Supplementary Information

Transition from Thermokinetic to Chemical Instabilities in a Modified Sal'nikov Model

Daniel Barragán* and Pablo A. Ochoa-Botache

Escuela de Química, Facultad de Ciencias, Universidad Nacional de Colombia,
Calle 59A No. 63-20, Oficina 16-413, Medellín, Colombia

MATLAB code

```
% Pablo Andrés Ochoa Botache
% Daniel Barragán
% Contains the stability analysis for the original thermokinetic Sal'nikov
% reaction model:
%
%     P ->[k1] Y
%     Y ->[k3] B          (E3 diff 0)
%
% (two variables model).

clc; clear all;

%%%%%%%%%%%%%%%% RENDERING DEFINITIONS %%%%%%%%%%%%%%%%%%%%%%%%%

% Defaults for rendering figures
width = 10;     % Width in inches
height = 7;    % Height in inches
alw = 0.75;    % AxesLineWidth
fsz = 18;      % Fontsize
lw = 0.8;      % LineWidth
msz = 8;       % MarkerSize

% The new defaults will not take effect if there are any open figures. To
% use them, we close all figures, and then repeat the first example.
close all;

% The properties we've been using in the figures
set(0,'defaultLineLineWidth',lw);   % set the default line width to lw
set(0,'defaultLineMarkerSize',msz); % set the default line marker size to msz
set(0,'defaultLineLineWidth',lw);   % set the default line width to lw
set(0,'defaultLineMarkerSize',msz); % set the default line marker size to msz
set(0,'DefaultAxesFontSize',fsz); % set the default font size to fsz

% Set the default Size for display
defpos = get(0,'defaultFigurePosition');
set(0,'defaultFigurePosition', [defpos(1) defpos(2) width*100, height*100]);
```

