```
% AmirHosein Sadeghimanesh
% 2022 January
%
% This script contains the computation for finding the PSS representation
% of the multistationarity region of the bistable autoregulatory motif,
% similar to the script
% "Matlab_bistable_autoregulatory_motif_PSS_from_sampling.m", but with all
% 8 parameters free. We use the result of another script
% "Matlab_bistable_autoregulatory_motif_sampling_8_parameters.m". The PSS
% polynomial of degree 2 is reported in Section 4.5 of the paper.
%
% PART 1
%
% Computing the target function for the optimization problem of PART 2.
%
n = 8;
syms x [1, n]
B = [1, 4; 0, 2; 0.0005, 0.001; 0, 2; 1, 4; 1, 3; 40, 50; 0, 2];
d = 2;
[p, c] = multPoly(n, x, d);
f = intOverB(p, n, x, B);
disp(f)
%
% PART 2
%
% Computing the coefficients of the polynomial 'p' of the PSS
% representation using optimization by YALMIP and SEDUMI having only 1 SOS
% constraints, the rest of constraints are linear inequalities.
%
K = [1.711851,0.917698,0.000982,1.093611,2.563407,1.463189,44.888977,1.248120;
    1.097802,1.122400,0.000941,1.338351,1.571300,1.737833,44.607259,1.963276;
    1.206418,0.639199,0.000765,1.308891,2.222858,2.639962,47.183589,1.937299
    1.565986,0.574996,0.000546,1.152419,3.050090,2.093186,44.257288,1.288886;
    2.213740,0.896746,0.000683,1.527009,2.883689,2.543961,49.328536,1.945482;
    3.085489,0.998232,0.000768,0.890366,1.371797,1.980715,48.529982,1.747855;
    2.673367,0.626858,0.000583,1.244995,3.963804,1.340864,42.577923,0.793599;
    1.483402,1.516225,0.000936,0.701553,3.056607,1.588297,45.306293,1.664847;
    1.542213,0.510773,0.000510,1.847351,2.961100,2.865227,41.635124,1.842195;
    2.771826,1.320876,0.000524,0.697570,2.354022,1.481810,47.150450,1.712365;
    1.902457,1.878819,0.000990,0.573241,3.402461,2.792223,45.975266,1.768033;
    1.974566,0.492456,0.000671,0.751384,2.639661,2.123840,43.958222,0.796262;
    1.184204,0.992578,0.000821,0.442531,3.511169,2.942150,48.463729,1.011999;
    3.291020,1.117641,0.000592,0.995898,2.553537,2.988486,48.548517,1.924808;
    3.036823,0.807003,0.000967,0.958969,1.695375,1.792580,47.050775,1.117118;
    1.027998,1.830052,0.000821,0.002838,1.091156,1.416940,44.549661,0.254532;
    1.657851,0.651613,0.000548,1.495067,3.245527,2.086599,43.381323,1.664667;
    2.452116,1.278062,0.000944,0.397474,2.186099,2.984351,44.023516,1.317713;
    3.704044,1.990764,0.000827,0.216873,1.108342,2.236182,45.671444,1.923929;
    1.659044,0.919284,0.000979,1.580091,2.355624,1.666856,40.590953,1.481811;
    2.341022,1.175143,0.000939,0.938201,2.312255,2.492370,44.679105,1.721655;
    1.579607,1.232843,0.000635,1.119356,3.834352,2.428943,46.792197,1.918761;
    3.042699,0.833870,0.000690,0.426541,2.148814,1.059336,44.723211,0.666745;
    1.205018,0.819633,0.000562,0.886033,3.696819,1.707278,41.201781,1.138222;
    1.995874,1.497494,0.000822,0.338476,3.856615,2.086540,42.514135,1.157145;
    1.989712,0.684210,0.000909,1.063371,2.563366,2.548619,41.202628,1.250900;
```

```
    2.562551,0.438155,0.000921,1.325866,3.448706,2.587755,44.691052,0.619050;
    1.018677,0.748691,0.000951,0.636690,2.791249,1.595590,41.250144,0.776711;
    1.529984,1.027358,0.000774,0.330555,2.481679,2.070235,41.988072,1.246338;
    1.454022,0.993441,0.000904,1.265738,3.065204,2.279140,47.293217,1.719691;
    1.038267,0.754319,0.000584,1.080446,1.304987,1.078535,49.332291,1.943184;
    1.315961,0.536322,0.000882,1.611020,1.312759,1.939518,42.190619,1.845416;
    2.501202,0.865526,0.000952,1.260367,3.949109,2.170401,48.406365,0.937628;
