matTFA cheat sheet

Basic model variables

S	Stoichiometric matrix	FBA
rxns	Reaction abbreviations	
Mets	Metabolite abbreviations	
b		
metSEEDID		
metCompSymbol		
CompartmentData		
thermo_units		
metNames		
metFormulas		
description		
rev	Reversibility	
lb	Lower bound (reaction flux)	
ub	Upper bound (reaction flux)	
c	Objective function (reaction)	
rxnMapResult		
grRules	Gene reaction rules	
rules		
subSystems	Subsystems	
genes	Genes	
rxnGeneMat	Reaction gene Matrix	
isTrans		
metDeltaGFstd		
metDeltaGFerr		
metDeltaGFtr		
metCharge		
metMass		
struct_cues		
rxnThermo		
rxnDeltaGR		
rxnDeltaGRerr		
rxnComp		
A	“Stoichiometric” Constraint matrix	TFA
constraintType	Type of the constraint	
constraintNames	Name of a constraint	
rhs	Right hand side of the problem	
varNames	Variable names e.g. ”LC_met_id”	
vartypes	Types of Variables C or B	
var_lb	Lower bounds of the Variables	
var_ub	Upper bounds of the Variables	
f	Objective function on Variables	
objtype	Direction of the objective default -1 for max	
types		

FBA functions

changeRxnBounds	Change upper or lower bounds of a reaction or a set of reactions
optimizeCbModel	Solve a flux balance analysis problem
changeObjective	Changes the objective function of a constraint-based model
singleGeneDeletion	Performs single gene deletion analysis using FBA, MOMA or linearMOMA
fluxVariability	Performs flux variability analysis

TFA functions

prepModelforTFA	Prepare the cobra model for TFA
convToTFA	Convert the prepared model to TFA model
solveTFAmodelCplex	Solve the TFA optimization problem
runTMinMax	Performs thermodynamics variability analysis
addNetFluxVariables	The net flux variables work similar to the FBA fluxes: NF_rxn = F_rxn - B_rxn
changeTFArxnBounds	function (for PASB course) to modify the TFA bounds in a similar form as in COBRA

TFA Variable names

Variable Name	Meaning	Unit	Type
F_rxn_id	Forward reaction flux	mmol/gDW/hr	Continuous
R_rxn_id	Backward reaction flux	mmol/gDW/hr	Continuous
FU_rxn_id	Forward reaction flux is used	-	Binary
BU_rxn_id	Backward reaction flux is used	-	Binary
DG_rxn_id	Free energy of reaction	kcal/mol	Continuous
DGo_rxn_id	Standard free energy of reaction	kcal/mol	Continuous
LC_met_id	Logarithmic concentration	log(mol)	Continuous
Optional:			
NF_rxn_id	Net reaction flux	mmol/gDW/hr	Continuous

All variable names can be found in tmodel.varNames and are constructed by substituting 'rxn_id' with the respective reaction id in fba_model.rxns and 'met_id' by the respective metabolite id in fba_model.metabolites

Because of different structure of TFA models with respect to FBA models (i.e. we have different fields for upper- and lower-bounds of the reactions), you need to use another function in order to assign boundaries to uptake fluxes. This function is `changeTFArxnBounds.`

changeRxnBounds

```
changeRxnBounds Change upper or lower bounds of a reaction or a set of
reactions

 model = changeRxnBounds(model,rxnNameList,value,boundType)

INPUTS
 model          COBRA model structure
 rxnNameList    List of reactions (cell array or string)
 value          Bound values
                Can either be a vector or a single scalar value if the same
                bound value is to be assinged to all reactions

OPTIONAL INPUT
 boundType      'u' - upper, 'l' - lower, 'b' - both (Default = 'b')
                Bound type can either be a cell array of strings or a
                string with as many letters as there are reactions in
                rxnNameList

OUTPUT
 model          COBRA model structure with modified reaction bounds

 Markus Herrgard 4/21/06
```

