Appendix: Code

June 17, 2021

Reproducibility: Although, all the codes and files are attached to this document, the best way to reproduce the results in this article is to acces the Github repo and read through the README file for better understanding of the usability of the functions and the pipelines to be used.

Github Repository Link given below:

`https://github.com/Souvadra/CH248_Project`

Topo file and RACIPE simulations:

Circuit Topology

Source	Target	Type
Start	Start	2
Start	SK	1
SK	SK	2
SK	Ste9	2
SK	Rum1	2
Ste9	Cdc2/Cdc13	2
Cdc2/Cdc13	Ste9	2
Ste9	Cdc2/Cdc13*	2
Cdc2/Cdc13*	Ste9	2
Rum1	Cdc2/Cdc13	2
Cdc2/Cdc13	Rum1	2
Rum1	Cdc2/Cdc13*	2
Cdc2/Cdc13*	Rum1	2
Cdc2/Cdc13	Wee1/Mik1	2
Cdc2/Cdc13	Cdc25	1
PP	PP	2
PP	Ste9	1
PP	Rum1	1
PP	Wee1/Mik1	1
PP	Cdc25	2
Slp1	PP	1
Slp1	Slp1	2
Slp1	Cdc2/Cdc13	2
Slp1	Cdc2/Cdc13*	2
Cdc2/Cdc13*	Slp1	1
Wee1/Mik1	Cdc2/Cdc13*	2
Cdc25	Cdc2/Cdc13*	1

Running RACIPE Download and compile RACIPE, copy the write the .topo file of the circuit in the same directory of RACIPE and then type:

```
./RACIPE <file>.topo -num_paras <number_of_parameter_sets> -num_ode
<number_of_initial_conditions_for_each_parameter_set> -threads <number_of_processors>
```

G/K Normlalization, Z-normalization:

```
function GK_normalization(path,components_num,external_signal)

%% Reading the paths and making string array of solution files ------------
% and parameters file separately ----------------------------------------

path = strrep(path,'\','/');
x = strcat(path,"/*solution_*.dat");
y = strcat(path,"/*parameters.dat");

F = dir(x);
F2 = dir(y);

%%-------------------------------------------------------------------------
%% Replacing backward slash with forward slashes  for parameters file -----

for i = 1:length(F2)

    s2 = strcat(F2(i).folder,"/",F2(i).name);
    str2(i,1) = strrep(s2,'\','/');
```

```
end

%%-------------------------------------------------------------------------
%% Reading the parameters file, remove all coloumns after the degradation -
% rate values, divide production rates coloumn by degradation rates coloumn
% and then remove all coloumns after the G/K values ----------------------

%external_signal = str2double(external_signal); %Souvadra's addition
%components_num = str2double(components_num);   %Souvadra's addition

F1 = dlmread(str2(1));
F1(:,2+2*(components_num+external_signal)+1:end) = [];

 for j = 3:(3+components_num-1)

    F1(:,j) = F1(:,j)./F1(:,j+components_num+external_signal);

 end


F1(:,2+components_num+1:end) = [];

%%-------------------------------------------------------------------------
%% Replacing backward slash with forward slashes  for solution files ------

str = strings(length(F),1);

for i = 1:length(F)

    s = strcat(F(i).folder,"/",F(i).name);
    str(i,1) = strrep(s,'\','/');

end

%%-------------------------------------------------------------------------
%% Reading the solution files into matrix and performing G/K normalization
% and further the concatenation of all solutions and calculating the ------
% z-scores for plotting the scatter diagram -------------------------------

Mn = zeros(1,components_num);

for i = 1:length(str)

    A = dlmread(str(i));
    B = A(:,1);
    A = 2.^A;
    a = size(A);

    for k = 1:a(1,1)

       for m = 1:length(F1)

       if B(k,1) == F1(m,1)

                for j = 3:components_num + external_signal:a(1,2)

                    for l = 1:components_num

                        A(k,j+l-1) = A(k,j+l-1)./F1(k,2+l);

                    end

                end
       end

       end

    end

    A = log2(A);
    A(:,1) = B;
    newstr = split(str(i,1),"_");
    size(newstr,1);
    new = strings(1);
    for i = 1:size(newstr,1)-1
        new = strcat(new,newstr(i,1),"_");
    end
```

```
        new = strcat(new,"gk_",newstr(size(newstr,1),1));
        dlmwrite(new,A,'delimiter','\t');
    end
    %%-------------------------------------------------------------------------
    end
```

Stability State Counting:

Function 1

```
function universal_stability_state_counter(p1,p2,p3,name)
%{
This function can be used to easily plot the numbers of different
stability states in a given ciruit, using the data from the triplicates and
plots the results in the form of a .fig file. This function is made to
handle larger RACIPE simulations will 20th- and 30th-stable solutions and more general
than that of standard 10th-stable solutions used otherwise.
%}
%% Call the helper functinos
    file1 = p1 ;
    file2 = p2 ;
    file3 = p3 ;
    [val1] = stateCounter(file1);
    [val2] = stateCounter(file2);
    [val3] = stateCounter(file3);
    % let me first finish the helper functions
%% Do the required calculatios
    allStates = unique([val1(:,2); val2(:,2); val3(:,2)]);
    val = zeros(length(allStates),1); % val is actually my y axis
    x = zeros(length(allStates),1);
    err = zeros(length(allStates),1);
    i = 0;
    for currState = allStates'
        i = i + 1;
        z = [];
        participants = 0;

        for j1=1:size(val1,1)
            if val1(j1,2) == currState
                z = [z; val1(j1,1)];
                participants = participants + 1;
            end
        end

        for j2=1:size(val2,1)
            if val2(j2,2) == currState
                z = [z; val2(j2,1)];
                participants = participants + 1;
            end
        end

        for j3=1:size(val3,1)
            if val3(j3,2) == currState
                z = [z; val3(j3,1)];
                participants = participants + 1;
            end
        end

        val(i,1) = sum(z)/participants;
        err(i,1) = std(z);
        x(i,1) = int8(currState);
    end

%% Its time to plot the data
    bar(x,val); hold on;
    er = errorbar(x,val,err,err);
    er.Color = [0 0 0];
    er.LineStyle = 'none';

    ylabel('Number of the stability states');
    titl = ['Circuit topology: ',name];
    title(titl);
    saveName = [name,'_stability_state_counts_full.fig'];
    savefig(saveName);
end
```

