> glcn[u]
Ec1IEX_glcr[u]tr Ec1glcr[e]  <=> glcr[u]
Ec1IEX_glcur1p[u]tr Ec1glcur1p[e]  <=> glcur1p[u]
Ec1IEX_glcur[u]tr Ec1glcur[e]  <=> glcur[u]
Ec1IEX_gln__L[u]tr Ec1gln__L[e]  <=> gln__L[u]
Ec1IEX_glu__L[u]tr Ec1glu__L[e]  <=> glu__L[u]
Ec1IEX_gly[u]tr Ec1gly[e]  <=> gly[u]
Ec1IEX_glyald[u]tr Ec1glyald[e]  <=> glyald[u]
Ec1IEX_glyb[u]tr Ec1glyb[e]  <=> glyb[u]
Ec1IEX_glyc2p[u]tr Ec1glyc2p[e]  <=> glyc2p[u]
Ec1IEX_glyc3p[u]tr Ec1glyc3p[e]  <=> glyc3p[u]
Ec1IEX_glyc__R[u]tr Ec1glyc__R[e]  <=> glyc__R[u]
```

```
Ec1IEX_glyc[u]tr Ec1glyc[e]  <=> glyc[u]
Ec1IEX_glyclt[u]tr Ec1glyclt[e]  <=> glyclt[u]
Ec1IEX_gmp[u]tr Ec1gmp[e]  <=> gmp[u]
Ec1IEX_gsn[u]tr Ec1gsn[e]  <=> gsn[u]
Ec1IEX_gthox[u]tr Ec1gthox[e]  <=> gthox[u]
Ec1IEX_gthrd[u]tr Ec1gthrd[e]  <=> gthrd[u]
Ec1IEX_gtp[u]tr Ec1gtp[e]  <=> gtp[u]
Ec1IEX_gua[u]tr Ec1gua[e]  <=> gua[u]
Ec1IEX_h2[u]tr Ec1h2[e]  <=> h2[u]
Ec1IEX_h2o2[u]tr Ec1h2o2[e]  <=> h2o2[u]
Ec1IEX_h2o[u]tr Ec1h2o[e]  <=> h2o[u]
Ec1IEX_h2s[u]tr Ec1h2s[e]  <=> h2s[u]
Ec1IEX_h[u]tr Ec1h[e]  <=> h[u]
Ec1IEX_hacolipa[u]tr Ec1hacolipa[e]  <=> hacolipa[u]
Ec1IEX_halipa[u]tr Ec1halipa[e]  <=> halipa[u]
Ec1IEX_hdca[u]tr Ec1hdca[e]  <=> hdca[u]
Ec1IEX_hdcea[u]tr Ec1hdcea[e]  <=> hdcea[u]
Ec1IEX_hg2[u]tr Ec1hg2[e]  <=> hg2[u]
Ec1IEX_his__L[u]tr Ec1his__L[e]  <=> his__L[u]
Ec1IEX_hom__L[u]tr Ec1hom__L[e]  <=> hom__L[u]
Ec1IEX_hxa[u]tr Ec1hxa[e]  <=> hxa[u]
Ec1IEX_hxan[u]tr Ec1hxan[e]  <=> hxan[u]
Ec1IEX_idon__L[u]tr Ec1idon__L[e]  <=> idon__L[u]
Ec1IEX_ile__L[u]tr Ec1ile__L[e]  <=> ile__L[u]
Ec1IEX_imp[u]tr Ec1imp[e]  <=> imp[u]
Ec1IEX_indole[u]tr Ec1indole[e]  <=> indole[u]
Ec1IEX_inost[u]tr Ec1inost[e]  <=> inost[u]
Ec1IEX_ins[u]tr Ec1ins[e]  <=> ins[u]
Ec1IEX_isetac[u]tr Ec1isetac[e]  <=> isetac[u]
Ec1IEX_k[u]tr Ec1k[e]  <=> k[u]
Ec1IEX_kdo2lipid4[u]tr Ec1kdo2lipid4[e]  <=> kdo2lipid4[u]
Ec1IEX_lac__D[u]tr Ec1lac__D[e]  <=> lac__D[u]
Ec1IEX_lac__L[u]tr Ec1lac__L[e]  <=> lac__L[u]
Ec1IEX_lcts[u]tr Ec1lcts[e]  <=> lcts[u]
Ec1IEX_leu__L[u]tr Ec1leu__L[e]  <=> leu__L[u]
Ec1IEX_lipa_cold[u]tr Ec1lipa_cold[e]  <=> lipa_cold[u]
Ec1IEX_lipa[u]tr Ec1lipa[e]  <=> lipa[u]
Ec1IEX_lys__L[u]tr Ec1lys__L[e]  <=> lys__L[u]
Ec1IEX_lyx__L[u]tr Ec1lyx__L[e]  <=> lyx__L[u]
Ec1IEX_mal__D[u]tr Ec1mal__D[e]  <=> mal__D[u]
Ec1IEX_mal__L[u]tr Ec1mal__L[e]  <=> mal__L[u]
Ec1IEX_malt[u]tr Ec1malt[e]  <=> malt[u]
Ec1IEX_malthx[u]tr Ec1malthx[e]  <=> malthx[u]
Ec1IEX_maltpt[u]tr Ec1maltpt[e]  <=> maltpt[u]
Ec1IEX_malttr[u]tr Ec1malttr[e]  <=> malttr[u]
Ec1IEX_maltttr[u]tr Ec1maltttr[e]  <=> maltttr[u]
Ec1IEX_man6p[u]tr Ec1man6p[e]  <=> man6p[u]
Ec1IEX_man[u]tr Ec1man[e]  <=> man[u]
Ec1IEX_manglyc[u]tr Ec1manglyc[e]  <=> manglyc[u]
Ec1IEX_melib[u]tr Ec1melib[e]  <=> melib[u]
Ec1IEX_met__D[u]tr Ec1met__D[e]  <=> met__D[u]
Ec1IEX_met__L[u]tr Ec1met__L[e]  <=> met__L[u]
Ec1IEX_metsox_R__L[u]tr Ec1metsox_R__L[e]  <=> metsox_R__L[u]
Ec1IEX_metsox_S__L[u]tr Ec1metsox_S__L[e]  <=> metsox_S__L[u]
Ec1IEX_mg2[u]tr Ec1mg2[e]  <=> mg2[u]
Ec1IEX_minohp[u]tr Ec1minohp[e]  <=> minohp[u]
Ec1IEX_mmet[u]tr Ec1mmet[e]  <=> mmet[u]
Ec1IEX_mn2[u]tr Ec1mn2[e]  <=> mn2[u]
Ec1IEX_mnl[u]tr Ec1mnl[e]  <=> mnl[u]
Ec1IEX_mobd[u]tr Ec1mobd[e]  <=> mobd[u]
Ec1IEX_mso3[u]tr Ec1mso3[e]  <=> mso3[u]
Ec1IEX_n2o[u]tr Ec1n2o[e]  <=> n2o[u]
Ec1IEX_na1[u]tr Ec1na1[e]  <=> na1[u]
```

```
EcCom.csense = char('E' * ones(1,numel(EcCom.mets)));  % correct the csense
clear Ec1 Ec2 Ec3 Ec4
```

The model `EcCom` contains a community compartment denoted by `[u]` to allow exchange between organisms. Each organism-specific reaction/metabolite is prepended with the corresponding tag.

Retreive the names and ids for organism/community exchange reactions/metabolites which are necessary for computation:

```
[EcCom.infoCom, EcCom.indCom] = getMultiSpeciesModelId(EcCom, nameTagsModel);
disp(EcCom.infoCom);
```

```
    spAbbr: {4×1 cell}
    spName: {4×1 cell}
     EXcom: {299×1 cell}
    EXhost: {0×1 cell}
      EXsp: {299×4 cell}
      Mcom: {299×1 cell}
     Mhost: {0×1 cell}
       Msp: {299×4 cell}
    rxnSps: {9831×1 cell}
    metSps: {6971×1 cell}
```

`EcCom.infoCom` contains reaction/metabolite names (from `EcCom.rxns`/`EcCom.mets`) for the community exchange reactions (`*.EXcom`), organism-community exchange reactions (`*.EXsp`), community metabolites (`*.Mcom`), organism-specific extracellular metabolite (`*.Msp`). If a host model is specified, there will also be non-empty `*.EXhost` and `*.Mhost` for the host-specific exchange reactions and metabolites. The fields `*.rxnSps`/`*.metSps` give information on which organism a reaction/metabolite belongs to.

`indCom` has the same structure as `infoCom` but contains the indices rather than names. `infoCom` and `indCom` are attached as fields of the model `EcCom` because SteadyCom requires this information from the input model for computation. Incorporate also the names and indices for the biomass reactions which are necessary for computing growth:

