```
% AmirHosein Sadeghimanesh
% 2020 February, modified 2021 August
%
% This script contains the computation for finding the rectangular
% representation of the multistationarity region of the LacI-TetR network
% introduced in Section 4.2 of the paper, which is depicted in Figure 6a.
%
tic
%
% The symbolic variables of this file.
%
syms x [1, 12]
syms k [1, 20]
%
% Equations before substituting the parameter values.
%
eqn = [k1*x7*x5-k5*x1,
    k2*x8*x6-k6*x2,
    k3*x1-k7*x3,
    k4*x2-k8*x4,
    k9*x7*x4-k11*x9,
    k10*x8*x3-k12*x10,
    k13*x9*x4-k15*x11,
    k14*x10*x3-k16*x12,
    x5-k17,
    x6-k18,
    x7+x9+x11-k19,
    x8+x10+x12-k20]; %#ok<COMNL>
%
% Equations after fixing values of all the parameters other than k7 and k8.
%
eqns = subs(eqn, [k1, k2, k3, k4, k5, k6, k9, k10, k11, k12, k13, k14, k15, k16, k17,↙
k18, k19, k20], [1, 1, 1, 1, 0.0082, 0.0149, 0.01, 0.01, 10000, 10000, 2, 25, 1, 9, 1,↙
1, 1, 4]);
%
% Adding the positivity assumption on our variables. Note that weare going
% to sample the parameters from positive intervals. Therefore we do not
% need extra explicit positivity assumptions on k7 and k8.
%
assume(x > 0)
%
m = 10; % the grid will have m^2 subrectangles.
NN = 10; % number of the sample points from each subrectangle.
rectRep = zeros(m); % pre-allocation of the array that stores the rectangular↙
representation.
%
for idx1 = 1:m
    aa1 = 0+(idx1-1)*(0.1-0)/m; % The start of the k7-interval of the subrectangles.
    aa2 = 0+idx1*(0.1-0)/m; % The end of the k7-interval of the subrectangles.
    for idx2 = 1:m
        bb1 = 0+(idx2-1)*(0.1-0)/m; % The start of the k8-interval of the↙
subrectangles.
        bb2 = 0+idx2*(0.1-0)/m; % The end of the k8-interval of the subrectangles.
        idxL0 = 0; % number of the points with no solution.
        idxL1 = 0; % number of the points with 1 solution.
```

```
        idxL2 = 0; % number of the points with 2 solutions.
        idxL3 = 0; % number of the points with 3 solutions. We know that 3 is the↙
upper bound, so we do not need more lists.
        for idx3 = 1:NN
            AA = sampleo(aa1, aa2); % Generating a random sample for k7 from the↙
uniform distribution.
            BB = sampleo(bb1, bb2); % Generating a random sample for k8 from the↙
uniform distribution.
            Equations = subs(eqns, [k7, k8], [AA, BB]); % Substituting the values of↙
k3 and k8 in the equations.
            eval("[" + join(arrayfun(@(idx4) "x" + idx4 + "Sol", 1:12), ",") + "]↙
=vpasolve(Equations,[" + join(arrayfun(@(idx4) "x" + idx4, 1:12), ",") + "]);"); %↙
Solving the system of equations numerically. Using the common 'solve' command is not↙
adviced, as it may give the solutions in algebraic form as root of some polynomials↙
etc.
            solutions_number = length(x1Sol); % Number of solutions.
            switch(solutions_number)
                case 1
                    idxL1 = idxL1+1;
                case 3
                    idxL3 = idxL3+1;
                case 2
                    idxL2 = idxL2+1;
                otherwise
                    idxL0 = idxL0+1;
            end
        end
        rectRep(m-idx2+1, idx1) = (idxL1+3*idxL3)/NN; % Putting the average number of↙
steady states of each subrectangle in an entry of the rectangular representation's↙
matrix. The (i,j)-entry stands for the subrectangle in the i-th section from left to↙
right and j-th section from top to buttom.
    end
end
%
% Here all the computations are completed, so we consider this place to
% stop the time.
%
toc
%
disp(rectRep) % displaying the matrix of average numbers.
%
% Writing the rectRep matrix in a txt files. Because we are not only
% going to plot the rectangular representation, but we also need this matrix to
% find the PSS representation via the rectangular representation in the
% next parts of Figure 6.
%
folder = 'C:\Home\PSS\Codes\LacI_TetR'; % replace this directory to the directory of↙
the folder you are using.
baseFileName = 'RectangularRepresentation_output.txt';
fullFileName = fullfile(folder, baseFileName);
rectRep_file = fopen(fullFileName, 'w');
fprintf(rectRep_file, 'The data matrix of the rectangular representation.\n\n');
fprintf(rectRep_file, 'm: %d\n', m);
fprintf(rectRep_file, 'n: %d\n\n', NN);
for idx1 = 1:m
```

```
    for idx2 = 1:m
        fprintf(rectRep_file, '%f,', rectRep(idx1, idx2));
    end
    fprintf(rectRep_file, '\n');
end
fclose(rectRep_file);
%
% Plotting the rectangular representation.
%
fig = figure;
fig.Units = 'pixels';
fig.Position(1:2) = [100, 100]; % The bottom left corner of the figure window on the computer screen. This has no effect on the plot itself.
fig.Position(3:4) = [540, 460]; % The whole output figure size.
for idx1 = 1:m
    aa1 = 0+(idx1-1)*(0.1-0)/m;
    aa2 = 0+idx1*(0.1-0)/m;
    for idx2 = 1:m
        bb1 = 0+(idx2-1)*(0.1-0)/m;
        bb2 = 0+idx2*(0.1-0)/m;
        fill([aa1, aa2, aa2, aa1, aa1], [bb1, bb1, bb2, bb2, bb1],...
        [
            0.57+min(max((rectRep(m-idx2+1, idx1)-1)/(3-1), 0) ,1)*(1-0.57),...
            0.88+min(max((rectRep(m-idx2+1, idx1)-1)/(3-1), 0), 1)*(1-0.88),...
            1+min(max((rectRep(m-idx2+1, idx1)-1)/(3-1), 0), 1)*(0-1)
        ]);
        hold on;
    end
end
axis([0 0.1 0 0.1])
xticks([0 0.02 0.04 0.06 0.08 0.1])
ax = gca;
ax.XRuler.Exponent = 0;
ax.Units = 'pixels';
ax.Position(1:2) = [50, 60]; % Position of the bottom left corner of the plot inside the figure.
ax.Position(3:4) = [480, 380]; % The size of the main plot.
yticks([0 0.02 0.04 0.06 0.08 0.1])
xlabel('$k_7$', 'interpreter', 'latex', 'FontName', 'Times New Roman', 'FontSize', 18)
ylabel('$k_8$', 'interpreter', 'latex', 'FontName', 'Times New Roman', 'FontSize', 18)
set(get(gca, 'ylabel'), 'rotation', 0)
% introducing the colors that we want the colorbar to have.
map = zeros(11, 3);
map(:, 1) = linspace(0.57, 1, 11);
map(:, 2) = linspace(0.88, 1, 11);
map(:, 3) = linspace(1, 0, 11);
colormap(map); % generating the colormap for the colorbar.
colorbar % The colorbar size is adjusted by the plot's vertical size.
caxis([1 3]) % the range for the colorbar ticks.
hold off
%
% Function to generate a random real number from a given interval.
%
function sampleo = sampleo(a, b)
    sampleo = a + (b-a) * rand;
```

```
end
%
% End of the file.
```