```
%% EFM analysis routines starting from CellNetAnalyzer EFM output files

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Script to combine EFM output files from CellNetAnalyzer into one matrix
% for further manipulation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Load each CellNetAnalyzer output file individually and concatenate
load('efm_0.mat')
fullefms = mnet.efms;
clear mnet
load('efm_1.mat')
fullefms = horzcat(fullefms,mnet.efms);
clear mnet
load('efm_2.mat')
fullefms = horzcat(fullefms,mnet.efms);
clear mnet
load('efm_3.mat')
fullefms = horzcat(fullefms,mnet.efms);
clear mnet
load('efm_4.mat')
fullefms = horzcat(fullefms,mnet.efms);
clear mnet
fullefms = transpose(fullefms);
% Save concatenated matrix
save('fullefms.mat','fullefms')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Select and save reaction columns of interest from the full EFM matrix
% to separate variables for faster downstream analysis
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Load entire EFM matrix
savedmodes = load('fullefms.mat');
elmoden = savedmodes.fullefms;

% Assign relevant reaction columns to variables
X_col = 79; % Biomass production
CO2_col = 86; % Carbon dioxide consumed at RuBisCO
O2_col = 231; % Oxygen consumed at RuBisCO
O2_evol = 262; % Oxygen transport reaction
hv_col = 265; % Light
PS1_col = 240; % photosystem II
PS3_col = 242; % photosystem I
ace_col = 267; % Acetate production
ala_col = 274; % Alanine production
eth_col = 271; % Ethanol production
for_col = 263; % Formate production
gly_col = 269; % Glycolate production
lac_col = 270; % Lactate production
pyr_col = 273; % Pyruvate production
suc_col = 272; % Sucrose production
Fecol1 = 1; % Iron requirement, reaction #1
Fecol111 = 111; % Iron requirement, reaction #111
Fecol119 = 119; % Iron requirement, reaction #119
Fecol155 = 155; % Iron requirement, reaction #155
```

```
Fecol156 = 156; % Iron requirement, reaction #156
Fecol240 = 240; % Iron requirement, reaction #240
Fecol241 = 241; % Iron requirement, reaction #241
Fecol242 = 242; % Iron requirement, reaction #242
Fecol243 = 243; % Iron requirement, reaction #243
Fecol277 = 277; % Iron requirement, reaction #277
Fecol281 = 281; % Iron requirement, reaction #281
Fecol282 = 282; % Iron requirement, reaction #282

% Extract reactions from full EFM matrix
X = elmoden(:,X_col);
CO2 = elmoden(:,CO2_col);
O2 = elmoden(:,O2_col);
O2evol = elmoden(:,O2_evol);
hv = elmoden(:,hv_col);
PS1 = elmoden(:,PS1_col);
PS3 = elmoden(:,PS3_col);
ace = elmoden(:,ace_col);
ala = elmoden(:,ala_col);
eth = elmoden(:,eth_col);
form = elmoden(:,for_col);
gly = elmoden(:,gly_col);
lac = elmoden(:,lac_col);
pyr = elmoden(:,pyr_col);
suc = elmoden(:,suc_col);
Fe1 = elmoden(:,Fecol1);
Fe111 = elmoden(:,Fecol111);
Fe119 = elmoden(:,Fecol119);
Fe155 = elmoden(:,Fecol155);
Fe156 = elmoden(:,Fecol156);
Fe240 = elmoden(:,Fecol240);
Fe241 = elmoden(:,Fecol241);
Fe242 = elmoden(:,Fecol242);
Fe243 = elmoden(:,Fecol243);
Fe277 = elmoden(:,Fecol277);
Fe281 = elmoden(:,Fecol281);
Fe282 = elmoden(:,Fecol282);

% Save extracted columns for use in downstream analysis
save('X','X');
save('CO2','CO2');
save('O2','O2');
save('O2evol','O2evol');
save('hv','hv');
save('PS1','PS1');
save('PS3','PS3');
save('ace','ace');
save('ala','ala');
save('eth','eth');
save('form','form');
save('gly','gly');
save('lac','lac');
save('pyr','pyr');
save('suc','suc');
save('Fe1','Fe1');
save('Fe111','Fe111');
```

```
save('Fe119','Fe119');
save('Fe155','Fe155');
save('Fe156','Fe156');
save('Fe240','Fe240');
save('Fe241','Fe241');
save('Fe242','Fe242');
save('Fe243','Fe243');
save('Fe277','Fe277');