    1.507646,0.860905,0.000708,1.457527,2.219421,2.903615,49.119850,1.902829;
    2.136376,0.536339,0.000576,1.261999,1.949124,2.918224,44.986740,1.477217;
    1.867451,0.995736,0.000909,1.190257,2.609274,1.661746,44.116902,1.588012;
    1.404106,0.901035,0.000786,1.584047,2.259210,2.065073,49.257044,1.798164;
    1.342085,0.885082,0.000830,0.589546,3.851105,2.388572,42.068065,1.109525;
    1.230130,0.708946,0.000566,0.316359,1.186441,2.403687,40.864817,1.233574;
    3.177609,0.678892,0.000636,0.340549,2.992055,2.071703,48.291094,0.534724;
    2.912568,0.848571,0.000953,0.834642,1.462175,2.079999,49.370914,1.321910;
    3.786638,1.345435,0.000686,0.811391,2.316472,2.357298,44.650703,1.906528;
    1.765349,0.611647,0.000508,1.174976,3.887675,2.699695,40.079403,1.268051;
    1.004422,0.590792,0.000524,0.885481,3.369539,2.827041,45.332544,1.608153;
    1.177460,0.645305,0.000890,0.670957,2.858706,2.985771,46.480057,1.079552;
    2.639452,1.101656,0.000847,1.854988,3.832631,2.806527,46.041063,1.852420;
    1.168287,0.670395,0.000815,1.783956,3.020190,2.370534,46.957448,1.599663;
    2.100628,0.551691,0.000633,0.453077,1.003962,2.786015,44.534975,1.156842;
    2.207986,1.348579,0.000776,0.102914,1.922488,2.930890,49.314751,0.756053;
    3.687160,0.548198,0.000999,1.668707,3.374031,2.313225,45.436730,0.773363;
    1.260686,0.750995,0.000594,1.621053,3.169745,2.349348,49.139814,1.641922;
    2.008337,1.906691,0.000875,0.667848,2.898498,1.831091,45.764998,1.919001;
    1.166651,0.768095,0.000620,0.455617,2.071361,2.848205,45.024983,1.606475;
    3.636194,0.785899,0.000621,0.563253,3.714313,2.405350,48.530557,0.757367;
    1.002426,1.806013,0.000841,0.147702,3.989152,1.628914,48.092198,0.922901;
    1.656705,0.624351,0.000987,0.618381,3.838743,1.821124,40.103336,0.620138;
    1.761183,1.557092,0.000905,0.995906,3.747169,1.201967,41.358246,1.716634;
    3.193881,0.493244,0.000541,1.705817,3.059656,2.916221,49.850556,1.069445;
    1.703547,1.413968,0.000847,0.530242,1.979721,1.186468,40.415994,1.518560];
a = partSOSFitting(n, x, p, c, f, B, K);
disp(a)
% Saving the coefficient vector of the PSS polynomial in a txt file.
folder = 'C:\Home\PSS\Codes\Bistable_autoregulatory_motif\Higher_dimension'; % replace↙
this directory to the directory of the folder you are using.
baseFileName = 'PSS_via_sampling_degree_2_output_vector_a.txt';
fullFileName = fullfile(folder, baseFileName);
a_vec_file = fopen(fullFileName, 'w');
fprintf(a_vec_file, 'The coefficient vector of the polynomial p of the PSS↙
respresentation.\n\n');
fprintf(a_vec_file, 'a: ');
for i = 1:length(a)
    fprintf(a_vec_file, '%f,', a(i));
end
fclose(a_vec_file);
%
disp(subs(p, c, a)); % displaying p with its coefficients as predicted in a.
%
%%%%%%%%%%%%%
% Functions %
%%%%%%%%%%%%%
%
```

```
% Generating a matrix that each of its rows is an exponent vector of a
% monomial of degree at most d in n variables.
% The monomials are ordered with respect to the graded lexicographic
% monomial order.
%
function vMtx = allMonomials(n, d)
    % if isinteger(n)==0 || isinteger(d)==0 || n<=0 || d<0
    %     error("The input arguments are not valid. The first argument must be a positive integer and the second argument must be a nonnegative integer.");
    % end
    vMtx = zeros(nchoosek(d+n, n), n);
    for idx = 1:1:nchoosek(d+n, n)-1
        vMtx(idx+1, :) = nxtMonomial(vMtx(idx, :));
    end
end
%
% Generating the next expnent vector of the next monomial in the graded
% lexicographic order.