Function 2

```
%{
    This function outputs the percentage of RACIPE solutions belonging each
    stability state (obviously incorporating the triplicates) in the form of
    a .xls file.
%}
function output_table = MakeStabilityStateCounter(path,name)
%% Call the helper functions
vec1 = stabilityStateCounter(path + "/1");
vec2 = stabilityStateCounter(path + "/2");
vec3 = stabilityStateCounter(path + "/3");
%% take mean and std-dev
vec = [vec1 vec2, vec3];
vec = vec';
Mean = mean(vec);
Std = std(vec);
Mean = Mean';
Std = Std';
output_table = [Mean Std];
%% time for plotting the function
x_axis = ["mono", "bi", "tri", "other"];
x = 1:4;

figure
bar(x,Mean); hold on;
er = errorbar(x,Mean,Std,Std);
er.Color = [0 0 0];
er.LineStyle = 'none';

set(gca,'xticklabel',x_axis)
ylabel('Percentage Stability States');
Name = "Circuit topology: "+ string(name);
title(Name);

savefig(string(name) + "-stability-state-percentage.fig");
Name = string(name) + "stability-state-percentage" + ".xls";
writematrix(output_table,Name)
end
```

Function 3

```
function output_table = stabilityStateCounter(path)
%% Reading the data and arranging them properly
path = strrep(path,'\','/');
x = strcat(path,"/*solution_gk_*.dat");
solution_path_dir = dir(x);
solution_paths = strings(length(solution_path_dir),1);
for i = 1:length(solution_path_dir)

    s = strcat(solution_path_dir(i).folder,"/",solution_path_dir(i).name);
    solution_paths(i,1) = strrep(s,'\','/');

end
xxx = char(solution_paths(2));
while xxx(length(xxx)-5) == '1'
    solution_paths = [solution_paths; solution_paths(2)];
    solution_paths(2) = [];
    xxx = char(solution_paths(2));
end
xxx = char(solution_paths(3));
if xxx(length(xxx)-5) == '2'
    solution_paths = [solution_paths; solution_paths(3)];
    solution_paths(3) = [];
end
%% Select each stability state file and work the calculation separetely
regex_pattern = '\_\d+';
output_table = zeros(4,1);
for i = 1:length(solution_paths)
    match = regexp(solution_paths(i), regex_pattern, 'match');
    match = strsplit(match, '_');
    match = match(2);
    stability_state =  str2num(match);

    solution_values_all = dlmread(solution_paths(i));
    total_number = size(solution_values_all,1);

    if stability_state < 4
        output_table(stability_state,1) = total_number;
    else
```

```
        output_table(4) = output_table(4) + total_number;
    end
end
output_table = output_table./sum(output_table);
end
```

Function 4

```
function output_table = stabilityStateCounter(path)
%% Reading the data and arranging them properly
path = strrep(path,'\','/');
x = strcat(path,"/*solution_gk_*.dat");
solution_path_dir = dir(x);
solution_paths = strings(length(solution_path_dir),1);
for i = 1:length(solution_path_dir)

    s = strcat(solution_path_dir(i).folder,"/",solution_path_dir(i).name);
    solution_paths(i,1) = strrep(s,'\','/');

end
xxx = char(solution_paths(2));
while xxx(length(xxx)-5) == '1'
    solution_paths = [solution_paths; solution_paths(2)];
    solution_paths(2) = [];
    xxx = char(solution_paths(2));
end
xxx = char(solution_paths(3));
if xxx(length(xxx)-5) == '2'
    solution_paths = [solution_paths; solution_paths(3)];
    solution_paths(3) = [];
end
%% Select each stability state file and work the calculation separetely
regex_pattern = '\_\d+';
output_table = zeros(4,1);
for i = 1:length(solution_paths)
    match = regexp(solution_paths(i), regex_pattern, 'match');
    match = strsplit(match, '_');
    match = match(2);
    stability_state =  str2num(match);

    solution_values_all = dlmread(solution_paths(i));
    total_number = size(solution_values_all,1);

    if stability_state < 4
        output_table(stability_state,1) = total_number;
    else
        output_table(4) = output_table(4) + total_number;
    end
end
output_table = output_table./sum(output_table);
end
```

State Frequency Calculation:

Function 1

```
function [y, Fn1, Fn] = err_bar_maker(path,sol_num,components_num,ex_sig)

%% Reading the data into matrices and arraging them -----------------------

C = dlmread(path);
C(:,1:2) = [];

for i = 1:sol_num
    C(:,(i*components_num)+1:(i*components_num)+ex_sig) = [];
end

Cm = mean(C);
Csd = std(C);
Cn = (C - Cm)./(Csd);

a = size(Cn);
Cn = Cn > 0 ;

%%-------------------------------------------------------------------------
%% Determining the phase of the solution obtained ---------------
```

```
% A1 = [0 0 0];
% A2 = [1 0 0];
% A3 = [0 1 0];
% A4 = [0 0 1];
% A5 = [1 1 0];
% A6 = [1 0 1];
% A7 = [0 1 1];
% A8 = [1 1 1];

v1 = repmat(0,components_num,1);
v2 = repmat(1,components_num,1);
v = horzcat(v1',v2');

n_sh = components_num;
lim_sh = 1;
for ii=(1:n_sh)
    lim_sh = lim_sh + (2^(ii-1));
end

A3 = zeros(1,n_sh);
for ii=(1:lim_sh-1)
    c_sh = ii;
    vector_sh = [];
    while c_sh > 0
        if c_sh == 1
            vector_sh = [vector_sh, c_sh];
            c_sh = 0;
        else
            rem_sh = mod(c_sh,2);
            vector_sh = [vector_sh, rem_sh];
            c_sh = (c_sh - rem_sh) / 2;
        end
    end
    l_sh = n_sh - length(vector_sh);
    for jjj=(1:l_sh)
        vector_sh = [vector_sh, 0];
    end
    vector_sh = fliplr(vector_sh);
    A3 = [A3 ; vector_sh];
end

 a3 = size(A3);

Dn = zeros(a(1,1),components_num*sol_num);

for i=1:components_num:components_num*sol_num
    for j = 1:a(1,1)
        for k = 1:a3(1,1)
            if isequal(Cn(j,i:i+components_num-1),A3(k,:))
                Dn(j,i)=k;
            end
        end
    end
end

%%-------------------------------------------------------------------------
%% Sorting the solutions for ease of counting ----------------------------

Zn = zeros(a(1,1),sol_num);

for i = 1:sol_num
    Zn(:,i) = Dn(:,((components_num*(i-1)+1)));
end

Zn = sort(Zn,2);
Zn = sortrows(Zn);
%%-------------------------------------------------------------------------
%% Explicitly putting the naming system (Binary-ish) - Souvadra
naming = [];
L = size(A3);
L = L(1,1);
B = size(A3);
B = B(1,2);
for i=(1:L)
    sSH = "";
    for j=(1:B)
        sSH = sSH + A3(i,j);
    end
```

```
    naming = [naming; sSH];
end

Zn_naming = [];
BB = size(Zn);
BB = BB(1,2);
for i=(1:length(Zn))
    eSH = [];
    for j=(1:BB)
        aSH = naming(Zn(i,j));
        eSH = [eSH aSH];
    end
    Zn_naming = [Zn_naming; eSH];
end

BBB = size(Zn_naming);
BBB = BBB(1,2);
for i=(1:length(Zn_naming))
    for j=(2:BBB)
    Zn_naming(i,1) = Zn_naming(i,1)+Zn_naming(i,j);
    end
end
Zn_naming = Zn_naming(:,1);
%Fn_naming = unique(Zn_naming);

%%-------------------------------------------------------------------------
%% Making the X-tick labels ---------------------------------------------

% writing the matrix to txt file, then remove delimiters followed by
% reading the txt file so that the rows are horizontally combines i.e. the
% three coloumn single digit elements are converted to a single coloumn
% three digit number. Finally the vector is converted to a string array.

dlmwrite('data.txt',Zn,'delimiter','');
Fn = dlmread('data.txt');
Var_SH = Fn;
delete data.txt;
Fn = [Fn, Zn_naming]; % Souvadra's addition
Fn = unique(Fn,'rows');
f = size(Fn);
Fn1 = Fn(:,1); % Souvadra's addition

%%-------------------------------------------------------------------------
%% Souvadra's alternative to the happeing issue
y = zeros(f(1,1),1);
for i=(1:length(Var_SH))
    for j=(1:f(1,1))
        if Var_SH(i,1) == double(Fn(j,1))
            y(j,1) = y(j,1) + 1;
        end
    end
end
y = y ./ sum(y); % giving percentage result rather than the full value
%%-------------------------------------------------------------------------
end
```