save('Fe281','Fe281');
save('Fe282','Fe282');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Analysis routines used to construct each figure
% Analyzes reactions of interest for desired EFMs (e.g. biomass-producing
% EFMs): calculates costs and computes tradeoff curves using minenv2
% function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Figure 2

% Load variables
load('O2evol');
load('hv');
load('X');
load('PS1');
load('PS3');
CmolX = 43.93; % Cmol biomass

% Select biomass-producing EFMs
[r,~,~] = find(X~=0);

% Calculate x and y axes and PSII/I ratio
O2X = O2evol(r)./(X(r)*CmolX); % O2 evolved per biomass
hvX = hv(r)./(X(r)*CmolX); % Photon absorption per biomass
PSII_I = PS1(r)./(PS3(r)/4); % PSII/I photon absorption ratio

% Combine into matrix for manipulation
totalX = [r O2X hvX PSII_I];

% Downsize matrix by interval sampling
O2X_d = O2X(1:100:length(O2X));
hvX_d = hvX(1:100:length(hvX));
PSII_I_d = PSII_I(1:100:length(PSII_I));
total_d = [O2X_d hvX_d PSII_I_d];

% Save x and y values as .csv file for plotting
csvwrite('total_O2X_hvX.csv',total_d);


%% Figure 3

% Load variables
load('O2');
load('CO2');
load('hv');
```

```
load('X');
CmolX = 43.93; % Cmol biomass

% Select biomass-producing EFMs
[r,~,~] = find(X~=0);

% Calculate x and y axes
O2CO2 = O2(r)./CO2(r); % O2/CO2 competition
hvX = hv(r)./(X(r)*CmolX); % Photon absorption per biomass

% Combine into matrix for manipulation
totalX = [r O2CO2 hvX];

% Downsize matrix by interval sampling
O2CO2_d = O2CO2(1:100:length(O2CO2));
hvX_d = hvX(1:100:length(hvX));
total_d = [O2CO2_d hvX_d];

% Find tradeoff curve
env = minenv2(totalX,2,3);

% Save x and y values as .csv file for plotting, save full tradeoff curve
EFMs
csvwrite('total.csv',total_d);
csvwrite('total_env.csv',env);
total_env = env;
save('total_env','total_env');


%% Figure S4A

% Load variables
savedmodes = load('fullefms.mat');
elmoden = savedmodes.fullefms;
load('O2');
load('CO2');
load('X');
load('N'); % Contains nitrogen content of enzymes for each enzymatic reaction

% Select biomass-producing EFMs
[r,~,~] = find(X~=0);

% Calculate y axis
O2CO2 = O2(r)./CO2(r); % O2/CO2 competition

% Calculate nitrogen requirement for each EFM
efms = elmoden(r,:);
clear elmoden savedmodes
for i = 1:size(efms,1)
    for j = 1:size(efms,2)
        if efms(i,j) ~= 0
            efms(i,j) = 1; % Translate matrix into binary 0's/1's
        end
    end
```

```
end
Nreq = efms*N; % Collapses down to give N atom requirement for each mode

% Combine into matrix for manipulation
totalX = [r O2CO2 Nreq];

% Downsize matrix by interval sampling
O2CO2_d = O2CO2(1:100:length(O2CO2));
Nreq_d = Nreq(1:100:length(Nreq));
total_d = [O2CO2_d Nreq_d];

% Find tradeoff curve
env = minenv2(totalX,2,3);

% Save x and y values as .csv file for plotting, save full tradeoff curve
EFMs
csvwrite('total_N_binary.csv',total_d);
csvwrite('total_env_N_binary.csv',env);
total_env_N_bin = env;
save('total_env_N_bin');


%% Figure S4B

% Load variables
load('O2');
load('CO2');
load('X');
load('Fe1');
load('Fe111');
load('Fe119');
load('Fe155');
load('Fe156');
load('Fe240');
load('Fe241');
load('Fe242');
load('Fe243');
load('Fe277');
load('Fe281');
load('Fe282');
Fecols = [Fe1 Fe111 Fe119 Fe155 Fe156 Fe240 Fe241 Fe242 Fe243 Fe277 Fe281
Fe282];
clear Fe1 Fe111 Fe119 Fe155 Fe156 Fe240 Fe241 Fe242 Fe243 Fe277 Fe281 Fe282
Fe = [3;4;4;4;4;3;5;12;2;4;11;4]; % Iron requirement for selected enzymatic
reactions (photosynthesis and central metabolism)

% Select biomass-producing EFMs
[r,~,~] = find(X~=0);

% Calculate y axis
O2CO2 = O2(r)./CO2(r); % O2/CO2 competition