%
function v = nxtMonomial(v, n)
    % if nargin>1
    %     if isinteger(n)==0 || n<1 || n~=length(v)
    %         error("The second input argument must be a positive integer equal to the length of the first input argument.");
    %     end
    % else
    %     n=length(v);
    % end
    if nargin == 1
        n = length(v);
    end
    if n == 1
        v(1) = v(1)+1;
        return
    end
    idx1 = 0;
    for idx2 = n:-1:1
        if v(idx2) ~= 0
            idx1 = idx2;
            break;
        end
    end
    if idx1 == 0
        v(1) = 1;
        return
    end % so there is a first nonzero index.
    if idx1 ~= n
        v(idx1) = v(idx1)-1;
        v(idx1+1) = v(idx1+1)+1;
        return
    end % here we know idx1=n, so no point in keeping idx1 for saving a fixed known number which is already saved at some variable.
    idx1 = 0;
    for idx2 = n-1:-1:1
        if v(idx2) ~= 0
```

```
            idx1 = idx2;
            break;
        end
    end
    if idx1 == 0
        v(1) = v(n)+1;
        v(n) = 0;
        return
    end % so there is a second nonzero index.
    tmpMem = v(n);
    v(n) = 0;
    v(idx1) = v(idx1)-1;
    v(idx1+1) = v(idx1+1)+tmpMem+1;
end
%
% The following function receives an integer, a symbol and another integer,
% respectively n, x and d. Then returns a polynomial in n variables of
% total degree d with all monomials. It also returns a second output which
% is a vector of coefficients of this polynomial.
%
function [p, c] = multPoly(n, x, d)
    %syms x [1,n] % remove x from the input arguments and ncomment this
    %line if you want the function generate the variables as x1 ... xn
    %itself.
    Mtx = allMonomials(n, d);
    c = str2sym(arrayfun(@(idx) "c" + join("_" + Mtx(idx,:), ""), 1:size(Mtx, 1)));
    p = sum(arrayfun(@(idx) c(idx).*prod(x.^Mtx(idx,:)), 1:size(Mtx, 1)));
end
%
% The following function receives a polynomial, an integer, a symbol and a
% hyperrectangle, respectively called p, n, x and B. Then it returns the
% n-dimensional integral of p in x over B.
%
function f = intOverB(p, n, x, B)
    f = p;
    for idx = n:-1:1
        f = int(f, x(idx), B(idx, :));
    end
end
%
% The following function is the SOS + linear optimization formulating the
% PSS polynomial with the help of YALMIP and SeDuMi.
%
function PSSCoeffs = partSOSFitting(n, x, p, c, f, B, K) %#ok<STOUT>
    % The following string should not be produced after the spdvar xi's,
    % otherwise you may get a strange string with x61 or x82 etc.
    tmpStr = "Goal=[sos(p";
    tmpStr = tmpStr + join(arrayfun(@(idx) "-s" + idx + "B*(" + string(x(idx)) + "-("↙
+ B(idx, 1) + "))*((" + B(idx ,2)+ ")-" + string(x(idx)) + ")", 1:n), "") + "),\n";
    tmpStr = tmpStr + join(arrayfun(@(idx) "sos(s" + idx + "B)", 1:n), ",\n") + ",\n";
    tmpStr = tmpStr + join(arrayfun(@(idx) ...
        string(subs(p, x, K(idx, :))) + ">=1", 1:size(K, 1)), ",\n") + "];";
    % back to the expected order of the lines.
    for idx = 1:n
        eval("sdpvar " + string(x(idx)));
```

```
    end
    for idx = 1:length(c)
        eval("sdpvar " + string(c(idx)));
    end
    eval("p=" + string(p));
    eval("F=" + string(f));
    for idx1 = 1:n
        eval("[s" + idx1 + "B,c" + idx1 + "B]=polynomial([" + join(arrayfun(@(idx2)↙
string(x(idx2)), 1:n)) + "],0)");
    end
    eval(sprintf(tmpStr));
    eval("solvesos(Goal, F, [], ["+ ...
        join(arrayfun(@(idx) "c" + idx + "B;", 1:n), "") + ...
        join(arrayfun(@(idx) string(c(idx)), 1:length(c)), ";") + "])");
    eval("PSSCoeffs=[" + join(arrayfun(@(idx) "value(" + string(c(idx)) + ")", 1:↙
length(c)), ",") + "]");
end
%
% End of the file.
```