Function 2

```
function matrix_sh = make_an_errorbar_conditions_apply(p1,p2,p3,sol_num,
components_num,ext_signals_num,condition,name)

%% Reading all the solution files into matrices ---------------------------

file1 = p1 ;
file2 = p2 ;
file3 = p3 ;

[X1, F1, F1sh] = err_bar_maker(file1,sol_num,components_num,ext_signals_num);
[X2, F2, F2sh] = err_bar_maker(file2,sol_num,components_num,ext_signals_num);
[X3, F3, F3sh] = err_bar_maker(file3,sol_num,components_num,ext_signals_num);

%%-------------------------------------------------------------------------
%% Making all data vectors of same length --------------------------------

maxlen = [length(X1) length(X2) length(X3)];
maxlen = max(maxlen);
X1(end+1:maxlen) = 0;
X2(end+1:maxlen) = 0;
```

```
X3(end+1:maxlen) = 0;
F1(end+1:maxlen) = "0";
F2(end+1:maxlen) = "0";
F3(end+1:maxlen) = "0";

%%-------------------------------------------------------------------------
%% Making the Categories ------------------------------------------------

C = unique([F1;F2;F3]);
C = double(C);
c = size(C);

%%-------------------------------------------------------------------------
%% Make the data from different simulations match based on the category ---

data = zeros(c(1,1),3);

for i = 1:c(1,1)
    for j = 1:maxlen

        if C(i,1)==double(F1(j,1))
            data(i,1) = X1(j,1);
        end

    end
end
for i = 1:c(1,1)
    for j = 1:maxlen

        if C(i,1)==double(F2(j,1))
            data(i,2) = X2(j,1);
        end

    end
end
for i = 1:c(1,1)
    for j = 1:maxlen

        if C(i,1)==double(F3(j,1))
            data(i,3) = X3(j,1);
        end

    end
end

data_mod = data;

%%-------------------------------------------------------------------------
%% Remove the rows containing all zeros from the C and data matrix --------

a = sum(data,2);
a1 = zeros(length(a),1);

for i = 1:c(1,1)

    if a(i) == 0
        a1(i) = i;
    end

end

a1 = a1(a1 ~= 0);
data_mod = horzcat(data_mod,C);
data_mod(a1,:) = [] ;

data_mod = sortrows(data_mod,'descend');

%%-------------------------------------------------------------------------
%% Calculation errors required for barplot -------------------------------

data_final = data_mod(:,1:3)';
mean_data = mean(data_final);
std_data = std(data_final);

cnn = string(data_mod(:,4));

%%-------------------------------------------------------------------------
%% Conditioning of the data ----------------------------------------------
```

```
% required_indexes = find(mean_data>50);
B = mean_data > condition ;
new_mean_data = mean_data(B);
new_std_data = std_data(B);
new_cnn = cnn(B);


%%-------------------------------------------------------------------------
%% Plotting the bar plot with errorbars ----------------------------------
% Souvadra is commentinf this block of code
cn = categorical(new_cnn);

grid on
bar(cn,new_mean_data);
hold on

er = errorbar(cn,new_mean_data,new_std_data);
er.Color = [0 0 0];
er.LineStyle = 'none';

hold off

%%-------------------------------------------------------------------------
%% Souvadra's modificaiton, Let's not plot the data and store it instead in
%% .csv file
%cn = categorical(new_cnn)
QSH = [F1sh; F2sh; F3sh];
QSH = unique(QSH,'rows');
Map = containers.Map(QSH(:,1),QSH(:,2));
nameSH = strings(length(new_mean_data),1);
for i=(1:length(new_mean_data)) % THis was initially 'length(QSH)'
    variable = {new_cnn(i,1)};
    mappedTo = string(values(Map,variable));
    mappedTo = "<" + mappedTo + ">";
    %nameSH = [nameSH ; mappedTo];
    nameSH(i,1) = mappedTo;
end

Name = name + ".xls";
matrix_sh = [new_cnn, nameSH, new_mean_data', new_std_data']
writematrix(matrix_sh,Name)
%%-------------------------------------------------------------------------
end
```

Plotting the Dynamics of the network:

Function 1

```
% A simple representation of Hill equation
function H = hill(X,X0,lambda,n)
% H+
H1 = ((X/X0)^n)/(1+(X/X0)^n);
% H-
H2 = 1/(1+(X/X0)^n);

% Hill function =  H- plus (lambda)* H+
H = H2 + (lambda)*H1;
if lambda > 1
     H=H/lambda;
end

end

% function H = hill(X,X0,lambda,n)
% H = lambda + (1.0 - lambda) * (1.0/(1.0 + (X/X0)^n));
% if lambda > 1
%     H = H / lambda;
% end
%
% if H < 0
%     H
% end
% end
```

Function 2

```
function [prs_file_info, parameters] = parameter_generator(path)
```

```
%% Reading the topo file and generating the variables ----------------------

path = strrep(path,'\','/');
path = strcat(path,"/*.prs");
file_dir = dir(path);
prs_path = strcat(file_dir(1).folder,"/",file_dir(1).name);
prs_path = strrep(prs_path,'\','/');
prs_file_info = tdfread(prs_path);

parameters = prs_file_info.Parameter ;

%%-------------------------------------------------------------------------
```

