Solution to Homework 5

```
% clear; clc;
load('../Data/e_coli_core.mat');
model=readCbModel('../Data/tcacycle.xml');
```

1. (1 point) Use function *FluxRange* from the exercise to calculate the operational flux range of reactions in the *E. coli* core model.

```
% first determine max biomass
Sol = optimizeCbModel(e_coli_core);
% fix flux through biomass to optimum
e_coli_core.lb(e_coli_core.c~=0) = Sol.f;
e_coli_core.ub(e_coli_core.c~=0) = Sol.f;

% run FluxRange to get flux range at optimal flux, which is the operational
% range
[min_o,max_o] = FluxRange(e_coli_core,1:length(e_coli_core.rxns));
```

2. (1 point) Given model M, suppose we use the following two LPs to find blocked reactions, with z* being the optimal biomass for model M

```
min/max v_j

s.t.

Nv=0              (LP1)

lb<=v<=ub
```

```
min/max v_j

s.t.

Nv=0             (LP2)

V_bio=z*

lb<=v<=ub
```

Which statement is correct?

- The set of blocked reactions obtained from LP1 must be a subset of blocked reactions obtained from LP2.

  --> we may find additional blocked reactions with an additional constraint (v_bio = z*)

3. (8 points) Write function *C = essentiality_check(M,target)* that classifies the relationship of each model reaction on target trait production into three groups: *essential*, *dispensable* and *redundant.* A reaction is called *essential*, if its removal causes zero production of the target of interest. The removal of a *dispensable* reaction decreases the production of target compared to its optimum production. In the case of a *redundant* reaction its removal has no effect on the target production.

```
C_tca = essentiality_check(model,find(strcmp(model.rxnNames,'ex_G6P')))
```

C_tca = 28×2 table

	Reaction_name	Classification
1	'ME1'	'redundant'
2	'ex_G6P'	'target production'
3	'ex_AcCoA'	'essential'
4	'ACO1_ACO2'	'essential'
5	'GAPDH_PGK1_PGAM1_ENO1'	'essential'
6	'SUCLG2_SUCLG1_SUCLA2_ATP_'	'dispensable'
7	'GLUD1_NADPH_'	'redundant'
8	'OGDH_DLST_DLD'	'redundant'
9	'TPI1'	'essential'
10	'GOT1'	'redundant'
11	'MDH1_MDH2'	'dispensable'
12	'GLUD1_NADH_'	'redundant'
13	'PGI'	'essential'
14	'SUCC_DEH'	'essential'
15	'SUCLG2_SUCLG1_SUCLA2_GTP_'	'dispensable'
16	'PFKL'	'redundant'
17	'CS'	'essential'
18	'FBP1'	'essential'
19	'MAS'	'essential'
20	'PCK1'	'essential'
21	'ICL'	'essential'
22	'ALDOA_ALDOB'	'essential'
23	'PKLR'	'redundant'
24	'PDC'	'redundant'
25	'FUM'	'essential'

	Reaction_name	Classification
26	'GPT'	'redundant'
27	'IDH3A_IDH3B_IDH3G'	'redundant'
28	'PC'	'redundant'

Function essentiality_check (HW)

```
function C = essentiality_check(M,target)
%
% Input:
% M          a model struct
% target     a reaction index that represents maximum target production
%            (e.g. maximum flux through a reaction exporting metabolite A will
represent maximizing production of A)
%
% Output:
% C          Table having a column indicating the reaction and a column showing
the classification.
%
% Function body:
Flux_classification = cell(size(M.rxns));
Flux_classification{target} = 'target production';

M_rev = M;
M = convertToIrreversible(M);

% Find the maximum production rate of target

M.c(:) = 0; % clear flux previously optimized
M.c(target) = 1; % set target production of interest to be optimized
% Sol_opt_target = optimizeCbModel(M);
[Sol_opt_target.x,Sol_opt_target.f] = linprog(-M.c,[],[],M.S,M.b,M.lb,M.ub);
Sol_opt_target.f=-Sol_opt_target.f;
% Find the minimum sum over all fluxes at the optimal flux through v_target

% fix target production
M.lb(target) = Sol_opt_target.f;
M.ub(target) = Sol_opt_target.f;
% change objective to sum over all v
M.c(:) = 1;

% Sol_min_v = optimizeCbModel(M,'min');
[Sol_min_v.x,Sol_min_v.f] = linprog(M.c,[],[],M.S,M.b,M.lb,M.ub);
r = find(Sol_min_v.x==0);
% Use the solution of step 3. to classify redundant reaction(s)
```

```
for i=1:length(r(r<size(M_rev.S,2)))
    if M.match(r(i))==0
        Flux_classification(r(i)) = {'redundant'};
    elseif find(r==M.match(r(i))) % both direction are in r
        Flux_classification(r(i)) = {'redundant'};
    end
end
% --> if is empty no redundant flux identified directly

% Categorize the remaining reactions in the model solving an appropriate LP

% change objective back to target production
M.lb(target) = 0;
M.c(:) = 0; M.c(target) = 1;
M_orig = M;
check=find(cellfun(@isempty,Flux_classification));
for i=1:length(check)
    % block reaction i and find optimal flux through target
    M.lb(check(i)) = 0;
    M.ub(check(i)) = 0;
    % if reaction I is reversible block backward flux as well
    if M.match(check(i))~=0
        M.lb(M.match(check(i))) = 0; M.ub(M.match(check(i))) = 0;
    end
%     Sol = optimizeCbModel(M);
    [Sol.x,Sol.f] = linprog(-M.c,[],[],M.S,M.b,M.lb,M.ub);
    Sol.f=-Sol.f;

    if Sol.f == Sol_opt_target.f
        Flux_classification{check(i)} = 'redundant';
    elseif isempty(Sol.f) || Sol.f == 0
        Flux_classification{check(i)} = 'essential';
    elseif Sol.f < Sol_opt_target.f
        Flux_classification{check(i)} = 'dispensable';
    else
        Flux_classification{check(i)} = 'increasing';
    end
    % change bounds back to original
    M = M_orig;
end

C =
table(M_rev.rxns,Flux_classification,'VariableNames',{'Reaction_name','Classific
ation'});
```

```
% The final table has n rows, equal to number of reactions in the input model
(before reversible reaction split!)
end
```

FluxRange (Exercise)

```
function [minimum_flux_lp,maximum_flux_lp] = FluxRange(M,w)

for j=w
    % set the reaction for which we calculate the range
    M.c = zeros(size(M.c));
    M.c(j) = 1;

    % using optimizeCbModel
    minimum_flux(j) = optimizeCbModel(M,'min');
    maximum_flux(j) = optimizeCbModel(M,'max');

    % using linprog
    [~,minimum_flux_lp(j)] = linprog(M.c,[],[],M.S,M.b,M.lb,M.ub);
    [~,maximum_flux_lp(j)] = linprog(-M.c,[],[],M.S,M.b,M.lb,M.ub);

    % for linprog multiply maximization results with -1
    maximum_flux_lp = -maximum_flux_lp;
end
end
```