```
rxnBiomass = strcat(nameTagsModel, 'BIOMASS_Ec_iAF1260_core_59p81M');  % biomass reaction name
rxnBiomassId = findRxnIDs(EcCom, rxnBiomass);  % ids
EcCom.infoCom.spBm = rxnBiomass;  % .spBm for organism biomass reactions
EcCom.indCom.spBm = rxnBiomassId;
```

Finding Maximum Growth Rate Using SteadyCom

Set community and organism-specific uptake rates to be the same as in the orginal iAF1260 model:

```
[yn, id] = ismember(strrep(iAF1260.mets(metEx), '[e]', '[u]'), EcCom.infoCom.Mcom);  % map the
assert(all(yn));  % must be a 1-to-1 mapping
EcCom.lb(EcCom.indCom.EXcom(:,1)) = lbEx(id);  % assign community uptake bounds
EcCom.ub(EcCom.indCom.EXcom(:,1)) = 1e5;
EcCom.lb(EcCom.indCom.EXsp) = repmat(lbEx(id), 1, 4);  % assign organism-specific uptake bound
```

Set maximum allowed organism-specific uptake rates for the cross-feeding amino acids:

```
% only allow to take up the amino acids that one is auxotrophic for
exRate = 1;  % maximum uptake rate for cross feeding AAs
% Ec1
EcCom = changeRxnBounds(EcCom, {'Ec1IEX_arg__L[u]tr'; 'Ec1IEX_phe__L[u]tr'}, 0, 'l');
EcCom = changeRxnBounds(EcCom, {'Ec1IEX_met__L[u]tr'; 'Ec1IEX_lys__L[u]tr'}, -exRate, 'l');
```

```
% Ec2
EcCom = changeRxnBounds(EcCom, {'Ec2IEX_arg__L[u]tr'; 'Ec2IEX_phe__L[u]tr'}, -exRate, 'l');
EcCom = changeRxnBounds(EcCom, {'Ec2IEX_met__L[u]tr'; 'Ec2IEX_lys__L[u]tr'}, 0, 'l');
% Ec3
EcCom = changeRxnBounds(EcCom, {'Ec3IEX_arg__L[u]tr'; 'Ec3IEX_phe__L[u]tr'}, -exRate, 'l');
EcCom = changeRxnBounds(EcCom, {'Ec3IEX_met__L[u]tr'; 'Ec3IEX_lys__L[u]tr'}, 0, 'l');
% Ec4
EcCom = changeRxnBounds(EcCom, {'Ec4IEX_arg__L[u]tr'; 'Ec4IEX_phe__L[u]tr'}, 0, 'l');
EcCom = changeRxnBounds(EcCom, {'Ec4IEX_met__L[u]tr'; 'Ec4IEX_lys__L[u]tr'}, -exRate, 'l');
% allow production of anything for each member
EcCom.ub(EcCom.indCom.EXsp(:)) = 1000;
```

Before the calculation, print the community uptake bounds for checking using `printUptakeBoundCom`:

```
printUptakeBoundCom(EcCom, 1);
```

```
         Mets Comm.     Ec1       Ec2       Ec3       Ec4
 ( 53) arg__L 0         0         -1        -1        0
    ( 60) ca2 1e+06     -1e+06    -1e+06    -1e+06    -1e+06
   ( 62) cbl1 0.01      -0.01     -0.01     -0.01     -0.01
     ( 67) cl 1e+06     -1e+06    -1e+06    -1e+06    -1e+06
    ( 69) co2 1e+06     -1e+06    -1e+06    -1e+06    -1e+06
( 70) cobalt2 1e+06     -1e+06    -1e+06    -1e+06    -1e+06
    ( 76) cu2 1e+06     -1e+06    -1e+06    -1e+06    -1e+06
    (108) fe2 1e+06     -1e+06    -1e+06    -1e+06    -1e+06
    (109) fe3 1e+06     -1e+06    -1e+06    -1e+06    -1e+06
 (144) glc__D 8         -8        -8        -8        -8
    (167) h2o 1e+06     -1e+06    -1e+06    -1e+06    -1e+06
      (169) h 1e+06     -1e+06    -1e+06    -1e+06    -1e+06
      (186) k 1e+06     -1e+06    -1e+06    -1e+06    -1e+06
 (194) lys__L 0         -1        0         0         -1
 (208) met__L 0         -1        0         0         -1
    (211) mg2 1e+06     -1e+06    -1e+06    -1e+06    -1e+06
    (214) mn2 1e+06     -1e+06    -1e+06    -1e+06    -1e+06
   (216) mobd 1e+06     -1e+06    -1e+06    -1e+06    -1e+06
    (219) na1 1e+06     -1e+06    -1e+06    -1e+06    -1e+06
    (221) nh4 1e+06     -1e+06    -1e+06    -1e+06    -1e+06
     (228) o2 18.5      -18.5     -18.5     -18.5     -18.5
 (237) phe__L 0         0         -1        -1        0
     (239) pi 1e+06     -1e+06    -1e+06    -1e+06    -1e+06
    (260) so4 1e+06     -1e+06    -1e+06    -1e+06    -1e+06
  (280) tungs 1e+06     -1e+06    -1e+06    -1e+06    -1e+06
    (299) zn2 1e+06     -1e+06    -1e+06    -1e+06    -1e+06
```

Values under 'Comm.' are the community uptake bounds (+ve for uptake) and values under 'Ec1' are the Ec1-specific uptake bounds (-ve for uptake).

Create an option structure for calling SteadyCom and call the function. There are a range of options available, including setting algorithmic parameters, fixing growth rates for members, adding additional linear constraints in a general format, e.g., for molecular crowding effect. See `help SteadyCom` for more options.

```
options = struct();
options.GRguess = 0.5;  % initial guess for max. growth rate
options.GRtol = 1e-6;  % tolerance for final growth rate
options.algorithm = 1;  % use the default algorithm (simple guessing for bounds, followed by m
[sol, result] = SteadyCom(EcCom, options);
```

```
Find maximum community growth rate..
Model feasible at maintenance. Time elapsed: 1 / 1 sec
Iter        LB    To test        UB  Time elapsed (iteration/total)
```

```
   1  0.000000  0.500000       Inf  0 / 1 sec
   2  0.500000  0.721279       Inf  6 / 7 sec
   3  0.721279  0.735372       Inf  0 / 8 sec
   4  0.735372  0.742726       Inf  0 / 8 sec

 Func-count    x          f(x)             Procedure
    2        0.735372  -0.000807615        initial
    3        0.735378   -0.00079987        interpolation
    4         0.73599  -1.26398e-06        interpolation
    5         0.73599  -1.26398e-06        interpolation

Zero found in the interval [0.735372, 0.742726]
Maximum community growth rate: 0.735990 (abs. error < 1e-06). Time elapsed: 18 sec
```

The algorithm is an iterative procedure to find the maximum biomass at a given growth rate and to determine the maximum growth rate that is feasible for the required total biomass (default 1 gdw). Here the algorithm used is the simple guessing for find upper and lower bounds (Iter 1 to 4 in the output) followed by Matlab `fzero` (starting from the line '`Func-count`') to locate the root. The maximum growth rate calculated is 0.73599 /h, stored in `result.GRmax`.

The biomass for each organism (in gdw) is given by `result.BM`:

```
for jSp = 1:4
    fprintf('X_%s:  %.6f\n', EcCom.infoCom.spAbbr{jSp}, result.BM(jSp));
end
```

```
X_Ec1:  0.253294
X_Ec2:  0.324611
X_Ec3:  0.185004
X_Ec4:  0.237093
```

```
disp(result);
```

```
    GRmax: 0.7360
      vBM: [4×1 double]
       BM: [4×1 double]
       Ut: [299×1 double]
       Ex: [299×1 double]
     flux: [9831×1 double]
    iter0: [0 11.4198 0 9.9476e-14]
     iter: [4×6 double]
     stat: 'optimal'
```

`result.vBM` contains the biomass production rates (in gdw / h), equal to `result.BM * result.GRmax`. Since the total community biomass is defaulted to be 1 gdw, the biomass for each organism coincides with its relative abundance. Note that the community uptake bounds in this sense are normalized per gdw of the community biomass. So the lower bound for the exchange reaction `EX_glc__D[u]` being 8 can be interpreted as the maximum amount of glucose available to the community being at a rate of 8 mmol per hour for 1 gdw of community biomass. Similarly, all fluxes in `result.flux` (V_j^k) has the unit mmol / h / [gdw of comm. biomass]. It differs from the specific rate (traditionally denoted by v_j^k) of an organism in the usual sense (in the unit of mmol / h / [gdw of organism biomass]) by $V_j^k = X^k v_j^k$ where X^k is the biomass of the organism. `result.Ut` and `result.Ex` are the community uptake and export rates respectively, corresponding to the exchange reactions in `EcCom.infoCom.EXcom`.

`result.iter0` is the info for solving the model at zero growth rate and `result.iter` records the info during iteration of the algorithm:

```
iter = [0, result.iter0, NaN; result.iter];
for j = 0 : size(iter, 1)
    if j == 0
        fprintf('#iter\tgrowth rate (mu)\tmax. biomass (sum(X))\tmu * sum(X)\tmax. infeasibili
    else
        fprintf('%5d\t%16.6f\t%21.6f\t%11.6f\t%18.6e\t%d\n', iter(j,:))
    end
end
```

```
#iter growth rate (mu) max. biomass (sum(X)) mu * sum(X) max. infeasibility guess method
    0         0.000000             11.419845    0.000000       9.947598e-14 NaN
    1         0.500000              1.442559    0.721279       5.555648e-10 0
    2         0.721279              1.019539    0.735372       3.778331e-10 0
    3         0.735372              1.000808    0.735966       7.999141e-11 0
    4         0.742726              0.000000    0.000000       0.000000e+00 2
```

`mu * sum(X)` in the forth column is equal to the biomass production rate.

The fifth column contains the maximum infeasibility of the solutions in each iteration.

Guess method in the last column represents the method used for guessing the growth rate solved in the current iteration:

0: the default simple guess by $\mu_{\text{next}} = \mu_{\text{current}} \sum_{k=1}^{K} X_k^{\text{current}}$ (K is the total number of organisms)

1: bisection method

2: bisection or at least 1% away from the bounds if the simple guess is too close to the bounds (<1%)

3. 1% away from the current growth rate if the simple guess is too close to the current growth rate

From the table, we can see that at the growth rate 0.742726 (iter 4), the max. biomass is 0, while at growth rate 0.735372, max. biomass = 1.0008 > 1. Therefore we have both an lower and upper bound for the max. growth rate. Then fzero is initiated to solve for the max. growth rate that gives max. biomass >= 1.

Two other algorithms for the iterative procedure are also implemented: simple guessing only and the bisection method. Compare their results with simple guessing + matlab fzero run above:

```
options.algorithm = 2;  % use the simple guessing algorithm
[sol2, result2] = SteadyCom(EcCom, options);
```

```
Find maximum community growth rate..
Model feasible at maintenance. Time elapsed: 1 / 1 sec
Iter        LB    To test        UB   Time elapsed (iteration/total)
   1  0.000000  0.500000       Inf   0 / 1 sec
   2  0.500000  0.721279       Inf   5 / 5 sec
   3  0.721279  0.735372       Inf   0 / 6 sec
   4  0.735372  0.742726       Inf   0 / 6 sec
   5  0.735372  0.739049  0.742726   0 / 6 sec
   6  0.735372  0.737211  0.739049   0 / 7 sec
   7  0.735372  0.736291  0.737211   0 / 7 sec
   8  0.735372  0.735832  0.736291   0 / 7 sec
   9  0.735832  0.736062  0.736291   2 / 9 sec
  10  0.735832  0.735947  0.736062   0 / 9 sec
```

```
  11  0.735947  0.736004  0.736062  2 / 12 sec
  12  0.735947  0.735975  0.736004  0 / 12 sec
  13  0.735975  0.735990  0.736004  3 / 15 sec
  14  0.735990  0.735997  0.736004  0 / 15 sec
  15  0.735990  0.735993  0.735997  0 / 15 sec
  16  0.735990  0.735991  0.735993  1 / 16 sec
  17  0.735990  0.735991  0.735991  1 / 17 sec
Maximum community growth rate: 0.735991 (abs. error < 1e-06). Time elapsed: 35 sec
```

```
options.algorithm = 3;  % use the simple guessing algorithm
[sol3, result3] = SteadyCom(EcCom, options);
```

```
Find maximum community growth rate..
Model feasible at maintenance. Time elapsed: 1 / 1 sec
Iter        LB   To test        UB  Time elapsed (iteration/total)
   1  0.000000  0.500000       Inf  0 / 1 sec
   2  0.500000  1.000000       Inf  5 / 5 sec
   3  0.500000  0.750000  1.000000  1 / 7 sec
   4  0.500000  0.625000  0.750000  14 / 20 sec
   5  0.625000  0.687500  0.750000  12 / 32 sec
   6  0.687500  0.718750  0.750000  0 / 33 sec
   7  0.718750  0.734375  0.750000  0 / 33 sec
   8  0.734375  0.742188  0.750000  1 / 33 sec
   9  0.734375  0.738281  0.742188  0 / 34 sec
  10  0.734375  0.736328  0.738281  1 / 35 sec
  11  0.734375  0.735352  0.736328  1 / 35 sec
  12  0.735352  0.735840  0.736328  1 / 36 sec
  13  0.735840  0.736084  0.736328  1 / 37 sec
  14  0.735840  0.735962  0.736084  0 / 37 sec
  15  0.735962  0.736023  0.736084  2 / 40 sec
  16  0.735962  0.735992  0.736023  0 / 40 sec
  17  0.735962  0.735977  0.735992  1 / 41 sec
  18  0.735977  0.735985  0.735992  4 / 44 sec
  19  0.735985  0.735989  0.735992  0 / 45 sec
  20  0.735989  0.735991  0.735992  1 / 45 sec
  21  0.735991  0.735991  0.735992  0 / 46 sec
Maximum community growth rate: 0.735991 (abs. error < 1e-06). Time elapsed: 70 sec
```

The time used for each algorithm is:

(1) simple guess for bounds followed by Matlab fzero: 18 sec

(2) simple guess alone: 35 sec

(3) bisection: 70 sec

Algorithm (1) appears to be the fastest in most case although the simple guess algorithm can sometimes also outperform it. The most conservative bisection method can already guarantee convergence within around 20 iterations, i.e., solving ~20 LPs for an optimality gap (`options.GRtol`) of 1e-6.

Analyzing Flux Variability Using SteadyComFVA

Now we want to analyze the variability of the organism abundance at various growth rates. Choose more options and call `SteadyComFVA`:

```
% percentage of maximum total biomass of the community required. 100 for sum(biomass) = 1 (1 i
options.optBMpercent = 100;
n = size(EcCom.S, 2);  % number of reactions in the model
```

```
% options.rxnNameList is the list of reactions subject to FVA. Can be reaction names or indice
% Use n + j for the biomass variable of the j-th organism. Alternatively, use {'X_j'}
% for biomass variable of the j-th organism or {'X_Ec1'} for Ec1 (the abbreviation in EcCom.ir
options.rxnNameList = {'X_Ec1'; 'X_Ec2'; 'X_Ec3'; 'X_Ec4'};
options.optGRpercent = [89:0.2:99, 99.1:0.1:100];  % perform FVA at various percentages of the
[fvaComMin,fvaComMax] = SteadyComFVA(EcCom, options);
```

```
Find maximum community growth rate..
Model feasible at maintenance. Time elapsed: 1 / 1 sec
Iter        LB   To test        UB  Time elapsed (iteration/total)
   1  0.000000  0.500000       Inf  0 / 1 sec
   2  0.500000  0.721279       Inf  6 / 7 sec
   3  0.721279  0.735372       Inf  0 / 7 sec
   4  0.735372  0.742726       Inf  0 / 8 sec
   5  0.735372  0.739049  0.742726  0 / 8 sec
   6  0.735372  0.737211  0.739049  0 / 8 sec
   7  0.735372  0.736291  0.737211  0 / 8 sec
   8  0.735372  0.735832  0.736291  0 / 9 sec
   9  0.735832  0.736062  0.736291  2 / 10 sec
  10  0.735832  0.735947  0.736062  0 / 11 sec
  11  0.735947  0.736004  0.736062  3 / 13 sec
  12  0.735947  0.735975  0.736004  0 / 14 sec
  13  0.735975  0.735990  0.736004  3 / 17 sec
  14  0.735990  0.735997  0.736004  0 / 17 sec
  15  0.735990  0.735993  0.735997  0 / 18 sec
  16  0.735990  0.735991  0.735993  1 / 19 sec
  17  0.735990  0.735991  0.735991  1 / 19 sec
Maximum community growth rate: 0.735991 (abs. error < 1e-06). Time elapsed: 42 sec

FVA for 4 sets of fluxes/biomass at growth rate 0.655032 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.044053  0.787577
   2   50      X_Ec2  0.038253  0.720492
   3   75      X_Ec3  0.021200  0.696956
   4  100      X_Ec4  0.029222       NaN