Function 3

```
function parameter_sets_for_simulating = parameter_set_segregator(path,
components_num,external_signal,sol_num,categories)

[topo_file_info, parameter_names] = parameter_generator(path);

path = strrep(path,'\','/');
x = strcat(path,"/*solution_gk_*.dat");
y = strcat(path,"/*parameters.dat");

solution_path_dir = dir(x);
parameter_value_file_dir = dir(y);

paramater_value_path = strcat(parameter_value_file_dir(1).folder,"/",
parameter_value_file_dir(1).name);
paramater_value_path = strrep(paramater_value_path,'\','/');
parameter_values = readtable(paramater_value_path);

%% The paths copied will have backward slashes so have to be replaced with
% backward slashes ----------------------------------------------------------

solution_paths = strings(length(solution_path_dir),1);

for i = 1:length(solution_path_dir)

    s = strcat(solution_path_dir(i).folder,"/",solution_path_dir(i).name);
    solution_paths(i,1) = strrep(s,'\','/');

end

%%-------------------------------------------------------------------------

%% reordering the solution_paths to keep solution_10 at last rather than at the
second place
% -- Souvadra
xxx = char(solution_paths(2));
while xxx(length(xxx)-5) == '1'
    solution_paths = [solution_paths; solution_paths(2)];
    solution_paths(2) = [];
    xxx = char(solution_paths(2));
end
xxx = char(solution_paths(3));
if xxx(length(xxx)-5) == '2'
    solution_paths = [solution_paths; solution_paths(3)];
    solution_paths(3) = [];
end
%%

if sol_num == 1
    solution_values_all = dlmread(solution_paths(sol_num,1));
else, solution_values_all = dlmread(solution_paths(sol_num,1));
end

solution_values_all_copy = solution_values_all;
solution_values_all(:,1:2) = [];

for i = 1:sol_num

    solution_values_all(:,(i*components_num)+1:(i*components_num)+external_signal) = [];

end

solution_values_all_mean = mean(solution_values_all);
```

```
solution_values_all_sd = std(solution_values_all);
solution_values_all_normalized = (solution_values_all -
solution_values_all_mean)./(solution_values_all_sd);

a = size(solution_values_all_normalized);

%%-------------------------------------------------------------------------
%% Making the elements of the matrix binary to represent the states -------

% for i = 1:components_num*sol_num
%     for j = 1:a(1,1)
%         if solution_values_all_normalized(j,i) < 0
%             solution_values_all_normalized(j,i) = 0;
%         else, solution_values_all_normalized(j,i) = 1;
%         end
%     end
% end

solution_values_all_normalized = solution_values_all_normalized > 0;

%%-------------------------------------------------------------------------
%% Determining the phase of the solution obtained ---------------

v1 = repmat(0,components_num,1);
v2 = repmat(1,components_num,1);
v = horzcat(v1',v2');

% A1 = nchoosek(v,components_num);
% A1 = unique(A1,'rows');
% A1size = size(A1);

% This code-block is written by Souvadra
n_sh = components_num;
lim_sh = 1;
for ii=(1:n_sh)
    lim_sh = lim_sh + (2^(ii-1));
end

A3 = zeros(1,n_sh);
for ii=(1:lim_sh-1)
    c_sh = ii;
    vector_sh = [];
    while c_sh > 0
        if c_sh == 1
            vector_sh = [vector_sh, c_sh];
            c_sh = 0;
        else
            rem_sh = mod(c_sh,2);
            vector_sh = [vector_sh, rem_sh];
            c_sh = (c_sh - rem_sh) / 2;
        end
    end
    l_sh = n_sh - length(vector_sh);
    for jjj=(1:l_sh)
        vector_sh = [vector_sh, 0];
    end
    vector_sh = fliplr(vector_sh);
    A3 = [A3 ; vector_sh];
end

% A3 = zeros(1,components_num);
%
% for i = 1:A1size(1,1)
%
%     A2 = perms(A1(i,:));
%     A2 = unique(A2,'rows');
%     A3 = vertcat(A3,A2);
%
% end
%
% A3(1,:) = [];
a3 = size(A3);

solution_values_all_categorized = zeros(a(1,1),components_num*sol_num);

for i=1:components_num:components_num*sol_num
    for j = 1:a(1,1)
        for k = 1:a3(1,1)
```

```
            if isequal(solution_values_all_normalized(j,i:i+components_num-1),A3(k,:))
                solution_values_all_categorized(j,i)=k;
            end
        end
    end
end

clear A1 A1size A2 A3 a2 a3 v v1 v2 x y i j k s

%%-------------------------------------------------------------------------
%% Sorting the solutions for ease of counting ----------------------------

solutions_all_categorized_no_null = zeros(a(1,1),sol_num);

for i = 1:sol_num

    solutions_all_categorized_no_null(:,i) = solution_values_all_categorized(:,((
        components_num*(i-1)+1)));

end

% Zn = sort(Zn,2);
% Zn = sortrows(Zn);

clear i

%%-------------------------------------------------------------------------
%% Making the X-tick labels ----------------------------------------------

% writing the matrix to txt file, then remove delimiters followed by
% reading the txt file so that the rows are horizontally combines i.e. the
% three coloumn single digit elements are converted to a single coloumn
% three digit number. Finally the vector is converted to a string array.

dlmwrite('data.txt',solutions_all_categorized_no_null,'delimiter','');
solutions_all_categorized_final = dlmread('data.txt');
delete data.txt;
% Fn = unique(Fn);
% f = size(Fn);
% Fnn = string(Fn);

%%-------------------------------------------------------------------------

solutions_all_categorized_final = horzcat(solutions_all_categorized_final,
solution_values_all_copy(:,1));

if sol_num == 1
    for i = 1:size(solutions_all_categorized_final,1)
        if solutions_all_categorized_final(i,1)==categories(1,1)
            solution_indices_required(i) = solutions_all_categorized_final(i,2);
        end
    end
% if sol_num == 1
%     for i = 1:size(solutions_all_categorized_final,1)
%         if solutions_all_categorized_final(i,1)==categories(1,1)
%             solution_indices_required(i) = solutions_all_categorized_final(i,2);
%         end
%     end
elseif sol_num == 2
    for i = 1:size(solutions_all_categorized_final,1)
        if solutions_all_categorized_final(i,1)==categories(1,1) ||
            solutions_all_categorized_final(i,1)==categories(1,2)
            solution_indices_required(i) = solutions_all_categorized_final(i,2);
        end
    end
elseif sol_num == 3
    for i = 1:size(solutions_all_categorized_final,1)
        if solutions_all_categorized_final(i,1)==categories(1,1) ||
            solutions_all_categorized_final(i,1)==categories(1,2) ||
                solutions_all_categorized_final(i,1)==categories(1,3) ||
                    solutions_all_categorized_final(i,1)==categories(1,4) ||
                        solutions_all_categorized_final(i,1)==categories(1,5) ||
                            solutions_all_categorized_final(i,1)==categories(1,6)
            solution_indices_required(i) = solutions_all_categorized_final(i,2);
        end
    end
end
```

```
solution_indices_required = solution_indices_required';
solution_indices_required = nonzeros(solution_indices_required);

%%-------------------------------------------------------------------------

% parameters = zeros(size(solution_indices_required,1)+1,size(parameter_names,2));

for i = 1:size(parameter_names,1)
    parameters(1,i) = convertCharsToStrings(parameter_names(i,1:size(parameter_names,2)));
end

parameter_values_newfilenames = cellstr(parameters);

parameter_values.Properties.VariableNames = ["S_no" "States_number"
parameter_values_newfilenames];

clear i

%%-------------------------------------------------------------------------

% parameter_sets_for_simulating = table(size(solution_indices_required,1),
size(parameter_values,2));
xxx = parameter_values.S_no;
%LLL = size(parameter_values);
for i = 1:numel(solution_indices_required)
    for j = 1:numel(parameter_values.Prod_of_A) % Souvadra's commenting

    %for j = 1:LLL(1,1)
     if solution_indices_required(i,1) == xxx(j)
         parameter_sets_for_simulating(i,:)=parameter_values(j,:);
     end

   end
end

%%-------------------------------------------------------------------------

end
```