% Calculate iron requirement for each EFM
Fecols_X = Fecols(r,:);
for i = 1:size(Fecols_X,1)
```

```
    for j = 1:size(Fecols_X,2)
        if Fecols_X(i,j) ~= 0
            Fecols_X(i,j) = 1;
        end
    end
end
Fereq = Fecols_X*Fe; % Iron requirement

% Combine into matrix for manipulation
totalX = [r O2CO2 Fereq];

% Downsize matrix by interval sampling
O2CO2_d = O2CO2(1:100:length(O2CO2));
Fereq_d = Fereq(1:100:length(Fereq));
total_d = [O2CO2_d Fereq_d];

% Find tradeoff curve
env = minenv2(totalX,2,3);

% Save x and y values as .csv file for plotting, save full tradeoff curve
EFMs
csvwrite('total_Fe_binary.csv',total_d);
csvwrite('total_env_Fe_binary.csv',env);
total_env_Fe_bin = env;
save('total_env_Fe_bin');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Tradeoff curve analysis algorithm
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function env = minenv2(M2,abscol,ordcol)
% M2 = matrix of EFMs and/or costs associated with EFMs
% abscol = x-value cost (independent variable)
% ordcol = y-value cost (dependent variable)

% Set niter.
niter = 100;

% Sort modes according to abscol values in ascending order.
relM2 = sortrows(M2,abscol);
clear M2;

%% ------------------------------------------------------------------------ %%
%% Find initial modes for all potential envelopes.

% Collect all elementary modes which maximize the abscissa.  This finds the
% maximum O2:CO2 value possible, or the upper right edge of the curve.
i = size(relM2,1);
while relM2(i,abscol) == relM2(size(relM2,1),abscol);
    i = i - 1;
end
tempa = relM2(i+1:size(relM2,1),:);

%% ------------------------------------------------------------------------ %%
%% Calculate envelopes.
```

```
% Check slopes between last mode chosen for an envelope and all modes with
% a smaller abscissa.  Collect those modes with the largest positive slope
% and add them to the envelope (farthest last).  Iterate until either the
% number of cycles exceeds specifications or no positive slopes are found.
% Positive slopes are used because the algorithm works from the right
% uppermost point of the curve down toward the left bottommost point of the
% curve.
i = 1;
row = 0;
while i <= length(ordcol)
    pos = 1;
    tempa = sortrows(tempa,ordcol(i));
    j = 0;
    while size(tempa,1) >= j + 1 && tempa(j+1,ordcol(i)) ==
tempa(1,ordcol(i));
        j = j + 1;
    end
    env(row+1:row+j,:) = tempa(1:j,:);
    row = row + j;
    pos = pos + 1;
    j = 1;
    while j <= niter
        if j == 1
            relrelM2 = sortrows(relM2,ordcol(i));
        else
            relrelM2 = sortrows(relrelM2,ordcol(i));
        end
        k = 0;
        while k+1 <= size(relrelM2,1) && relrelM2(k+1,ordcol(i)) <
env(row,ordcol(i))
            k = k + 1;
        end
        if k == 0
            break
        end
        relrelM2 = relrelM2(1:k,:);
        slope = (relrelM2(:,ordcol(i)) - env(row,ordcol(i))) ./
(relrelM2(:,abscol) - env(row,abscol));
        relrelM2 = [relrelM2 slope];

        relrelM2 = sortrows(relrelM2,-size(relrelM2,2)); %Sort by descending
slope to find the largest positive slope (working down and backwards along
the curve)
        k = 0;
        while k+1 <= size(relrelM2,1) && relrelM2(k+1,size(relrelM2,2)) ==
relrelM2(1,size(relrelM2,2))
            k = k + 1;
        end
        if k ~= 0
            relrelM2 = relrelM2(:,1:size(relrelM2,2)-1);
            env(row+1:row+k,:) = sortrows(relrelM2(1:k,:),abscol);
            row = row + k;
            pos = pos + 1;
        end
        j = j + 1;
```

```
    end
    fprintf('\nMinimization envelope %d is complete.\n',i)
    i = i + 1;
end
end
```