*e-mail: dalbarraganr@unal.edu.co

```
% Set the defaults for saving/printing to a file
set(0,'defaultFigureInvertHardcopy','on'); % This is the default anyway
set(0,'defaultFigurePaperUnits','inches'); % This is the default anyway
defsize = get(gcf, 'PaperSize');
left = (defsize(1)- width)/2;
bottom = (defsize(2)- height)/2;
defsize = [left, bottom, width, height];
set(0, 'defaultFigurePaperPosition', defsize);

%%%%%%%%%%%%%%%%%% Stability analysis %%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Interval of parameters where stability analysis is made

num = 5;

k2min = 5; % Min value for k2
k2max = 5; % Max value for k2

k3omin = 0.1; % Min value for k3o
k3omax = 2; % Max value for k3o

k2num = linspace(k2min,k2max,1);
k3onum = linspace(k3omin,k3omax,num);

nrows = num;
ncols = num;

Umin = 0.0; % Min value for heat transfer coefficient \chi
Umax = 0.04; % Max value for heat transfer coefficient \chi

Unum = linspace(Umin,Umax,num);

A = ones(nrows,ncols);

for m = 1 : num;
   U = Unum(m);

for h = 1 : num;
   k3oLow = 0;
   k2 = 5;

   for j = 1 : num;
      k3o = k3onum(j);

      syms y T
      p = [ y, T ];

      % Parameters:
      h3 = 400; % kJ/mol
      E3 = 166; % kJ/mol
      R = 0.008314; % kJ/mol
      tau = 400; % K --> External bath temperature (Ta) in the paper
      Cp = 0.029; % kJ/KÂ·mol
      S = 0.05; % m^2 (surface area)
      V = 1; % 1 dm^3 = 1 liter, cylindrical vessel with h=0.15 m and r=0.046
      po = 0.01; % mol/L
      n = V*(po); % moles
      deltaT = (1/tau) - (1/T); % For sake of simplicity
```

```
        % Rate constants
        k1 = 0.01;
        k3 = k3o*exp((E3/R)*deltaT);

        % Auxiliar equations
        v1 = k1*po*exp(-k1*1);
        v3 = k3*y;

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
        %%%%%%%%%%%%%%%% Set of differential equations to be solved %%%%%%%%%%%%


        % Eq. (1): dy/dt
        dc(1) = v1 - v3;

        % Eq. (2): dT/dt
        dc(2) = (1/(n*Cp))*(V*h3*v3 - S*U*(T-tau));


        % Evaluate the Jacobian matrix

        J = jacobian(dc, p);

        % Evaluate the steady state solution
       [SteadyState] = solve('k1*po*exp(-k1*1) - k3o*exp((E3/R)*((1/tau) - (1/T)))*y = 0',...
     '(1/(n*Cp))*(V*h3*k3o*exp((E3/R)*((1/tau) - (1/T)))*y - S*U*(T-tau)) = 0',...
     'y, T');

        n = length(SteadyState.y);

        for i = 1 : n;
            y = eval(SteadyState.y);
            T = eval(SteadyState.T);

            % Jacobian matrix at steady state:
            eval(J);

            % Eigenvalues of Jacobian at steady state:
            ev = eig(eval(J));

            % Imaginary part of ev
            im = imag(ev);

           % Decides whether ev is imaginary or not

           if im == 0 % disp('Not imaginary part')
               A(h,j) = 0;
               %plot3(k2, U, k3o, '.','Color',[.88 .48 0],'markersize', 0.1);
           else       % disp('There is imaginary part')
               A(h,j) = 1;
               %plot3(k2, U, k3o, '.','Color',[0 0 0],'markersize', 10);

             % Condition to evaluate if the imaginary point is placed at
             % the frontier
               if k3o > k3oLow
                   k3oLow = k3o;
                   B(h,m) = k3oLow;
                   C(h,m) = k2;
```

```
                D(m) = U;
            end
        end
        %hold on;
    end
  end
end

end


% Stability diagram: Only unstable points from the frontier
figure(2)
for m = 1 : num;
  for h = 1 : num;
      plot3(C(h,m), D(m), B(h,m), '.','Color',[0 0 0],'markersize', 7);
      hold on;
  end
end

xlabel('$k_{2}$','Interpreter','latex','FontSize',20);
ylabel('$U$','Interpreter','latex','FontSize',20);
zlabel('$k_{3,o}$','Interpreter','latex','FontSize',20);
title('Phase plot for Salnikov U thermokinetic model','FontSize',20)
print('phase-plot-salnikov-frontier', '-dpng', '-r600');

% Stability diagram: Only unstable points from the frontier (a mesh of
% points):
figure(4)
for m = 1 : num;
  for h = 1: num;
  DD(h,m) = D(m);
  end
end

colormap cool
surf(C,DD,B);
xlabel('$k_{2}$','Interpreter','latex','FontSize',20);
ylabel('$U$','Interpreter','latex','FontSize',20);
zlabel('$k_{3,o}$','Interpreter','latex','FontSize',20);
title('Mesh grid Salnikov U','FontSize',20)
print('surface-salnikov-frontier', '-dpng', '-r600');
```

```
% Pablo Andrés Ochoa Botache
% Daniel Barragán
% Contains the stability analysis for the unified thermokinetic Sal'nikov
% reaction model using \gamma 'g' as a bifurcation parameter:
%
%      P ->[k1] Y           (E1 = 0)
% Y + gZ ->[k2] (1+g)Z      (E2 = 0)
%      Z ->[k3] B           (E3 diff 0)
% (three variables model).

clc; clear all;

%%%%%%%%%%%%%%%% RENDERING DEFINITIONS %%%%%%%%%%%%%%%%%%%%%%%%%

% Defaults for rendering figures
width = 10;     % Width in inches
height = 7;    % Height in inches
alw = 0.75;    % AxesLineWidth
fsz = 18;      % Fontsize
lw = 0.8;      % LineWidth
msz = 8;       % MarkerSize

% The new defaults will not take effect if there are any open figures. To
% use them, we close all figures, and then repeat the first example.
close all;

% The properties we've been using in the figures
set(0,'defaultLineLineWidth',lw);   % set the default line width to lw
set(0,'defaultLineMarkerSize',msz); % set the default line marker size to msz
set(0,'defaultLineLineWidth',lw);   % set the default line width to lw
set(0,'defaultLineMarkerSize',msz); % set the default line marker size to msz
set(0,'DefaultAxesFontSize',fsz); % set the default font size to fsz

% Set the default Size for display
defpos = get(0,'defaultFigurePosition');
set(0,'defaultFigurePosition', [defpos(1) defpos(2) width*100, height*100]);

% Set the defaults for saving/printing to a file
set(0,'defaultFigureInvertHardcopy','on'); % This is the default anyway
set(0,'defaultFigurePaperUnits','inches'); % This is the default anyway
defsize = get(gcf, 'PaperSize');
left = (defsize(1)- width)/2;
bottom = (defsize(2)- height)/2;
defsize = [left, bottom, width, height];
set(0, 'defaultFigurePaperPosition', defsize);

%%%%%%%%%%%%%%%%%% Stability analysis %%%%%%%%%%%%%%%%%%%%%%%%%%

% Interval of parameters where stability analysis is made

num = 90; % Number of points to evaluate in each interval (a greater
        % number means a fine stability surface but more computation
        % time)

k2min = 1; % Min value for k2
k2max = 500; % Max value for k2

k3omin = 0.001; % Min value for k3o
k3omax = 3; % Max value for k3o
```

```
k2num = linspace(k2min,k2max,num);
k3onum = linspace(k3omin,k3omax,num);

nrows = num;
ncols = num;

Umin = 0.01; % Min value for heat transfer coefficient \chi
Umax = 0.4; % Max value for heat transfer coefficient \chi

Unum = linspace(Umin,Umax,num);

A = ones(nrows,ncols); % Initialize matrix

for m = 1 : num;
   U = Unum(m);

for h = 1 : num;
   k3oLow = 0;
   k2 = k2num(h);

   for j = 1 : num;
      k3o = k3onum(j);

      syms y z T
      p = [ y, z, T ];

      % Conditions: h1=h2=0, h3 not equal 0

      % y = c(1);
      % z = c(2);
      % T = c(3);

      % Parameters:
      h3 = 400; % kJ/mol
      E3 = 166; % kJ/mol
      R = 0.008314; % kJ/mol
      tau = 400; % K, external bath temperature (Ta) in the paper
      Cp = 0.029; % kJ/KÂ·mol
      S = 0.05; % m^2 (surface area)
      V = 1; % 1 dm^3 = 1 L, cylindrical vessel with h=0.15 m and r=0.046
      po = 0.01; % mol/L
      n = V*(po); % moles
      deltaT = (1/tau) - (1/T); % For sake of simplicity

      % Stoichiometric bifurcation parameter \gamma 'g'
      g = 1;

      % Rate constants
      k1 = 0.01; % s^-1
      k3 = k3o*exp((E3/R)*deltaT);

      % Auxiliar equations
      v1 = k1*po*exp(-k1*1);
      v1 = sym(v1);
      v2 = k2*y*power(z,g);
      v3 = k3*z;
```