Function 4

```
function dydt = dynamics_simulation(t,y,parameter_set)

dydt = zeros(10,1);
Start = y(1);
SK = y(2);
Ste9 = y(3);
Rum1 = y(4);
Cdc2byCdc13 = y(5);
Cdc2byCdc13Star = y(6);
Wee1byMik1 = y(7);
Cdc25 = y(8);
PP = y(9);
Slp1 = y(10);


ga = parameter_set.("Prod_of_Start               ");
gb = parameter_set.("Prod_of_SK                  ");
gc = parameter_set.("Prod_of_Ste9                ");
gd = parameter_set.("Prod_of_Rum1                ");
ge = parameter_set.("Prod_of_Cdc2/Cdc13          ");
gf = parameter_set.("Prod_of_Cdc2/Cdc13*         ");
gg = parameter_set.("Prod_of_Wee1/Mik1           ");
gh = parameter_set.("Prod_of_Cdc25               ");
gi = parameter_set.("Prod_of_PP                  ");
gj = parameter_set.("Prod_of_Slp1                ");

ka = parameter_set.("Deg_of_Start                ");
kb = parameter_set.("Deg_of_SK                   ");
kc = parameter_set.("Deg_of_Ste9                 ");
kd = parameter_set.("Deg_of_Rum1                 ");
ke = parameter_set.("Deg_of_Cdc2/Cdc13           ");
kf = parameter_set.("Deg_of_Cdc2/Cdc13*          ");
kg = parameter_set.("Deg_of_Wee1/Mik1            ");
kh = parameter_set.("Deg_of_Cdc25                ");
ki = parameter_set.("Deg_of_PP                   ");
kj = parameter_set.("Deg_of_Slp1                 ");
```

```
H_Start_Start = hill(Start, parameter_set.("Trd_of_StartToStart          "),
parameter_set.("Inh_of_StartToStart          "),
parameter_set.("Num_of_StartToStart          "));

H_SK_SK = hill(SK, parameter_set.("Trd_of_SKToSK                "),
parameter_set.("Inh_of_SKToSK                "),
parameter_set.("Num_of_SKToSK                "));
H_Start_SK = hill(Start, parameter_set.("Trd_of_StartToSK             "),
parameter_set.("Act_of_StartToSK             "),
parameter_set.("Num_of_StartToSK             "));

H_Cdc2byCdc13Star_Ste9 = hill(Cdc2byCdc13Star,
parameter_set.("Trd_of_Cdc2/Cdc13*ToSte9     "),
parameter_set.("Inh_of_Cdc2/Cdc13*ToSte9     "),
parameter_set.("Num_of_Cdc2/Cdc13*ToSte9     "));
H_SK_Ste9 = hill(SK, parameter_set.("Trd_of_SKToSte9              "),
parameter_set.("Inh_of_SKToSte9              "),
parameter_set.("Num_of_SKToSte9              "));
H_Cdc2byCdc13_Ste9 = hill(Cdc2byCdc13,
parameter_set.("Trd_of_Cdc2/Cdc13ToSte9      "),
parameter_set.("Inh_of_Cdc2/Cdc13ToSte9      "),
parameter_set.("Num_of_Cdc2/Cdc13ToSte9      "));
H_PP_Ste9 = hill(PP, parameter_set.("Trd_of_PPToSte9              "),
parameter_set.("Act_of_PPToSte9              "),
parameter_set.("Num_of_PPToSte9              "));

H_SK_Rum1 = hill(SK, parameter_set.("Trd_of_SKToRum1              "),
parameter_set.("Inh_of_SKToRum1              "),
parameter_set.("Num_of_SKToRum1              "));
H_Cdc2byCdc13_Rum1 = hill(Cdc2byCdc13,
parameter_set.("Trd_of_Cdc2/Cdc13ToRum1      "),
parameter_set.("Inh_of_Cdc2/Cdc13ToRum1      "),
parameter_set.("Num_of_Cdc2/Cdc13ToRum1      "));
H_Cdc2byCdc13Star_Rum1 = hill(Cdc2byCdc13Star,
parameter_set.("Trd_of_Cdc2/Cdc13*ToRum1     "),
parameter_set.("Inh_of_Cdc2/Cdc13*ToRum1     "),
parameter_set.("Num_of_Cdc2/Cdc13*ToRum1     "));
H_PP_Rum1 = hill(PP, parameter_set.("Trd_of_PPToRum1              "),
parameter_set.("Act_of_PPToRum1              "),
parameter_set.("Num_of_PPToRum1              "));

H_Rum1_Cdc2byCdc13 = hill(Rum1, parameter_set.("Trd_of_Rum1ToCdc2/Cdc13      "),
parameter_set.("Inh_of_Rum1ToCdc2/Cdc13      "),
parameter_set.("Num_of_Rum1ToCdc2/Cdc13      "));
H_Ste9_Cdc2byCdc13 = hill(Ste9, parameter_set.("Trd_of_Ste9ToCdc2/Cdc13      "),
