```
% AmirHosein Sadeghimanesh
% 2020 February, modified 2021 July
%
% This script contains the computation for finding the rectangular
% representation of the multistationarity region of the bistable
% autoregulatory motif introduced in Figure 3a of the paper, which is
% depicted in Figure 4c.
%
% The parametric system of equations after fixing all parameters other than
% k3 and k8 to the chosen values given in the paper was already computed in
% the Maple worksheet "Maple_bistable_autoregulatory_motif_CAD.mw".
% Therefore we do not repeat the substituation here.
%
% We also use 'tic'-'toc' to get the amount of time that Matlab takes to
% finish the computation. Note that "rectangular representation" is not a
% figure, the plotting is way to display this representation when the
% number of parameters is 2 or 3. Therefore we do not include the time
% needed for the plotting.
%
tic
%
% The symbolic variables of this file.
%
syms x1 x2 x3 x4 k3 k8
%
% 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 k3 and k8.
%
assume([x1, x2, x3, x4] > 0)
%
% Equations.
%
eqns = [-(2.76)*x1*x3+(1.55)*x4, -2*k3*x2^2+(2.81)*x1-x2+(1.96)*x3+(46.9)*x4, k3*x2^2-↙
(2.76)*x1*x3-(0.98)*x3+(1.55)*x4, x1+x4-k8];
%
% Recall that in Matlab, functions are defined at the end of the file.
% There is one function in this script called 'sampleo' which generates a
% random real number in a given interval.
%
% We are going to pick up 10 random sample points from each of 100
% subrectangles of the parameter region. These 100 subrectangles are made
% by considering a 10 by 10 grid on the parameter region. Then we solve the
% system in these parameter points and count the number of solutions.
% Finally associate the average number of solutions to each subrectangle.
%
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 = (0.0005)+(idx1-1)*((0.001)-(0.0005))/m; % The start of the k3-interval of↙
the subrectangles.
    aa2 = (0.0005)+idx1*((0.001)-(0.0005))/m; % The end of the k3-interval of the↙
```

```
subrectangles.
    for idx2 = 1:m
        bb1 = 0+(idx2-1)*(2-0)/m; % The start of the k8-interval of the subrectangles.
        bb2 = 0+idx2*(2-0)/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 k3 from the↙
uniform distribution.
            BB = sampleo(bb1, bb2); % Generating a random sample for k8 from the↙
uniform distribution.
            Equations = subs(eqns, [k3, k8], [AA, BB]); % Substituting the values of↙
k3 and k8 in the equations.
            [x1Sol, x2Sol, x3Sol, x4Sol] = vpasolve(Equations, [x1, x2, x3, x4]); %↙
Solving the system of equations numerically. This Matlab command gives all complex↙
solutions that it can find in a float form. 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 later on in
% Section 4.2.
%
folder = 'C:\Home\PSS\Codes\Bistable_autoregulatory_motif'; % 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 = (0.0005)+(idx1-1)*((0.001)-(0.0005))/m;
    aa2 = (0.0005)+idx1*((0.001)-(0.0005))/m;
    for idx2 = 1:m
        bb1 = 0+(idx2-1)*(2-0)/m;
        bb2 = 0+idx2*(2-0)/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([0.0005 0.001 0 2])
xticks([0.0005 0.0006 0.0007 0.0008 0.0009 0.0010])
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 0.5 1 1.5 2])
xlabel('$k_3$', '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.
```