FVA for 4 sets of fluxes/biomass at growth rate 0.656504 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.045103  0.785490
   2   50      X_Ec2  0.039074  0.717227
   3   75      X_Ec3  0.021640  0.693122
   4  100      X_Ec4  0.029851       NaN

FVA for 4 sets of fluxes/biomass at growth rate 0.657976 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.046186  0.783368
   2   50      X_Ec2  0.039919  0.713899
   3   75      X_Ec3  0.022092  0.689206
   4  100      X_Ec4  0.030498  0.689833

FVA for 4 sets of fluxes/biomass at growth rate 0.659448 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.047304  0.781210
   2   50      X_Ec2  0.040788  0.710505
   3   75      X_Ec3  0.022556  0.685205
   4  100      X_Ec4  0.031163  0.686016

FVA for 4 sets of fluxes/biomass at growth rate 0.660920 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.048458  0.779015
   2   50      X_Ec2  0.041682  0.707043
   3   75      X_Ec3  0.023033  0.681116
   4  100      X_Ec4  0.031848  0.682120

FVA for 4 sets of fluxes/biomass at growth rate 0.662392 :
  No    %       Name       Min       Max
```

```
    1   25      X_Ec1  0.049649  0.776783
    2   50      X_Ec2  0.042603  0.703511
    3   75      X_Ec3  0.023523  0.676937
    4  100      X_Ec4  0.032553  0.678141

FVA for 4 sets of fluxes/biomass at growth rate 0.663864 :
  No    %       Name       Min       Max
    1   25      X_Ec1  0.050880  0.774508
    2   50      X_Ec2  0.043552  0.699897
    3   75      X_Ec3  0.024028  0.672653
    4  100      X_Ec4  0.033283  0.674078

FVA for 4 sets of fluxes/biomass at growth rate 0.665335 :
  No    %       Name       Min       Max
    1   25      X_Ec1  0.052152  0.772192
    2   50      X_Ec2  0.044530  0.696202
    3   75      X_Ec3  0.024547  0.668265
    4  100      X_Ec4  0.034036  0.669927

FVA for 4 sets of fluxes/biomass at growth rate 0.666807 :
  No    %       Name       Min       Max
    1   25      X_Ec1  0.053466  0.769834
    2   50      X_Ec2  0.045538  0.692430
    3   75      X_Ec3  0.025082  0.663775
    4  100      X_Ec4  0.034812       NaN

FVA for 4 sets of fluxes/biomass at growth rate 0.668279 :
  No    %       Name       Min       Max
    1   25      X_Ec1  0.054825  0.767433
    2   50      X_Ec2  0.046576  0.688579
    3   75      X_Ec3  0.025631  0.659181
    4  100      X_Ec4  0.035612  0.661351

FVA for 4 sets of fluxes/biomass at growth rate 0.669751 :
  No    %       Name       Min       Max
    1   25      X_Ec1  0.056231  0.764987
    2   50      X_Ec2  0.047646  0.684644
    3   75      X_Ec3  0.026197  0.654478
    4  100      X_Ec4  0.036437       NaN

FVA for 4 sets of fluxes/biomass at growth rate 0.671223 :
  No    %       Name       Min       Max
    1   25      X_Ec1  0.057686  0.762497
    2   50      X_Ec2  0.048750  0.680624
    3   75      X_Ec3  0.026779  0.649662
    4  100      X_Ec4  0.037288  0.652387

FVA for 4 sets of fluxes/biomass at growth rate 0.672695 :
  No    %       Name       Min       Max
    1   25      X_Ec1  0.059191  0.759959
    2   50      X_Ec2  0.049888  0.676516
    3   75      X_Ec3  0.027379  0.644730
    4  100      X_Ec4  0.038166  0.647751

FVA for 4 sets of fluxes/biomass at growth rate 0.674167 :
  No    %       Name       Min       Max
    1   25      X_Ec1  0.060750  0.757372
    2   50      X_Ec2  0.051063  0.672316
    3   75      X_Ec3  0.027996  0.639676
    4  100      X_Ec4  0.039073       NaN

FVA for 4 sets of fluxes/biomass at growth rate 0.675639 :
  No    %       Name       Min       Max
    1   25      X_Ec1  0.062365  0.754735
    2   50      X_Ec2  0.052275  0.668021
    3   75      X_Ec3  0.028632  0.634496
    4  100      X_Ec4  0.040009  0.638153
```

```
FVA for 4 sets of fluxes/biomass at growth rate 0.677111 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.064038  0.752047
   2   50      X_Ec2  0.053526  0.663629
   3   75      X_Ec3  0.029287  0.629185
   4  100      X_Ec4  0.040976  0.633182

FVA for 4 sets of fluxes/biomass at growth rate 0.678583 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.065772  0.749305
   2   50      X_Ec2  0.054818  0.659135
   3   75      X_Ec3  0.029963  0.623738
   4  100      X_Ec4  0.041975  0.628092

FVA for 4 sets of fluxes/biomass at growth rate 0.680055 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.067571  0.746507
   2   50      X_Ec2  0.056153  0.654536
   3   75      X_Ec3  0.030659  0.618150
   4  100      X_Ec4  0.043007  0.622877

FVA for 4 sets of fluxes/biomass at growth rate 0.681527 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.069437  0.743652
   2   50      X_Ec2  0.057533  0.649827
   3   75      X_Ec3  0.031377  0.612415
   4  100      X_Ec4  0.044075  0.617533

FVA for 4 sets of fluxes/biomass at growth rate 0.682999 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.071373  0.740737
   2   50      X_Ec2  0.058959  0.645005
   3   75      X_Ec3  0.032118  0.606526
   4  100      X_Ec4  0.045179       NaN

FVA for 4 sets of fluxes/biomass at growth rate 0.684471 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.073384  0.737761
   2   50      X_Ec2  0.060434  0.640066
   3   75      X_Ec3  0.032883  0.600478
   4  100      X_Ec4  0.046322  0.606437

FVA for 4 sets of fluxes/biomass at growth rate 0.685943 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.075473  0.734721
   2   50      X_Ec2  0.061960  0.635005
   3   75      X_Ec3  0.033672  0.594264
   4  100      X_Ec4  0.047505  0.600674

FVA for 4 sets of fluxes/biomass at growth rate 0.687415 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.077644  0.731615
   2   50      X_Ec2  0.063539  0.629817
   3   75      X_Ec3  0.034486  0.587876
   4  100      X_Ec4  0.048731  0.594760

FVA for 4 sets of fluxes/biomass at growth rate 0.688887 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.079901  0.728440
   2   50      X_Ec2  0.065175  0.624497
   3   75      X_Ec3  0.035328  0.581307
   4  100      X_Ec4  0.050000  0.588689

FVA for 4 sets of fluxes/biomass at growth rate 0.690359 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.082249  0.725193
   2   50      X_Ec2  0.066868  0.619039
   3   75      X_Ec3  0.036197  0.574550
```

```
   4  100      X_Ec4  0.051316  0.582454

FVA for 4 sets of fluxes/biomass at growth rate 0.691831 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.084697  0.721873
   2   50      X_Ec2  0.068624  0.613425
   3   75      X_Ec3  0.037096  0.567594
   4  100      X_Ec4  0.052681  0.576024

FVA for 4 sets of fluxes/biomass at growth rate 0.693303 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.087248  0.718475
   2   50      X_Ec2  0.070444  0.607659
   3   75      X_Ec3  0.038025  0.560432
   4  100      X_Ec4  0.054097  0.569413

FVA for 4 sets of fluxes/biomass at growth rate 0.694775 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.089906  0.714997
   2   50      X_Ec2  0.072331  0.601736
   3   75      X_Ec3  0.038986  0.553054
   4  100      X_Ec4  0.055567  0.562614

FVA for 4 sets of fluxes/biomass at growth rate 0.696247 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.092676  0.711435
   2   50      X_Ec2  0.074290  0.595650
   3   75      X_Ec3  0.039980  0.545450
   4  100      X_Ec4  0.057093  0.555620

FVA for 4 sets of fluxes/biomass at growth rate 0.697719 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.095566  0.707785
   2   50      X_Ec2  0.076323  0.589407
   3   75      X_Ec3  0.041009  0.537608
   4  100      X_Ec4  0.058679  0.548420

FVA for 4 sets of fluxes/biomass at growth rate 0.699191 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.098582  0.704045
   2   50      X_Ec2  0.078435  0.583010
   3   75      X_Ec3  0.042075  0.529518
   4  100      X_Ec4  0.060328  0.541006

