Solution HW 7 - flux coupling

The type of coupling between two reactions and can be determined by solving the linear fractional program (LFP)

$$\max_{v} \left(\min\right) \frac{v_i}{v_j}$$

s.t.

$$Nv = b$$

$$\forall i, 1 \leq i \leq n, 0 \leq v_i \leq v_i^{max}$$

The LFP can be transformed into a LP, using Charnes-Cooper transformation:

$$\max_{v'} \left(\min\right) v'_i$$

s.t.

$$Nv' = bt$$

$$\forall i, 1 \leq i \neq j \leq n, 0 \leq v'_i \leq v_i^{max} t$$

$$v'_j = 1$$

$$t \geq 0$$

Task 1

Write function C*T = coupling(model)*, that provided a metabolic model in Cobra model format calculates pairwise coupling between all model reactions. The output *CT* is a numeric double matrix with the following numbers indicating the respective type of coupling:

0: not coupled

1: fully coupled

2: partially coupled

3: directionally coupled from i to j ($v_i \neq 0 \Rightarrow v_j \neq 0$)

4: directionally coupled from j to i ($v_j \neq 0 \Rightarrow v_i \neq 0$).

Hint: use function round to round to nearest 4 decimal digits to avoid numeric problems.

Which of the coupling types are symmetric? To improve performance, use the symmetry to avoid calculating the coupling of reaction pairs, where this is not necessary.

Task 2

Calculate reaction coupling for the *E. coli* core model.

For the set of fully coupled reaction pairs $\{R_i, R_j\}$, where $i \neq j$, provide a table that shows the corresponding coupling constant $\alpha_{ij} = \frac{v_i}{v_j}$, $\alpha_{ij} > 0$.

```
load('e_coli_core.mat')
[CT,FC] = coupling(e_coli_core);

% since fully coupled reactions have the same ratio in all flux
% distributions, we can also find one flux distribution v in which reactions
% i and j, that are fully coupled, carry flux and take the ratio v_i/v_j from
there
```

FUNCTIONS

```
function [CT,FC] = coupling(model)

% since we do not want to solve a specific objective we first clear the
% objective in case one is set by defualt
model.c(:) = 0;

% convert to irreversible model since 0 <= v'_i
model = convertToIrreversible(model);

% remove blocked reactions sice we do not consider it as coupling type
[~,vmax] = FluxRange(model);
blk = model.rxns(vmax<1e-10);
model = removeRxns(model,blk);
vmax(vmax<1e-10) = [];

% build updated equality constraint matrix - add column for variable t

%       v        t
Aeq = [model.S zeros(size(model.S,1),1)]; % this part models N of the
constraint Nv'=bt
beq = model.b; % since all entries in model.b are 0 multiplication with t dies
not change it

% build inequality matrix for constraints ∀i,1≤i≠j≤n, 0 ≤ v'_i ≤ v_i^max*t
```

```
 A = [eye(size(model.S,2)) -vmax]; % rewrite constraint v'_i - v_i^max <= 0
 b = zeros(size(A,1)-1,1); % -1 because we will remove the row where i=j

 LB = [model.lb; 0]; % add one zero for t
 UB = [ones(size(model.ub))*1e9; 1e6]; % set it to large numbers to use v'_i ≤
v_i^max*t as upper bound

 f = zeros(length(model.c)+1,1); % all zeros objective vector that will be
changed in loop

 CT = eye(length(model.rxns)); % initialize CT matrix
 % diagonal is one since every reaction is fully coupled to itself
 % off-diagonal entries set to nan to check if value is computed already
 CT(CT==0) = nan;

 OPTIONS = optimset('linprog');
 OPTIONS.Display = 'off';
 FC=[];
 for i=1:length(model.rxns)
     % set objective to v'_i
     f_i = f;
     f_i(i) = 1;
     for j=1:length(model.rxns)
         if i~=j && isnan(CT(i,j)) % check if the value is computed already to
make use of symmetry
             % set constraint v'_j=1
             LB_j = LB;
             LB_j(j) = 1;
             UB_j = UB;
             UB_j(j) = 1;
             % remove row where i=j from A
             A_i = A;
             A_i(j,:) = [];
             [X,Rmax]=linprog(-f_i,A_i,b,Aeq,beq,LB_j,UB_j,OPTIONS);
             Rmax = round(-Rmax,4);
             [~,Rmin]=linprog(f_i,A_i,b,Aeq,beq,LB_j,UB_j,OPTIONS);
             Rmin = round(Rmin,4);
             if isempty(Rmax) || isempty(Rmin) % double check that problem is
feasible
                 disp('Feasibility problem for:');...
                     disp([i j])
             else
                 % categorize coupling
                 if Rmin == 0 && Rmax == round(vmax(i)*X(end),4) % from 0 to
upper bound vmax*t
```

```
                    CT(i,j) = 0; % non-symmetric j could be directinally
coupled to i
                elseif Rmin == Rmax && Rmax > 0 %
                    CT(i,j) = 1;
                    CT(j,i) = 1; % full coupling is a symmetric relation
                    FC(end+1,:) = [i j Rmax];
                elseif Rmin > 0 && Rmax < round(vmax(i)*X(end),4)
                    CT(i,j) = 2;
                    CT(j,i) = 2; % partial coupling is a symmetric relation
                elseif Rmin == 0 && Rmax < round(vmax(i)*X(end),4)
                    CT(i,j) = 3;
                elseif Rmin > 0 && Rmax == round(vmax(i)*X(end),4)
                    CT(j,i) = 4; % CT(j,i) since we have calculated v_j/v_i in
this case
                end
            end
        end
    end

 end
 FC=array2table(FC,'VariableNames',{'i','j','ratio vi/vj'});
 end

 function [minimum_flux,maximum_flux] = FluxRange(M)
 OPTIONS = optimset('linprog');
 OPTIONS.Display = 'off';
    for j=1:length(M.rxns)
        % set the reaction for which we calculate the range
        M.c = zeros(size(M.c));
        M.c(j) = 1;

        % using linprog
        [~,minimum_flux(j,1)] = linprog(M.c,[],[],M.S,M.b,M.lb,M.ub,OPTIONS);
        [~,maximum_flux(j,1)] = linprog(-M.c,[],[],M.S,M.b,M.lb,M.ub,OPTIONS);
    end

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