parameter_set.("Inh_of_Ste9ToCdc2/Cdc13      "),
parameter_set.("Num_of_Ste9ToCdc2/Cdc13      "));
H_Slp1_Cdc2byCdc13 = hill(Slp1, parameter_set.("Trd_of_Slp1ToCdc2/Cdc13      "),
parameter_set.("Inh_of_Slp1ToCdc2/Cdc13      "),
parameter_set.("Num_of_Slp1ToCdc2/Cdc13      "));

H_Cdc2byCdc13_Wee1byMik1 = hill(Cdc2byCdc13,
parameter_set.("Trd_of_Cdc2/Cdc13ToWee1/Mik1 "),
parameter_set.("Inh_of_Cdc2/Cdc13ToWee1/Mik1 "),
parameter_set.("Num_of_Cdc2/Cdc13ToWee1/Mik1 "));
H_PP_Wee1byMik1 = hill(PP, parameter_set.("Trd_of_PPToWee1/Mik1         "),
parameter_set.("Act_of_PPToWee1/Mik1         "),
 parameter_set.("Num_of_PPToWee1/Mik1         "));

H_Cdc2byCdc13_Cdc25 = hill(Cdc2byCdc13, parameter_set.("Trd_of_Cdc2/Cdc13ToCdc25     "),
parameter_set.("Act_of_Cdc2/Cdc13ToCdc25     "),
parameter_set.("Num_of_Cdc2/Cdc13ToCdc25     "));
H_PP_Cdc25 = hill(PP, parameter_set.("Trd_of_PPToCdc25             "),
parameter_set.("Inh_of_PPToCdc25             "),
parameter_set.("Num_of_PPToCdc25             "));

H_Cdc2byCdc13Star_Slp1 = hill(Cdc2byCdc13Star,
parameter_set.("Trd_of_Cdc2/Cdc13*ToSlp1     "),
parameter_set.("Act_of_Cdc2/Cdc13*ToSlp1     "),
parameter_set.("Num_of_Cdc2/Cdc13*ToSlp1     "));
H_Slp1_Slp1 = hill(Slp1, parameter_set.("Trd_of_Slp1ToSlp1            "),
parameter_set.("Inh_of_Slp1ToSlp1            "),
parameter_set.("Num_of_Slp1ToSlp1            "));

H_PP_PP = hill(PP, parameter_set.("Trd_of_PPToPP                "),
parameter_set.("Inh_of_PPToPP                "),
parameter_set.("Num_of_PPToPP                "));
H_Slp1_PP = hill(Slp1, parameter_set.("Trd_of_Slp1ToPP              "),
```

```
    parameter_set.("Act_of_Slp1ToPP              "),
    parameter_set.("Num_of_Slp1ToPP              "));

    H_Slp1_Cdc2byCdc13Star = hill(Slp1,
    parameter_set.("Trd_of_Slp1ToCdc2/Cdc13*     "),
    parameter_set.("Inh_of_Slp1ToCdc2/Cdc13*     "),
    parameter_set.("Num_of_Slp1ToCdc2/Cdc13*     "));
    H_Wee1byMik1_Cdc2byCdc13Star = hill(Wee1byMik1,
    parameter_set.("Trd_of_Wee1/Mik1ToCdc2/Cdc13*"),
    parameter_set.("Inh_of_Wee1/Mik1ToCdc2/Cdc13*"),
    parameter_set.("Num_of_Wee1/Mik1ToCdc2/Cdc13*"));
    H_Cdc25_Cdc2byCdc13Star = hill(Cdc25, parameter_set.("Trd_of_Cdc25ToCdc2/Cdc13*    "),
    parameter_set.("Act_of_Cdc25ToCdc2/Cdc13*    "),
    parameter_set.("Num_of_Cdc25ToCdc2/Cdc13*    "));
    H_Ste9_Cdc2byCdc13Star = hill(Ste9, parameter_set.("Trd_of_Ste9ToCdc2/Cdc13*     "),
    parameter_set.("Inh_of_Ste9ToCdc2/Cdc13*     "),
    parameter_set.("Num_of_Ste9ToCdc2/Cdc13*     "));
    H_Rum1_Cdc2byCdc13Star = hill(Rum1, parameter_set.("Trd_of_Rum1ToCdc2/Cdc13*     "),
    parameter_set.("Inh_of_Rum1ToCdc2/Cdc13*     "),
    parameter_set.("Num_of_Rum1ToCdc2/Cdc13*     "));


    dydt(1) = ga * H_Start_Start - ka*Start;
    dydt(2) = gb * H_Start_SK * H_SK_SK - kb*SK;
    dydt(3) = gc * H_Cdc2byCdc13Star_Ste9 * H_SK_Ste9 * H_Cdc2byCdc13_Ste9 * H_PP_Ste9 - kc*Ste9;
    dydt(4) = gd * H_SK_Rum1 * H_Cdc2byCdc13_Rum1 * H_Cdc2byCdc13Star_Rum1 * H_PP_Rum1 - kd*Rum1;
    dydt(5) = ge * H_Rum1_Cdc2byCdc13 * H_Ste9_Cdc2byCdc13 * H_Slp1_Cdc2byCdc13 - ke*Cdc2byCdc13;
    dydt(6) = gf * H_Slp1_Cdc2byCdc13Star * H_Wee1byMik1_Cdc2byCdc13Star * H_Cdc25_Cdc2byCdc13Star *
            H_Ste9_Cdc2byCdc13Star * H_Rum1_Cdc2byCdc13Star - kf*Cdc2byCdc13Star;
    dydt(7) = gi * H_Cdc2byCdc13_Wee1byMik1 * H_PP_Wee1byMik1 - ki*Wee1byMik1;
    dydt(8) = gj *H_Cdc2byCdc13_Cdc25 * H_PP_Cdc25 - kj*Cdc25;
    dydt(9) = gg * H_PP_PP * H_Slp1_PP - kg*PP;
    dydt(10) = gh * H_Cdc2byCdc13Star_Slp1 * H_Slp1_Slp1 - kh*Slp1;
    %dydt
end
```

