```
% This script estimates the optimal influences
% T1_A_opt,T2_A_opt,T1_B_opt,T2_B_opt for a set of given ranges for
 the
% optimal influence model parameters defined for the  worst case
 within a
% 14 dimmensional grid space.

% IMPORTANT NOTE!!!
% This script file provides a slight modification of the original
 script
% file "OligopolisticOptimalInfluencesGridSearcher.m" which lies upon
 the
% computation of the optimal prices (pA_opt, pB_opt), the optimal
% quantities (Q_A_opt, Q_B_opt) and the optimal profits (F_A_opt,
 F_B_opt)
% and their constituent quantities associated with the corresponding
 revenue
% and cost (F_A_Rev_opt, F_A_Cost_opt, F_B_Rev_opt, F_B_Cost_opt).
 This script
% computes the aforementioned quantities at a later stage outside the
% multiple nested loop.

% For some reason, conducting the computation of the aforementioned
% quantities within the multiple nested loop affects the validity of
 the
% obtained results. This is an issue that demands for further
% investigation.

clc
clear all
% Define external optimization parameters that will remain constant
% throughout the whole experimentation within the context of the
% oligopolistic competition environment. That is, each firm holds the
% maximum possible initial belief for its own product and the minimum
% possible initial belief for the other firm's product.
P_A_A = 1;
P_A_B = 0;
P_B_A = 0;
P_B_B = 1;

% Define ranges and corresponding increment step for each parameter of
 the
% optimal influence model.

% Parameter #1
Lambda_A_1_MIN = 0.25;
Lambda_A_1_MAX = 0.25;
Lambda_A_1_STEP = 0.01;
% Parameter #2
Lambda_A_2_MIN = 0.25;
Lambda_A_2_MAX = 0.25;
Lambda_A_2_STEP = 0.01;
```

```
% Parameter #3
Lambda_B_1_MIN = 0.25;
Lambda_B_1_MAX = 0.25;
Lambda_B_1_STEP = 0.01;
% Parameter #4
Lambda_B_2_MIN = 0.25;
Lambda_B_2_MAX = 0.25;
Lambda_B_2_STEP = 0.01;
% Parameter #5
Theta1_MIN = 0.2;
Theta1_MAX = 0.2;
Theta1_STEP = 0.05;
% Parameter #6
Theta2_MIN = 0.4;
Theta2_MAX = 0.4;
Theta2_STEP = 0.01;
% Parameter #7
P_A_1_MIN = 0.0;
P_A_1_MAX = 1.0;
P_A_1_STEP = 0.05;
% Parameter #8
P_A_2_MIN = 0.2;
P_A_2_MAX = 0.2;
P_A_2_STEP = 0.01;
% Parameter #9
P_B_1_MIN = 0.2;
P_B_1_MAX = 0.2;
P_B_1_STEP = 0.01;
% Parameter #10
P_B_2_MIN = 0.2;
P_B_2_MAX = 0.2;
P_B_2_STEP = 0.01;
% Parameter #11
M_MIN = 0.3;
M_MAX = 0.3;
M_STEP = 0.05;
% Parameter #12
K_MIN = 0.3;
K_MAX = 0.3;
K_STEP = 0.05;
% Parameter #13
C_MIN = 0.0001;
C_MAX = 0.0001;
C_STEP = 0.000001;
% Parameter #14
Gamma_MIN = 0.5;
Gamma_MAX = 0.5;
Gamma_STEP = 0.01;

% Set the corresponding ranges for each one of the model parameters.