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%% Set of differential equations to be solved %%%%%%%%%%%%%%%%%%


    % Eq. (1): dy/dt
    dc(1) = v1 - v2;

    % Eq. (2): dz/dt
    dc(2) =  v2 - v3;

    % Eq. (3): dT/dt
    dc(3) = (1/(n*Cp))*(V*h3*v3 - S*U*(T-tau));


    % Evaluate the Jacobian matrix

    J = jacobian(dc, p);

    % Evaluate the steady state solution
    [SteadyState] = solve('k1*po*exp(-k1*1)- k2*y*z^g = 0',...
      'k2*y*z^g - k3o*exp((E3/R)*((1/tau) - (1/T)))*z = 0',...
      '(1/(n*Cp))*(V*h3*k3o*exp((E3/R)*((1/tau) - (1/T)))*z - S*U*(T-tau)) = 0',...
      'y, z, T');

    n = length(SteadyState.y);

    for i = 1 : n;
      y = eval(SteadyState.y); z = eval(SteadyState.z);
      T = eval(SteadyState.T);

      % Jacobian matrix at steady state:
      eval(J);

      % Eigenvalues of Jacobian at steady state:
      ev = eig(eval(J));

      % Imaginary part of ev
      im = imag(ev);

     % Decides whether ev is imaginary or not

     if im == 0 % disp('Not imaginary part')
        A(h,j) = 0;
        %plot3(k2, U, k3o, '.','Color',[.88 .48 0],'markersize', 0.1);
     else      % disp('Theres imaginary par')
        A(h,j) = 1;
        %plot3(k2, U, k3o, '.','Color',[0 0 0],'markersize', 10);

       % Condition to evaluate if the imaginary point is placed at
       % the frontier
        if k3o > k3oLow
          k3oLow = k3o;
          B(h,m) = k3oLow;
          C(h,m) = k2;
          D(m) = U;
        end
     end
     %hold on;
```

```
        end
    end
end

end


% Stability diagram: Only unstable points from the frontier
figure(2)
for m = 1 : num;
    for h = 1 : num;
          plot3(C(h,m), D(m), B(h,m), '.','Color',[0 0 0],'markersize', 7);
          hold on;
    end
end

xlabel('$k_{2}$','Interpreter','latex','FontSize',20);
ylabel('$U$','Interpreter','latex','FontSize',20);
zlabel('$k_{3,o}$','Interpreter','latex','FontSize',20);
title('Phase plot for Salnikov U thermokinetic model','FontSize',20)
print('phase-plot-salnikov-frontier', '-dpng', '-r600');


