```
% AmirHosein Sadeghimanesh
% 2020 February, modified 2021 July
%
% This script contains the computation for finding the sampling
% representation of the multistationarity region of the bistable
% autoregulatory motif introduced in Figure 3a of the paper, which is
% depicted in Figure 4b.
%
% 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 "sampling 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 in this script.
%
syms x1 x2 x3 x4 k3 k8
%
% Adding the positivity assumption on our variables. Note that we are 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 want to pick up 1000 randomly chosen values for k3 and k8 and solve
% the above system of equations at them.
%
NN = 1000; % number of the random points that we want to have in the sampling↙
representation.
%
% Preallocation of four arrays to save the parameter points with 0, 1, 2
% and 3 positive real solutions in them. When all 1000 points got checked,
% we shrink the preallocated arrays to have proper sizes.
%
L0 = zeros(NN, 2); % List of the points with no solution.
L1 = zeros(NN, 2); % List of the points with 1 solution.
L2 = zeros(NN, 2); % List of the points with 2 solutions.
L3 = zeros(NN, 2); % List of the points with 3 solutions. We know that 3 is the upper↙
bound, so we do not need more lists.
```

```
%
% To track the length of L_i lists, we introduce the following counters.
%
idx0 = 0;
idx1 = 0;
idx2 = 0;
idx3 = 0;
%
for idx = 1:NN
    AA = sampleo(0.0005, 0.001); % Generating a random sample for k3 from the uniform↙
distribution on the interval (0.0005,0.001).
    BB = sampleo(0, 2); % Generating a random sample for k8 from the uniform↙
distribution on the interval (0,2).
    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. It is possible to use 'solve', but we do not need↙
that here, 'vpasolve' is enough for our purpose and can be faster than finding exact↙
solutions in some cases.
    solutions_number = length(x1Sol); % Number of solutions.
    switch(solutions_number)
        case 1
            idx1 = idx1+1;
            L1(idx1, :)=[AA, BB];
        case 3
            idx3 = idx3+1;
            L3(idx3,:) = [AA, BB];
        case 2
            idx2 = idx2+1;
            L2(idx2, :)=[AA, BB];
        otherwise
            idx0 = idx0+1;
            L0(idx0, :)=[AA, BB];
    end
end
%
% Shrinking L_i lists to their real lengths.
%
L1 = L1(1:idx1, :);
L2 = L2(1:idx2, :);
L3 = L3(1:idx3, :);
L0 = L0(1:idx0, :);
%
% Here all the computations are completed, so we consider this location to
% stop the timing.
%
toc
%
%
% Writing L1 and L3 in two txt files. Because we are not only
% going to plot the finite representation, but we also need these points to
% find the PSS representation via the sampling representation later on in
% Section 4.5.
%
folder = 'C:\Home\PSS\Codes\Bistable_autoregulatory_motif'; % replace this directory↙
```

```
to the directory of the folder you are using.
baseFileName = 'SamplingRepresentation_L1_output.txt';
fullFileName = fullfile(folder, baseFileName);
L1_file = fopen(fullFileName, 'w');
fprintf(L1_file, '%d points of %d points have %d solutions. These %d points are listed
in below.\n\n', length(L1), NN, 1, length(L1));
for idx = 1:idx1
    fprintf(L1_file, '%f,%f\n', L1(idx, 1), L1(idx, 2));
end
fclose(L1_file);
baseFileName = 'SamplingRepresentation_L3_output.txt';
fullFileName = fullfile(folder, baseFileName);
L3_file = fopen(fullFileName, 'w');
fprintf(L3_file, '%d points of %d points have %d solutions. These %d points are listed
in below.\n\n', length(L3), NN, 3, length(L3));
for idx = 1:idx3
    fprintf(L3_file, '%f,%f\n', L3(idx,1), L3(idx,2));
end
fclose(L3_file);
%
% Plotting the sampling 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) = [500, 460];
scatter(L1(:, 1), L1(:, 2), 40, [0.57, 0.88, 1], 'filled'); % Points with 1 solution
are colored by sky-blue.
hold on;
scatter(L3(:, 1), L3(:, 2), 40, [1, 1, 0], 'filled') % Points with 3 solutions are
colored by yellow.
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) = [400, 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)
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.
```