Lambda_A_1_RANGE = [Lambda_A_1_MIN:Lambda_A_1_STEP:Lambda_A_1_MAX];
Lambda_A_2_RANGE = [Lambda_A_2_MIN:Lambda_A_2_STEP:Lambda_A_2_MAX];
Lambda_B_1_RANGE = [Lambda_B_1_MIN:Lambda_B_1_STEP:Lambda_B_1_MAX];
Lambda_B_2_RANGE = [Lambda_B_2_MIN:Lambda_B_2_STEP:Lambda_B_2_MAX];
```

```
Theta1_RANGE = [Theta1_MIN:Theta1_STEP:Theta1_MAX];
Theta2_RANGE = [Theta2_MIN:Theta2_STEP:Theta2_MAX];
P_A_1_RANGE = [P_A_1_MIN:P_A_1_STEP:P_A_1_MAX];
P_A_2_RANGE = [P_A_2_MIN:P_A_2_STEP:P_A_2_MAX];
P_B_1_RANGE = [P_B_1_MIN:P_B_1_STEP:P_B_1_MAX];
P_B_2_RANGE = [P_B_2_MIN:P_B_2_STEP:P_B_2_MAX];
M_RANGE = [M_MIN:M_STEP:M_MAX];
K_RANGE = [K_MIN:K_STEP:K_MAX];
C_RANGE = [C_MIN:C_STEP:C_MAX];
Gamma_RANGE = [Gamma_MIN:Gamma_STEP:Gamma_MAX];

% Initialize matrices that store fundamental intermediate quantities
 of the
% underlying optimization problem within the oligopolistic enviroment
 as
% well as the values of the external optimization parameters for each
 step
% of the multi-dimensional grid searching process.
Topts = []; % Optimal investment levels.
Sopts = []; % Optimal limiting influences.
Xopts = []; % Optimal limiting beliefs.
Popts = []; % Optimal prices.
Qopts = []; % Optimal quantities.
Fopts = []; % Optimal profits.
Fvals = []; % Optimal values for the combined optimization objective.
OptFlags = []; % Optimization flags (i.e. number of different valid
 solutions).
DigitFlags = []; % Length of maximal sequence of identical digits
 within the obtained optimal solutions.
Params = []; % 14-plet of the varying optimization parameters.

% Perform the actual grid searching.
for Lambda_A_1 = Lambda_A_1_RANGE
    for Lambda_A_2 = Lambda_A_2_RANGE
        for Lambda_B_1 = Lambda_B_1_RANGE
            for Lambda_B_2 = Lambda_B_2_RANGE
                for Theta1 = Theta1_RANGE
                    %Theta1
                    for Theta2 = Theta2_RANGE
                        for P_A_1 = P_A_1_RANGE
                            P_A_2 = P_A_1;
                            %for P_A_2 = P_A_2_RANGE
                                P_B_1 = P_A_1;
                                %for P_B_1 = P_B_1_RANGE
                                    P_B_2 = P_A_1;
                                    %for P_B_2 = P_B_2_RANGE
                                        for M = M_RANGE
                                            for K = K_RANGE
                                                for C = C_RANGE
                                                    for Gamma =
 Gamma_RANGE