FVA for 4 sets of fluxes/biomass at growth rate 0.700663 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.101732  0.700210
   2   50      X_Ec2  0.080630  0.576441
   3   75      X_Ec3  0.043179  0.521166
   4  100      X_Ec4  0.062043  0.533368

FVA for 4 sets of fluxes/biomass at growth rate 0.702135 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.105024  0.696275
   2   50      X_Ec2  0.082912  0.569710
   3   75      X_Ec3  0.044323  0.512540
   4  100      X_Ec4  0.063828  0.525494

FVA for 4 sets of fluxes/biomass at growth rate 0.703607 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.108465  0.692237
   2   50      X_Ec2  0.085286  0.562859
```

Similar to the output by `fluxVariability`, `fvaComMin` contains the minimum fluxes corresponding to the reactions in `options.rxnNameList`. `fvaComMax` contains the maximum fluxes.

options.rxnNameList can be supplied as a (#rxns + #organism)-by-K matrix to analyze the variability of the K linear combinations of flux/biomass variables in the columns of the matrix. See `help SteadyComFVA` for more details.

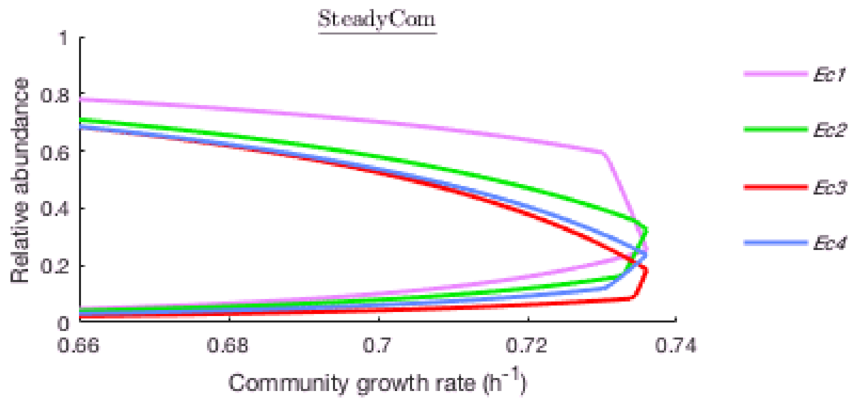
We would also like to compare the results against the direct use of FBA and FVA by calling `optimizeCbModel` and `fluxVariability`:

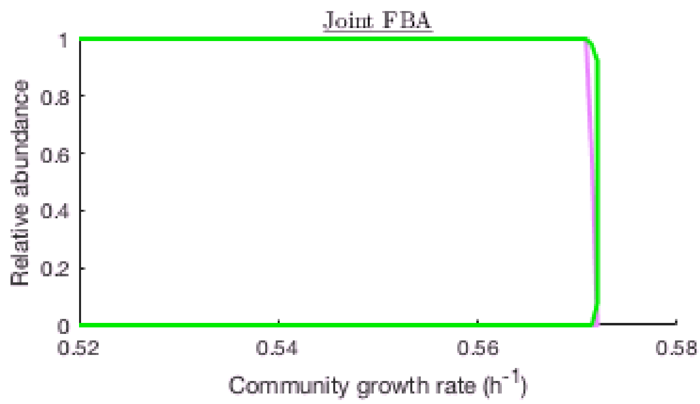
```
optGRpercentFBA = [89:2:99 99.1:0.1:100];  % less dense interval to save time because the resu
nGr = numel(optGRpercentFBA);
[fvaFBAMin, fvaFBAMax] = deal(zeros(numel(options.rxnNameList), nGr));
% change the objective function to the sum of all biomass reactions
EcCom.c(:) = 0;
EcCom.c(EcCom.indCom.spBm) = 1;
EcCom.csense = char('E' * ones(1, numel(EcCom.mets)));
s = optimizeCbModel(EcCom);  % run FBA
grFBA = s.f;
for jGr = 1:nGr
    fprintf('Growth rate %.4f :\n', grFBA * optGRpercentFBA(jGr)/100);
    [fvaFBAMin(:, jGr), fvaFBAMax(:, jGr)] = fluxVariability(EcCom, optGRpercentFBA(jGr), 'max
end
```

```
Growth rate 0.5091 :
  No Perc       Name       Min       Max
Growth rate 0.5205 :
  No Perc       Name       Min       Max
Growth rate 0.5319 :
  No Perc       Name       Min       Max
Growth rate 0.5434 :
  No Perc       Name       Min       Max
Growth rate 0.5548 :
  No Perc       Name       Min       Max
Growth rate 0.5663 :
  No Perc       Name       Min       Max
Growth rate 0.5668 :
  No Perc       Name       Min       Max
Growth rate 0.5674 :
  No Perc       Name       Min       Max
Growth rate 0.5680 :
  No Perc       Name       Min       Max
Growth rate 0.5686 :
  No Perc       Name       Min       Max
Growth rate 0.5691 :
  No Perc       Name       Min       Max
Growth rate 0.5697 :
  No Perc       Name       Min       Max
Growth rate 0.5703 :
  No Perc       Name       Min       Max
Growth rate 0.5708 :
  No Perc       Name       Min       Max
Growth rate 0.5714 :
  No Perc       Name       Min       Max
Growth rate 0.5720 :
  No Perc       Name       Min       Max
```

Plot the results to visualize the difference (see also Figure 2 in ref. [1]):

```
grComV = result.GRmax * options.optGRpercent / 100;  % vector of growth rates tested
lgLabel = {'{\itEc1 }';'{\itEc2 }';'{\itEc3 }';'{\itEc4 }'};
col = [235 135 255; 0 235 0; 255 0 0; 95 135 255 ]/255;  % color
f = figure;
% SteadyCom
subplot(2, 1, 1);
hold on
x = [grComV(:); flipud(grComV(:))];
for j = 1:4
    y = [fvaComMin(j, :), fliplr(fvaComMax(j, :))];
    p(j, 1) = plot(x(~isnan(y)), y(~isnan(y)), 'LineWidth', 2);
    p(j, 1).Color = col(j, :);
end
tl(1) = title('\underline{SteadyCom}', 'Interpreter', 'latex');
tl(1).Position = [0.7 1.01 0];
ax(1) = gca;
ax(1).XTick = 0.66:0.02:0.74;
ax(1).YTick = 0:0.2:1;
xlim([0.66 0.74])
ylim([0 1])

lg = legend(lgLabel);
lg.Box = 'off';
yl(1) = ylabel('Relative abundance');
xl(1) = xlabel('Community growth rate (h^{-1})');
% FBA
grFBAV = grFBA * optGRpercentFBA / 100;
x = [grFBAV(:); flipud(grFBAV(:))];
subplot(2, 1, 2);
hold on
% plot j=1:2 only because 3:4 overlap with 1:2
for j = 1:2
    y = [fvaFBAMin(j, :), fliplr(fvaFBAMax(j, :))] ./ x';
    % it is possible some values > 1 because the total biomass produced is
    % only bounded below when calling fluxVariability. Would be strictly
    % equal to 1 if sum(biomass) = optGRpercentFBA(jGr) * grFBA is constrained. Treat them as
    y(y>1) = 1;
    p(j, 2)= plot(x(~isnan(y)), y(~isnan(y)), 'LineWidth', 2);
    p(j, 2).Color = col(j, :);
end
tl(2) = title('\underline{Joint FBA}', 'Interpreter', 'latex');
tl(2).Position = [0.55 1.01 0];
ax(2) = gca;
ax(2).XTick = 0.52:0.02:0.58;
ax(2).YTick = 0:0.2:1;
xlim([0.52 0.58])
ylim([0 1])
xl(2) = xlabel('Community growth rate (h^{-1})');
yl(2) = ylabel('Relative abundance');
ax(1).Position = [0.1 0.6 0.5 0.32];
ax(2).Position = [0.1 0.1 0.5 0.32];
lg.Position = [0.65 0.65 0.1 0.27];
```





The direct use of FVA compared to FVA under the SteadyCom framework gives very little information on the organism's abundance. The ranges for almost all growth rates span from 0 to 1. In contrast, `SteadyComFVA` returns results with the expected co-existence of all four mutants. When the growth rates get closer to the maximum, the ranges shrink to unique values.

Analyze Pairwise Relationship Using SteadyComPOA

Now we would like to see at a given growth rate, how the abundance of an organism influences the abundance of another organism. We check this by iteratively fixing the abundance of an organism at a level (independent variable) and optimizing for the maximum and minimum allowable abundance of another organism (dependent variable). This is what `SteadyComPOA` does.

Set up the option structure and call `SteadyComPOA`. `Nstep` is an important parameter to designate how many intermediate steps are used or which values between the min and max values of the independent variable are used for optimizing the dependent variable. `savePOA` options must be supplied with a non-empty string or a default name will be used for saving the POA results. By default, the function analyzes all possible pairs in `options.rxnNameList`. To analyze only particular pairs, use `options.pairList`. See `help SteadyComPOA` for more details.