% Stability diagram: Only unstable points from the frontier (a mesh of
% points):
figure(4)
for m = 1 : num;
    for h = 1: num;
    DD(h,m) = D(m);
    end
end

surf(C,DD,B);
xlabel('$k_{2}$','Interpreter','latex','FontSize',20);
ylabel('$U$','Interpreter','latex','FontSize',20);
zlabel('$k_{3,o}$','Interpreter','latex','FontSize',20);
title('Mesh grid Salnikov U','FontSize',20)
print('surface-salnikov-frontier', '-dpng', '-r600');
```

```
% Pablo Andrés Ochoa Botache
% Daniel Barragán
% Contains the discretized version of the
% unified thermokinetic Sal'nikov model.

% Conditions: h1=h2=0, h3 not equal 0

close all
clear all
clc

xspan = 1;
tspan = 400; % Simulation time interval

xpasos = 3; % Number of coupled cells
tpasos = 10000000; % Number of time steps

dx = xspan/(xpasos-1);
dt = tspan/(tpasos-1);

tg = 3; % Sampling rate
ni = fix(tpasos-1)/tg;

i = 0;
j = 0;

%% Parameters

h3 = 400; % kJ/mol
E3 = 166; % kJ/mol
R = 0.008314; % kJ/mol
Ratm = 0.08205; % atmÂ·L/molÂ·K
tau = 310; % K, External bath temperature (Ta) in the paper.
Cp = 0.029; % kJ/KÂ·mol
Surf = 0.05; % m^2 (surface area = 5 dm^2)
U = 0.05; % kW/m^2Â·K (heat transfer coefficient) or 0.00005 kJÂ·s^-1Â·dm^-2Â·K^-1 0.148
V = 1; % 1 dm^3 = 1 L, cylindrical vessel with h=0.15 m and r=0.046
po = 0.01; % Concentration of P specie in mol/L, orig = 0.01
n = V*(po); % Total moles n = nb + ny + nz moles

g = 1; % \gamma bifurcation parameter

ktrans = 0.056*1E-4;

% Rate constants
k1 = 0.01; % s^-1
k2 = 300; % MÂ·s^-1
k3o = 0.7; % s^-1 k3(Ta)
vp = k1*po*exp(-k1*1);

% Vectors initialization

y = zeros(1, xpasos);
yf = zeros(ni, xpasos);
   yy = zeros(1, xpasos);
z = zeros(1, xpasos);
zf = zeros(ni, xpasos);
   zz = zeros(1, xpasos);
T = zeros(1, xpasos);
```

```
Tf = zeros(ni, xpasos);
   TT = zeros(1, xpasos);

time = zeros(xpasos, 1);

%%
%% Initial conditions


tmax = 312;

for i=1:xpasos
      y(1,i) = 0.000001; % All cells start with equal chemical concentration
      z(1,i) = 0.000001;
      T(1,i) = i*0.1 + tmax; % A heat gradient appear

end

%% Main loop

cont = 0;
a=0;

for j=1:tpasos
   for i=2:xpasos-1

      %% Equations
      deltaT = (1/tau) - (1/T(i)); % For sake of simplicity
      k3 = k3o*exp((E3/R)*deltaT);

% \gamma = 1, mixed thermokinetical and chemical oscillator

      yy(i) = y(i) + dt*(vp - k2*y(i)*power(z(i),g));
      zz(i) = z(i) + dt*(k2*y(i)*power(z(i),g) - k3*z(i));
      TT(i) = T(i) + (dt/(n*Cp))*( V*h3*k3*z(i) - Surf*U*(T(i) - tau) + (ktrans/(dx^2))*(T(i+1) - 2*T(i) + T(i-1)) );


   end

%% Frontier conditions

      yy(1) = yy(2);
      yy(xpasos) = yy(xpasos-1);

      zz(1) = zz(2);
      zz(xpasos) = zz(xpasos-1);

      TT(1) = TT(2);
      TT(xpasos) = TT(xpasos-1);

       if mod(j,tg) == 0
          cont = cont + 1;
          yf(cont,:) = yy(1,:);
          zf(cont,:) = zz(1,:);
          Tf(cont,:) = TT(1,:);

        end

      y = yy;
```

```
    z = zz;
    T = TT;


end

for o=1:cont

   time(o) = (tspan/cont)*o;

end
```