optimizeCbModel

```
optimizeCbModel Solve a flux balance analysis problem

 Solves LP problems of the form: max/min c'*v
                                 subject to S*v = b         : y
                                            lb <= v <= ub   : w
 FBAsolution = optimizeCbModel(model,osenseStr,minNormFlag)

INPUT
 model (the following fields are required - others can be supplied)
   S            Stoichiometric matrix
   b            Right hand side = dx/dt
   c            Objective coefficients
   lb           Lower bounds
   ub           Upper bounds

OPTIONAL INPUTS
 osenseStr      Maximize ('max')/minimize ('min') (opt, default = 'max')

 minNorm        {(0), 'one', > 0 , n x 1 vector}, where [m,n]=size(S);
                0       Default, normal LP
                'one'   Minimise the Taxicab Norm using LP.
                                  min |v|
                                    s.t. S*v = b
                                         c'v = f
                                         lb <= v <= ub
                -----
                The remaining options work only with a valid QP solver:
                -----
                > 0     Minimises the Euclidean Norm of internal fluxes.
                        Typically 1e-6 works well.
                                  min ||v||
                                    s.t. S*v = b
                                         c'v = f
```

```
                                    lb <= v <= ub
                n x 1    Forms the diagonal of positive definiate
                         matrix F in the quadratic program
                                 min 0.5*v'*F*v
                                 st. S*v = b
                                     c'*v = f
                                     lb <= v <= ub

  allowLoops     {0,(1)} If false,  then instead of a conventional FBA,
                 the solver will run an MILP version which does not allow
                 loops in the final solution.  Default is true.
                 Runs much slower when set to false.
                 See addLoopLawConstraints.m to for more info.

 OUTPUT
  FBAsolution
    f          Objective value
    x          Primal
    y          Dual
    w          Reduced costs
    s          Slacks
    stat       Solver status in standardized form
                1   Optimal solution
                2   Unbounded solution
                0   Infeasible
               -1  No solution reported (timelimit, numerical problem etc)
    origStat   Original status returned by the specific solver
```

changeObjective

changeObjective Changes the objective function of a constraint-based model

```
  model = changeObjective(model,rxnNameList,objectiveCoeff)

 INPUTS
  model              COBRA structure
  rxnNameList        List of reactions (cell array or string)

 OPTIONAL INPUT
  objectiveCoeff     Value of objective coefficient for each reaction
                     (Default = 1)

 OUTPUT
  model              COBRA model structure with new objective

  Monica Mo & Markus Herrgard - 8/21/06
```

singleGeneDeletion

singleGeneDeletion Performs single gene deletion analysis using FBA, MOMA or linearMOMA

```
  [grRatio,grRateKO,grRateWT,delRxns,hasEffect] =
singleGeneDeletion(model,method,geneList,verbFlag)

 INPUT
  model          COBRA model structure including gene-reaction associations
```

```
 OPTIONAL INPUT
  method         Either 'FBA', 'MOMA', or 'lMOMA' (Default = 'FBA')
  geneList       List of genes to be deleted (default = all genes)
  verbFlag       Verbose output (Default false)
  uniqueGene     Run unique gene deletion (default = 0).


 OUTPUTS
  grRatio        Computed growth rate ratio between deletion strain and wild
type
  grRateKO       Deletion strain growth rates (1/h)
  grRateWT       Wild type growth rate (1/h)
  hasEffect      Does a gene deletion affect anything (i.e. are any
reactions
                 removed from the model)
  delRxns        List of deleted reactions for each gene KO
  fluxSolution   FBA/MOMA/lMOMA fluxes for KO strains

  Markus Herrgard 8/7/06
  Aurich/Thiele 11/2015 unique gene deletion option (delete all alternate
transcripts and if solKO.stat not 1 or 5, grRateKO(i) = NaN;)
```

fluxVariability

```
 fluxVariability Performs flux variablity analysis

  [minFlux,maxFlux] =
fluxVariability(model,optPercentage,osenseStr,rxnNameList,verbFlag,
allowLoops)

 INPUT
  model              COBRA model structure

 OPTIONAL INPUTS
  optPercentage      Only consider solutions that give you at least a
certain
                     percentage of the optimal solution (Default = 100
                     or optimal solutions only)
  osenseStr          Objective sense ('min' or 'max') (Default = 'max')
  rxnNameList        List of reactions for which FVA is performed
                     (Default = all reactions in the model)
  verbFlag           Verbose output (opt, default false)
  allowLoops         Whether loops are allowed in solution. (Default = true)
                     See optimizeCbModel for description

 OUTPUT
  minFlux            Minimum flux for each reaction
  maxFlux            Maximum flux for each reaction

 OPTIONAL OUTPUT
  Vmin           Matrix of column flux vectors, where each column is a
                 separate minimization.
  Vmax           Matrix of column flux vectors, where each column is a
                 separate maximization.
```

prepModelforTFA

```
  prepares a COBRA toolbox model for TFBA analysis by doing the following:
  - checks if a reaction is a transport reaction
  - checks the ReactionDB for Gibbs energies of formation of metabolites
  - computes the Gibbs energies of reactions

  INPUTS:
    model - COBRA toolbox metabolic model structure
    ReactionDB - ReactionDB database with all the compounds and reactions
        data
    CompartmentData - structure storing the compartmental pH,
        ionic strength, membrane potential, etc.
    replaceData - Use metabolite names from the ReactionDB
    verboseFlag - change verbosity
    writeToFileFlag - write LP to file

  OUTPUT:
    modeloutput - processed COBRA toolbox model ready to be converted to
TFBA
        model
```

convToTFA

converts a model into a TFA ready model by adding the
thermodynamic constraints required