```
options.savePOA = ['POA' filesep 'EcCom'];  % directory and fila name for saving POA results
options.optGRpercent = [99 90 70 50];  % analyze at these percentages of max. growth rate
% Nstep is the number of intermediate steps that the independent variable will take different
% or directly the vector of values, e.g. Nsetp = [0, 0.5, 1] implies fixing the independent va
% 50% from the min to the max and the maximum value respectively to find the attainable range
% Here use small step sizes when getting close to either ends of the flux range
```

```
a = 0.001*(1000.^((0:14)/14));
options.Nstep = sort([a (1-a)]);
[POAtable, fluxRange, Stat, GRvector] = SteadyComPOA(EcCom, options);
```

```
Find maximum community growth rate..
Model feasible at maintenance. Time elapsed: 1 / 1 sec
Iter        LB   To test        UB  Time elapsed (iteration/total)
   1  0.000000  0.500000       Inf  0 / 1 sec
   2  0.500000  0.721279       Inf  6 / 8 sec
   3  0.721279  0.735372       Inf  0 / 8 sec
   4  0.735372  0.742726       Inf  0 / 8 sec
   5  0.735372  0.739049  0.742726  0 / 9 sec
   6  0.735372  0.737211  0.739049  0 / 9 sec
   7  0.735372  0.736291  0.737211  0 / 9 sec
   8  0.735372  0.735832  0.736291  0 / 10 sec
   9  0.735832  0.736062  0.736291  2 / 12 sec
  10  0.735832  0.735947  0.736062  0 / 12 sec
  11  0.735947  0.736004  0.736062  3 / 15 sec
  12  0.735947  0.735975  0.736004  0 / 15 sec
  13  0.735975  0.735990  0.736004  3 / 18 sec
  14  0.735990  0.735997  0.736004  0 / 19 sec
  15  0.735990  0.735993  0.735997  0 / 19 sec
  16  0.735990  0.735991  0.735993  1 / 20 sec
  17  0.735990  0.735991  0.735991  1 / 20 sec
Maximum community growth rate: 0.735991 (abs. error < 1e-06). Time elapsed: 38 sec

FVA for 4 sets of fluxes/biomass at growth rate 0.728631 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.202839  0.601214
   2   50      X_Ec2  0.146588  0.407296
   3   75      X_Ec3  0.074774  0.287238
   4  100      X_Ec4  0.114380  0.325063

POA for 6 pairs of reactions at growth rate 0.728631
Start from #1 X_Ec1 vs #2 X_Ec2.
          Rxn1           Rxn2    corMin        r2    corMax        r2   Time
         X_Ec1          X_Ec2   -0.2899    0.7906   -0.3790    0.6965   2017-07-08 12:23:19
         X_Ec1          X_Ec3   -0.1544    0.6270   -0.2471    0.4450   2017-07-08 12:25:57
         X_Ec1          X_Ec4   -0.4653    0.9101   -0.5177    0.9973   2017-07-08 12:27:05
         X_Ec2          X_Ec3   -0.6496    0.7685   -0.6717    0.8522   2017-07-08 12:28:56
         X_Ec2          X_Ec4    0.1349    0.4751    0.0784    0.0430   2017-07-08 12:30:31
         X_Ec3          X_Ec4    0.1294    0.1494   -0.0104    0.0007   2017-07-08 12:32:54
Finished. Save final results to POA/EcCom_GR0.73.mat

FVA for 4 sets of fluxes/biomass at growth rate 0.662392 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.049649  0.776783
   2   50      X_Ec2  0.042603  0.703511
   3   75      X_Ec3  0.023523  0.676937
   4  100      X_Ec4  0.032553  0.678141

POA for 6 pairs of reactions at growth rate 0.662392
Start from #1 X_Ec1 vs #2 X_Ec2.
          Rxn1           Rxn2    corMin        r2    corMax        r2   Time
         X_Ec1          X_Ec2   -0.0435    0.1361   -0.2431    0.1910   2017-07-08 12:34:51
         X_Ec1          X_Ec3   -0.0156    0.0525   -0.2235    0.1529   2017-07-08 12:37:15
         X_Ec1          X_Ec4   -0.7412    0.7577   -0.9030    0.9959   2017-07-08 12:38:32
         X_Ec2          X_Ec3   -0.8039    0.7127   -0.9807    0.9995   2017-07-08 12:40:17
         X_Ec2          X_Ec4    0.0335    0.1287   -0.1552    0.0715   2017-07-08 12:42:58
         X_Ec3          X_Ec4    0.0420    0.1731   -0.2665    0.1762   2017-07-08 12:45:49
Finished. Save final results to POA/EcCom_GR0.66.mat

FVA for 4 sets of fluxes/biomass at growth rate 0.515193 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.007597  0.900618
   2   50      X_Ec2  0.007502  0.882572
```

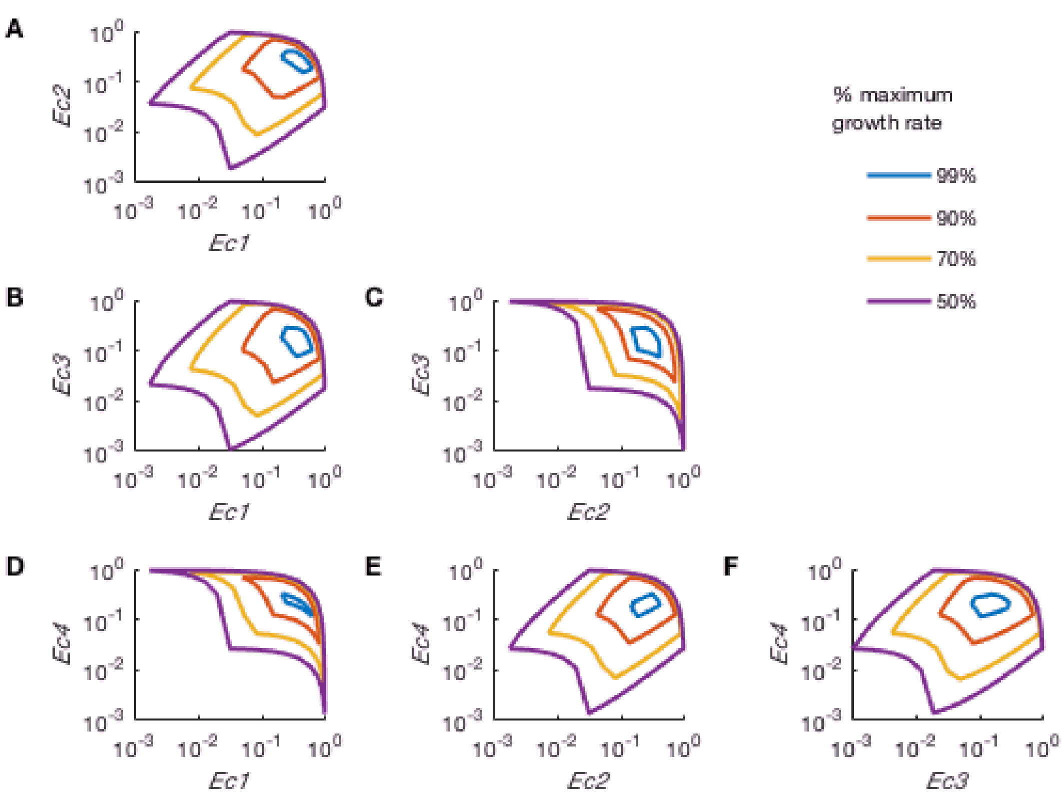
```
   3   75      X_Ec3  0.004281  0.878578
   4  100      X_Ec4  0.005731  0.875423

POA for 6 pairs of reactions at growth rate 0.515193
Start from #1 X_Ec1 vs #2 X_Ec2.
           Rxn1           Rxn2     corMin         r2     corMax         r2    Time
          X_Ec1          X_Ec2     0.0081     0.0272    -0.3249     0.2274    2017-07-08 12:47:20
          X_Ec1          X_Ec3     0.0048     0.0292    -0.3283     0.2194    2017-07-08 12:49:03
          X_Ec1          X_Ec4    -0.6082     0.5468    -0.9827     0.9997    2017-07-08 12:50:39
          X_Ec2          X_Ec3    -0.7091     0.5631    -0.9993     1.0000    2017-07-08 12:51:58
          X_Ec2          X_Ec4     0.0184     0.2030    -0.3124     0.1992    2017-07-08 12:53:46
          X_Ec3          X_Ec4     0.0233     0.2873    -0.3990     0.2885    2017-07-08 12:55:40
Finished. Save final results to POA/EcCom_GR0.52.mat