Main Function of this section

```
    % ODE Solver RACIPE solution files
    clear
    path = '../data/fission_yeast_cc_normal/1';
    components_num = 10;
    external_signal = 0;
    sol_num = 10;

    %% Finding the solution files from the given paths
    [topo_file_info, parameter_names] = parameter_generator(path);

    path = strrep(path,'\','/');
    x = strcat(path,"/*solution_gk_*.dat");
    y = strcat(path,"/*parameters.dat");

    solution_path_dir = dir(x);
    parameter_value_file_dir = dir(y);

    paramater_value_path = strcat(parameter_value_file_dir(1).folder,"/",parameter_value_file_dir(1).name);
    paramater_value_path = strrep(paramater_value_path,'\','/');
    parameter_values = readtable(paramater_value_path);


    solution_paths = strings(length(solution_path_dir),1);

    for i = 1:length(solution_path_dir)

        s = strcat(solution_path_dir(i).folder,"/",solution_path_dir(i).name);
        solution_paths(i,1) = strrep(s,'\','/');

    end
    %% Extracting the parameter set files I want from the parameter_values table
    for i = 1:size(parameter_names,1)
        parameters(1,i) = convertCharsToStrings(parameter_names(i,1:size(parameter_names,2)));
    end

    parameter_values_newfilenames = cellstr(parameters);

    parameter_values.Properties.VariableNames = ["S_no" "States_number" parameter_values_newfilenames];

    clear i
```

```
j = 1;
for i=1:size(parameter_values,1)
    if parameter_values.States_number(i) == sol_num
        parameter_sets_for_simulating(j,:) = parameter_values(i,:);
        j=j+1;
    end
end

%% Now its time to solve ans ODE and plot the solutions
run_time = 0:1:100; % run time for ode solver
num_initials = 10; %100;

for ii = 22:42   %size(parameter_sets_for_simulating,1)
    ga = parameter_sets_for_simulating.("Prod_of_Start               ")(ii);
    gb = parameter_sets_for_simulating.("Prod_of_SK                  ")(ii);
    gc = parameter_sets_for_simulating.("Prod_of_Ste9                ")(ii);
    gd = parameter_sets_for_simulating.("Prod_of_Rum1                ")(ii);
    ge = parameter_sets_for_simulating.("Prod_of_Cdc2/Cdc13          ")(ii);
    gf = parameter_sets_for_simulating.("Prod_of_Cdc2/Cdc13*         ")(ii);
    gg = parameter_sets_for_simulating.("Prod_of_Wee1/Mik1           ")(ii);
    gh = parameter_sets_for_simulating.("Prod_of_Cdc25               ")(ii);
    gi = parameter_sets_for_simulating.("Prod_of_PP                  ")(ii);
    gj = parameter_sets_for_simulating.("Prod_of_Slp1                ")(ii);

    ka = parameter_sets_for_simulating.("Deg_of_Start                ")(ii);
    kb = parameter_sets_for_simulating.("Deg_of_SK                   ")(ii);
    kc = parameter_sets_for_simulating.("Deg_of_Ste9                 ")(ii);
    kd = parameter_sets_for_simulating.("Deg_of_Rum1                 ")(ii);
    ke = parameter_sets_for_simulating.("Deg_of_Cdc2/Cdc13           ")(ii);
    kf = parameter_sets_for_simulating.("Deg_of_Cdc2/Cdc13*          ")(ii);
    kg = parameter_sets_for_simulating.("Deg_of_Wee1/Mik1            ")(ii);
    kh = parameter_sets_for_simulating.("Deg_of_Cdc25                ")(ii);
    ki = parameter_sets_for_simulating.("Deg_of_PP                   ")(ii);
    kj = parameter_sets_for_simulating.("Deg_of_Slp1                 ")(ii);

   figure
   for jj = 1:num_initials
       [ii,jj]
       Startjj = (ga/ka)*(1+rand);
       SKjj = (gb/kb)*(rand);
       Ste9jj = (gc/kc)*(1+rand);
       Rum1jj = (gd/kd)*(1+rand);
       Cdc2byCdc13jj = (ge/ke)*(rand);
       Cdc2byCdc13Starjj = (gf/kf)*(rand);
       Wee1byMik1jj = (gg/kg)*(1+rand);
       Cdc25jj = (gh/kh)*(rand);
       PPjj = (gi/ki)*(rand);
       Slp1jj = (gj/kj)*(rand);

       I = [Startjj SKjj Ste9jj Rum1jj Cdc2byCdc13jj Cdc2byCdc13Starjj Wee1byMik1jj Cdc25jj PPjj Slp1jj];

       [t,y] = ode45(@(t,y)dynamics_simulation(t,y,parameter_sets_for_simulating(ii,:)),run_time,I);
       markers = {'k.-','b.-','r.-','k*-','b*-','k--','b--','r--','c--','m--'};
       for idx=1:size(y,2)
           plot(t,y(:,idx),markers{idx}); hold on
       end
   end
   legend('Start','SK','Ste9','Rum1','Cdc2/Cdc13','Cdc2/Cdc13*','Wee1/Mik1','Cdc25','PP','Slp1');
   hold off
end
```