INPUTS
- model: model in COBRA toolbox structure but with compound mappings and compartment data. Only adds constraints to those reactions with a 1 in model.rxnThermo
- ReactionDB: Database with all the thermodynamic information from Alberty. In this routine we only copy the types of variables from the ReactionDB as the other data has already been added
- rxnNameListNoThermo: List of reaction names for which we do not want to have thermodynamic constraints generated
- flagInfeasibility: flag that determines whether to investigate solution feasibility upon TFA model generation. If we do not wish to investigate this we can set this flag to 'NO' (default). If we wish to further investigate the cause of infeasibility, we have already implemented so far a way to do this with slack variables for the DG-naught (flagInfeasibility = 'DGo').
- rxnNameListNoDGoRelax: List of reaction names for which we do not want to have thermodynamic constraints generated. As there might exist alternative sets of DGo that can be relaxed o achieve feasibility, we might want to avoid relaxing ATP synthase reactions, or reactions from the ETC chain, and instead relax peripheral reactions. For the reactions that we put in this list there will be no slack-DGo variables created.
- minObjSolVal: minimum lower bound of the objective (that is maximized) that the desired solution should satisfy.
- flagToAddPotentials: flag to determine whether to introduce chemical potentials as variables or not (default:false)
- flagToAddLnThermoDisp: flag to determine whether to add thermodynamic displacement as a variable to the model by introducing its corresponding
  constraint or not (default:false)
- verboseFlag: prints out information about the generation of constraints
- printLP: prints out the LP formulation
- flagMCA_FarEquilibrium: flag to add Gamma constraints for MCA to ensure we have no zero displacement reactions

solveTFAmodelCplex

```
  Solve a model using specific solver settings. More details in
  changeToCPLEX_WithOptions

  INPUTS
    tModel: a TFA-ready model (has a .A matrix)
    TimeInSec: timelimit for the solver.
    manualScalingFactor: manual scaling factor for the solver
    mipTolInt: Integer tolerance of the solver
    emphPar: Solver emphasis (trade-offs between speed, feasibility,
        optimality, and moving bounds in MIP - see
https://www.ibm.com/support/knowledgecenter/en/SSSA5P_12.6.0/ilog.odms.cplex.help/CPLEX/Parameters/topics/MIPEmphasis.html)
    feasTol: solver tolerance for feasibility (error on constraints)
    mipDisplay: verbosity of the MIP info display
    CPXPARAMdisp: Turn on/off the cplex problem setup display

 % Changelog
  2017/04/26 - Modified by Pierre on Georgios Fengos's base, to incorporate
  Vikash's Gurobi hooks in a more global fashion, in a similar way COBRA
  solvers are handled.
  Calls the global parameter TFA_MILP_SOLVER, and check if it set to
  something. If not, default to CPLEX using Fengos's code.
  In the long run, we should rename thins function to remove CPLEX from its
  name.
  TODO : Add parameter setting in gurobi call
```

runTMinMax

```
  This function uses cplex to runs a min-max of the specified tFBA-model
  variables. Many of the cplex-solver parameters have been set to some
  default values, but can also be adjusted here.
  TO DO: Add other solvers as options
  runTMinMax(Model,Model.varNames(NFids),200,10^3)
```

addNetFluxVariables

```
  adds NetFluxes variables to the model and their associated constraints

  The variables will be prefixed with "NF_"

  INPUTS:
  - model: a TFA-ready model (has the .A matrix) with no NF_ variables

  OUTPUTS
  - model: a TFA-readay model with NF_ variables
```

changeTFArxnBounds

```
INPUTS
```

```
% model          TFA model structure
% rxnNameList    List of reactions (cell array or string)
% value          Bound values
%                Can either be a vector or a single scalar value if the same
%                bound value is to be assinged to all reactions
%
%OPTIONAL INPUT
% boundType      'u' - upper, 'l' - lower, 'b' - both
%
%OUTPUT
% model          TFA model structure with modified reaction bounds
```