FVA for 4 sets of fluxes/biomass at growth rate 0.367995 :
  No    %       Name       Min       Max
   1   25      X_Ec1  0.001740  0.949700
   2   50      X_Ec2  0.001818  0.943476
   3   75      X_Ec3  0.001044  0.942565
   4  100      X_Ec4  0.001389  0.940487

POA for 6 pairs of reactions at growth rate 0.367995
Start from #1 X_Ec1 vs #2 X_Ec2.
           Rxn1           Rxn2     corMin         r2     corMax         r2    Time
          X_Ec1          X_Ec2     0.0105     0.1604    -0.4027     0.3042    2017-07-08 12:57:48
          X_Ec1          X_Ec3     0.0061     0.1641    -0.4419     0.3235    2017-07-08 12:59:19
          X_Ec1          X_Ec4    -0.5686     0.4609    -0.9961     1.0000    2017-07-08 13:00:34
          X_Ec2          X_Ec3    -0.5386     0.4367    -1.0000     1.0000    2017-07-08 13:01:42
          X_Ec2          X_Ec4     0.0118     0.3041    -0.4265     0.3032    2017-07-08 13:03:00
          X_Ec3          X_Ec4     0.0142     0.4015    -0.5131     0.4019    2017-07-08 13:04:13
Finished. Save final results to POA/EcCom_GR0.37.mat
```

POAtable is a *n*-by-*n* cell if there are *n* targets in `options.rxnNameList`. `POAtable{i, i}` is a *Nstep*-by-1-by-*Ngr* matrix where *Nstep* is the number of intermediate steps detemined by `options.Nstep` and *Ngr* is the number of growth rates analyzed. `POAtable{i, i}(:, :, k)` is the values at which the *i*-th target is fixed for the community growing at the growth rate `GRvector(k)`. POAtable{i, j} is a *Nstep*-by-2-by-*Ngr* matrix where `POAtable{i, j}(:, 1, k)` and `POAtable{i, j}(:, 2, k)` are respectively the min. and max. values of the *j*-th target when fixing the *i*-th target at the corresponding values in `POAtable{i, i}(:, :, k)`. `fluxRange` contains the min. and max. values for each target (found by calling `SteadyComFVA`). `Stat` is a *n*-by-*n-by-Ngr* structure array, each containing two fields: `*.cor`, the correlatiion coefficient between the max/min values of the dependent variable and the independent variable, and `*.r2`, the R-squred of linear regression. They are also outputed in the command window during computation. All the computed results are also saved in the folder 'POA' starting with the name 'EcCom', followed by 'GRxxxx' denoting the growth rate at which the analysis is performed.

Plot the results (see also Figure 3 in ref. [1]):

```
nSp = 4;
spLab = {'{\it Ec1 }';'{\it Ec2 }';'{\it Ec3 }';'{\it Ec4 }'};
mark = {'A', 'B', 'D', 'C', 'E', 'F'};
nPlot = 0;
for j = 1:nSp
    for k = 1:nSp
        if k > j
            nPlot = nPlot + 1;
            ax(j, k) = subplot(nSp-1, nSp-1, (k - 2) * (nSp - 1) + j);
            hold on
            for p = 1:size(POAtable{1, 1}, 3)
                x = [POAtable{j, j}(:, :, p);POAtable{j, j}(end:-1:1, :, p);...
                    POAtable{j, j}(1, 1, p)];
                y = [POAtable{j, k}(:, 1, p);POAtable{j, k}(end:-1:1, 2, p);...
```

```
                    POAtable{j, k}(1, 1, p)];
                plot(x(~isnan(y)), y(~isnan(y)), 'LineWidth', 2)
            end
            xlim([0.001 1])
            ylim([0.001 1])
            ax(j, k).XScale = 'log';
            ax(j, k).YScale = 'log';
            ax(j, k).XTick = [0.001 0.01 0.1 1];
            ax(j, k).YTick = [0.001 0.01 0.1 1];
            ax(j, k).YAxis.MinorTickValues=[];
            ax(j, k).XAxis.MinorTickValues=[];
            ax(j, k).TickLength = [0.03 0.01];
            xlabel(spLab{j});
            ylabel(spLab{k});
            tx(j, k) = text(10^(-5), 10^(0.1), mark{nPlot}, 'FontSize', 12, 'FontWeight', 'bol
        end
    end
end
lg = legend(strcat(strtrim(cellstr(num2str(options.optGRpercent(:)))), '%'));
lg.Position = [0.7246 0.6380 0.1700 0.2015];
lg.Box='off';
subplot(3, 3, 3, 'visible', 'off');
t = text(0.2, 0.8, {'% maximum';'growth rate'});
for j = 1:nSp
    for k = 1:nSp
        if k>j
            ax(j, k).Position = [0.15 + (j - 1) * 0.3, 0.8 - (k - 2) * 0.3, 0.16, 0.17];
            ax(j, k).Color = 'none';
        end
    end
end
```



There are two patterns observed. The two pairs showing negative correlations, namely Ec1 vs Ec4 (panel D) and Ec2 vs Ec3 (panel C) are indeed competing for the same amino acids with each other (Ec1 and Ec4 competing for Lys and Met; Ec2 and Ec4 competing for Arg and Phe). Each of the other pairs showing positive correlations are indeed the cross feeding pairs, e.g., Ec1 and Ec2 (panel A) cross feeding on Arg and Lys. See ref. [1] for more detailed discussion.

Parallelization and Timing

SteadyCom in general can be finished within 20 iterations, i.e. solving 20 LPs (usually faster if using Matlab fzero) for an accuracy of 1e-6 for the maximum community growth rate. The actual computation time depends on the size of the community metabolic network. The current `EcCom` model has 6971 metabolites and 9831 reactions. It took 18 seconds for a MacBook Pro with 2.5 GHz Intel Core i5, 4 GB memory running Matlab R2016b and Cplex 12.7.1.

Since the FVA and POA analysis can be time-consuming for large models with a large number of reactions to be analyzed, SteadyComFVA and SteadyComPOA support parrallelization using the Matlab Distributed Computing Toolbox (`parfor` for SteadyComFVA and `spmd` for SteadyComPOA).

Test SteadyCom with 2 threads:

```
options.rxnNameList = EcCom.rxns(1:100);  % test FVA for the first 50 reactions
options.optGRpercent = 99;
options.algorithm = 1;
options.threads = 1;  % test single-thread computation first
options.verbFlag = 0;  % no verbose output
tic;
[minF1, maxF1] = SteadyComFVA(EcCom, options);
t1 = toc;
if isempty(gcp('nocreate'))
    parpool(2)  % start a parallel pool
end
options.threads = 2;  % two threads (0 to use all available workers)
tic;
[minF2, maxF2] = SteadyComFVA(EcCom, options);  % test single-thread computation first
```

```
t2 = toc;
fprintf('Maximum difference between the two solutions: %.4e\n', max(max(abs(minF1 - minF2)), m
```

```
Maximum difference between the two solutions: 5.2591e-09
```

```
fprintf('\nSingle-thread computation: %.0f sec\nTwo-thread computation: %.0f sec\n', t1, t2);
```

```
Single-thread computation: 96 sec
Two-thread computation: 58 sec
```

If there are many reactions to be analyzed, use `options.saveFVA` to give a relative path for saving the intermediate results. Even though the computation is interrupted, by calling `SteadyComFVA` with the same `options.saveFVA`, the program will detect previously saved results and continued from there.

Test SteadyComPOA with 2 threads:

```
options.rxnNameList = EcCom.rxns(find(abs(result.flux) > 1e-2, 6));
```

```
options.savePOA = 'POA/EcComParallel';  % save with a new name
options.verbFlag = 3;
options.threads = 2;
options.Nstep = 5;  % smaller steps for quicker computation
tic;
[POAtable1, fluxRange1] = SteadyComPOA(EcCom, options);
```

```
Find maximum community growth rate..
Model feasible at maintenance. Time elapsed: 2 / 2 sec
Iter        LB   To test        UB  Time elapsed (iteration/total)
   1  0.000000  0.500000       Inf  0 / 2 sec
   2  0.500000  0.721279       Inf  6 / 8 sec
   3  0.721279  0.735372       Inf  0 / 8 sec
   4  0.735372  0.742726       Inf  0 / 9 sec

 Func-count    x          f(x)             Procedure
    2        0.735372  -0.000807615        initial
    3        0.735378   -0.00079987        interpolation
    4         0.73599  -1.26398e-06        interpolation
    5         0.73599  -1.26398e-06        interpolation

Zero found in the interval [0.735372, 0.742726]
Maximum community growth rate: 0.735990 (abs. error < 1e-06). Time elapsed: 19 sec

FVA for 6 sets of fluxes/biomass at growth rate 0.728630 :

Thread 2: 33.33% finished. 2017-07-08 19:34:49
Thread 1: 33.33% finished. 2017-07-08 19:34:49
Thread 2: 66.67% finished. 2017-07-08 19:34:51
Thread 1: 66.67% finished. 2017-07-08 19:34:51
Thread 1: 100.00% finished. 2017-07-08 19:34:52
Thread 2: 100.00% finished. 2017-07-08 19:34:52

POA for 15 pairs of reactions at growth rate 0.728630
Start from #1 Ec13HAD100 vs #2 Ec13HAD120.
           Rxn1           Rxn2    corMin        r2    corMax        r2   Time
POA in parallel...
Lab 2:
       Ec13HAD120     Ec13HAD160    0.0956    0.5000   -0.8434    0.9667   2017-07-08 19:36:18
Lab 1:
       Ec13HAD100     Ec13HAD120   -0.2210    0.6000    1.1749    0.3718   2017-07-08 19:37:06
       Ec13HAD100     Ec13HAD121    0.2429    0.7227    0.4244    0.2168   2017-07-08 19:39:13
Lab 2:
       Ec13HAD121     Ec13HAD140   -0.0833    0.5000   -1.5267    0.9861   2017-07-08 19:40:17
Lab 1:
       Ec13HAD100     Ec13HAD140   -0.0915    0.5000   -0.3698    0.0569   2017-07-08 19:40:51
Lab 2:
       Ec13HAD121     Ec13HAD141    1.0000    1.0000    1.0000    1.0000   2017-07-08 19:42:15
Lab 1:
       Ec13HAD100     Ec13HAD141    0.0924    1.0000    1.2210    0.9786   2017-07-08 19:44:18
Lab 2:
       Ec13HAD121     Ec13HAD160   -0.0837    0.5000   -0.2478    0.0302   2017-07-08 19:45:56
Lab 1:
       Ec13HAD100     Ec13HAD160   -0.1232    0.9423    0.0763    1.0000   2017-07-08 19:47:21
Lab 2:
       Ec13HAD140     Ec13HAD141   -0.0026    0.0014   -0.7316    0.9577   2017-07-08 19:49:46
Lab 1:
       Ec13HAD120     Ec13HAD121   -0.0762    0.8288   -0.6769    1.0000   2017-07-08 19:49:58
       Ec13HAD120     Ec13HAD140    0.0956    0.5000   -0.7402    0.8596   2017-07-08 19:52:58
Lab 2:
       Ec13HAD140     Ec13HAD160    0.1547    0.6000   -0.8932    0.9556   2017-07-08 19:53:01
Lab 1:
       Ec13HAD120     Ec13HAD141    0.0637    1.0000   -0.6611    0.9793   2017-07-08 19:54:45
Lab 2:
       Ec13HAD141     Ec13HAD160   -0.1255    0.6000    0.1325    0.0153   2017-07-08 19:55:55
Lab 1:
```

```
  Current loop finished. Stop other workers...
  All workers have ceased. Redistributing...
Finished. Save final results to POA/EcComParallel_GR0.73.mat
```

```
t3 = toc;
```

The parallelization code uses spmd and will redistribute jobs once any of the workers has finished to maximize the computational efficiency.

```
options.savePOA = 'POA/EcComSingeThread';
options.threads = 1;
tic;
[POAtable2, fluxRange2] = SteadyComPOA(EcCom, options);
```

```
Find maximum community growth rate..
Model feasible at maintenance. Time elapsed: 1 / 1 sec
Iter        LB    To test        UB  Time elapsed (iteration/total)
   1  0.000000  0.500000       Inf  0 / 1 sec
   2  0.500000  0.721279       Inf  4 / 5 sec
   3  0.721279  0.735372       Inf  0 / 6 sec
   4  0.735372  0.742726       Inf  0 / 6 sec

 Func-count    x          f(x)             Procedure
    2        0.735372  -0.000807615        initial
    3        0.735378   -0.00079987        interpolation
    4         0.73599  -1.26398e-06        interpolation
    5         0.73599  -1.26398e-06        interpolation

Zero found in the interval [0.735372, 0.742726]
Maximum community growth rate: 0.735990 (abs. error < 1e-06). Time elapsed: 12 sec

FVA for 6 sets of fluxes/biomass at growth rate 0.728630 :
  No    %       Name        Min       Max
   1    17 Ec13HAD100  0.052591  0.217439
   2    33 Ec13HAD120  0.000000  0.262936
   3    50 Ec13HAD121  0.022231  0.202541
   4    67 Ec13HAD140  0.000000  0.243774
   5    83 Ec13HAD141  0.022231  0.202541
   6   100 Ec13HAD160  0.000000  0.251518

POA for 15 pairs of reactions at growth rate 0.728630
Start from #1 Ec13HAD100 vs #2 Ec13HAD120.
          Rxn1           Rxn2     corMin        r2     corMax        r2    Time
     Ec13HAD100     Ec13HAD120    -0.2210    0.6000    1.1749    0.3718   2017-07-08 19:56:57
     Ec13HAD100     Ec13HAD121     0.2429    0.7227    0.4244    0.2168   2017-07-08 19:57:37
     Ec13HAD100     Ec13HAD140    -0.0915    0.5000   -0.3698    0.0569   2017-07-08 19:58:17
     Ec13HAD100     Ec13HAD141     0.0924    1.0000    1.2210    0.9786   2017-07-08 19:59:25
     Ec13HAD100     Ec13HAD160    -0.1232    0.9423    0.0763    1.0000   2017-07-08 20:00:22
     Ec13HAD120     Ec13HAD121    -0.0762    0.8288   -0.6769    1.0000   2017-07-08 20:01:12
     Ec13HAD120     Ec13HAD140     0.0956    0.5000   -0.7402    0.8596   2017-07-08 20:02:07
     Ec13HAD120     Ec13HAD141     0.0637    1.0000   -0.6611    0.9793   2017-07-08 20:02:42
     Ec13HAD120     Ec13HAD160     0.0956    0.5000   -0.8434    0.9667   2017-07-08 20:02:56
     Ec13HAD121     Ec13HAD140    -0.0833    0.5000   -1.5267    0.9861   2017-07-08 20:04:16
     Ec13HAD121     Ec13HAD141     1.0000    1.0000    1.0000    1.0000   2017-07-08 20:04:59
     Ec13HAD121     Ec13HAD160    -0.0837    0.5000   -0.2478    0.0302   2017-07-08 20:06:07
     Ec13HAD140     Ec13HAD141    -0.0026    0.0014   -0.7316    0.9577   2017-07-08 20:07:25
     Ec13HAD140     Ec13HAD160     0.1547    0.6000   -1.0456    0.9754   2017-07-08 20:08:09
     Ec13HAD141     Ec13HAD160    -0.0837    0.5000   -0.2478    0.0302   2017-07-08 20:08:47
Finished. Save final results to POA/EcComSingeThread_GR0.73.mat
```

```
t4 = toc;
dev = 0;
for i = 1:size(POAtable1, 1)
    for j = i:size(POAtable1, 2)
```

```
        dev = max(max(max(abs(POAtable1{i, j} - POAtable2{i, j}))));
        dev = max(dev, max(max(abs(fluxRange1 - fluxRange2))));
    end
end
fprintf('Maximum difference between the two solutions: %.4e\n', dev);
```

```
Maximum difference between the two solutions: 1.3801e-10
```

```
fprintf('\nSingle-thread computation: %.0f sec\nTwo-thread computation: %.0f sec\n', t4, t3);
```

```
Single-thread computation: 769 sec
Two-thread computation: 908 sec
```

The advantage will be more significant for more targets to analyzed and more threads used. Similar to `SteadyComFVA`, `SteadyComPOA` also supports continuation from previously interrupted computation by calling with the same `options.savePOA`.

REFERENCES

[1] Chan SHJ, Simons MN, Maranas CD (2017) SteadyCom: Predicting microbial abundances while ensuring community stability. PLoS Comput Biol 13(5): e1005539. https://doi.org/10.1371/journal.pcbi.1005539

[2] Khandelwal RA, Olivier BG, Röling WFM, Teusink B, Bruggeman FJ (2013) Community Flux Balance Analysis for Microbial Consortia at Balanced Growth. PLoS ONE 8(5): e64567. https://doi.org/10.1371/journal.pone.0064567