                                                        P_A_1
                                                        params =
 [P_A_1,P_A_2,P_B_1,P_B_2,Lambda_A_1,Lambda_A_2,Lambda_B_1,Lambda_B_2,Theta1,Theta
```

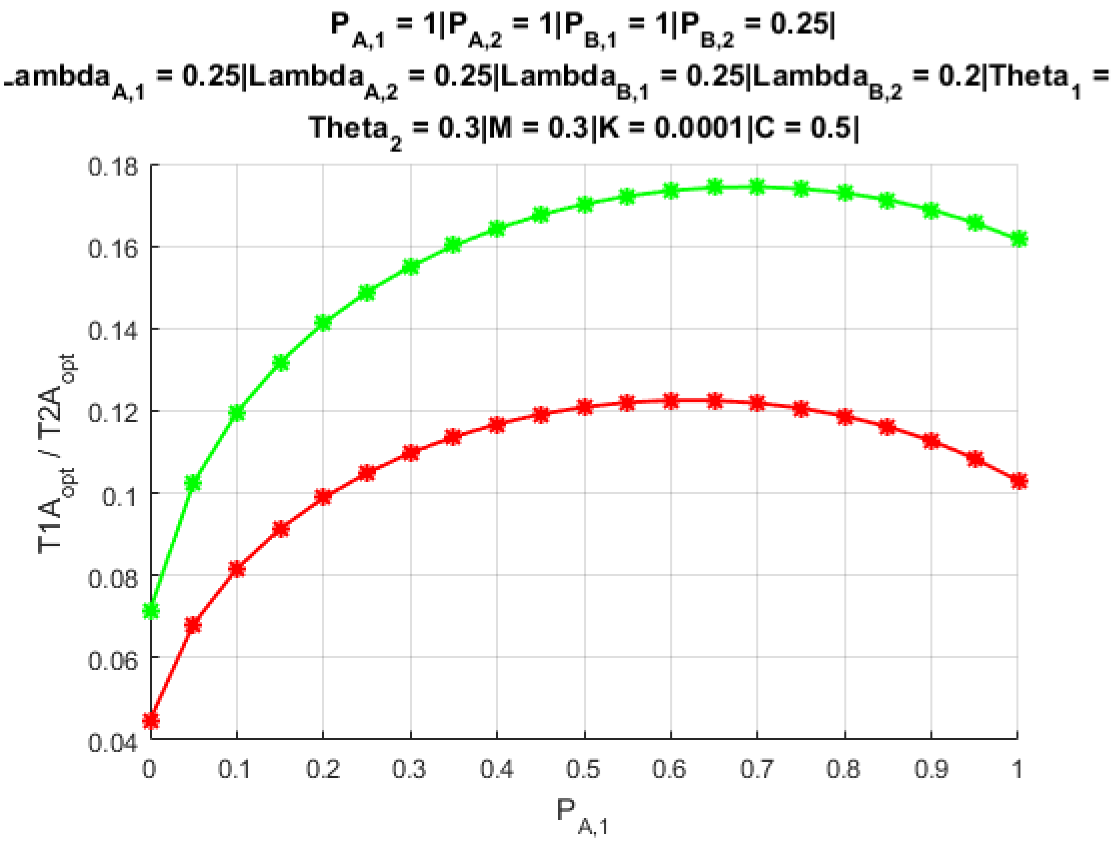
```
                                                        Params =
[Params;params];

[T1_A_opt,T2_A_opt,T1_B_opt,T2_B_opt,Fval,OptFlag,DigitFlag] =
OligopolisticOptimalInfluences(Lambda_A_1,Lambda_A_2,Lambda_B_1,Lambda_B_2,Theta1

            [SA_opt,S1_opt,S2_opt,SB_opt] =
OligopolisticSOptimal(T1_A_opt,T2_A_opt,T1_B_opt,T2_B_opt,Lambda_A_1,Lambda_A_2,L

[XA_opt,XB_opt] =
OligopolisticXOptimal(P_A_A,P_A_1,P_A_2,P_A_B,P_B_A,P_B_1,P_B_2,P_B_B,SA_opt,S1_o

[pA_opt,pB_opt] = OligopolisticPOptimal(XA_opt,XB_opt,C,K,M);

[Q_A_opt,Q_B_opt] = OligopolisticQOptimal(XA_opt,XB_opt,C,K,M);

[F_A_opt,F_B_opt,F_A_Rev_opt,F_A_Cost_opt,F_B_Rev_opt,F_B_Cost_opt]
=
OligopolisticFOptimal(T1_A_opt,T2_A_opt,T1_B_opt,T2_B_opt,pA_opt,pB_opt,Q_A_opt,Q
                                                        T =
[T1_A_opt,T2_A_opt,T1_B_opt,T2_B_opt];
                                                        S =
[SA_opt,S1_opt,S2_opt,SB_opt];
                                                        X =
[XA_opt,XB_opt];
                                                        P =
[pA_opt,pB_opt];
                                                        Q =
[Q_A_opt,Q_B_opt];
                                                        F =
[F_A_opt,F_B_opt,F_A_Rev_opt,F_A_Cost_opt,F_B_Rev_opt,F_B_Cost_opt];
                                                        Fvals =
[Fvals;Fval];
                                                        OptFlags =
[OptFlags;OptFlag];
                                                        DigitFlags =
[DigitFlags;DigitFlag];
                                                        Topts =
[Topts;T];
                                                        Sopts =
[Sopts;S];
                                                        Xopts =
[Xopts;X];
                                                        Popts =
[Popts;P];
                                                        Qopts =
[Qopts;Q];
                                                        Fopts =
[Fopts;F];
                                                    end;
                                                end;
                                            end;
                                        end;
                                    %end;
```

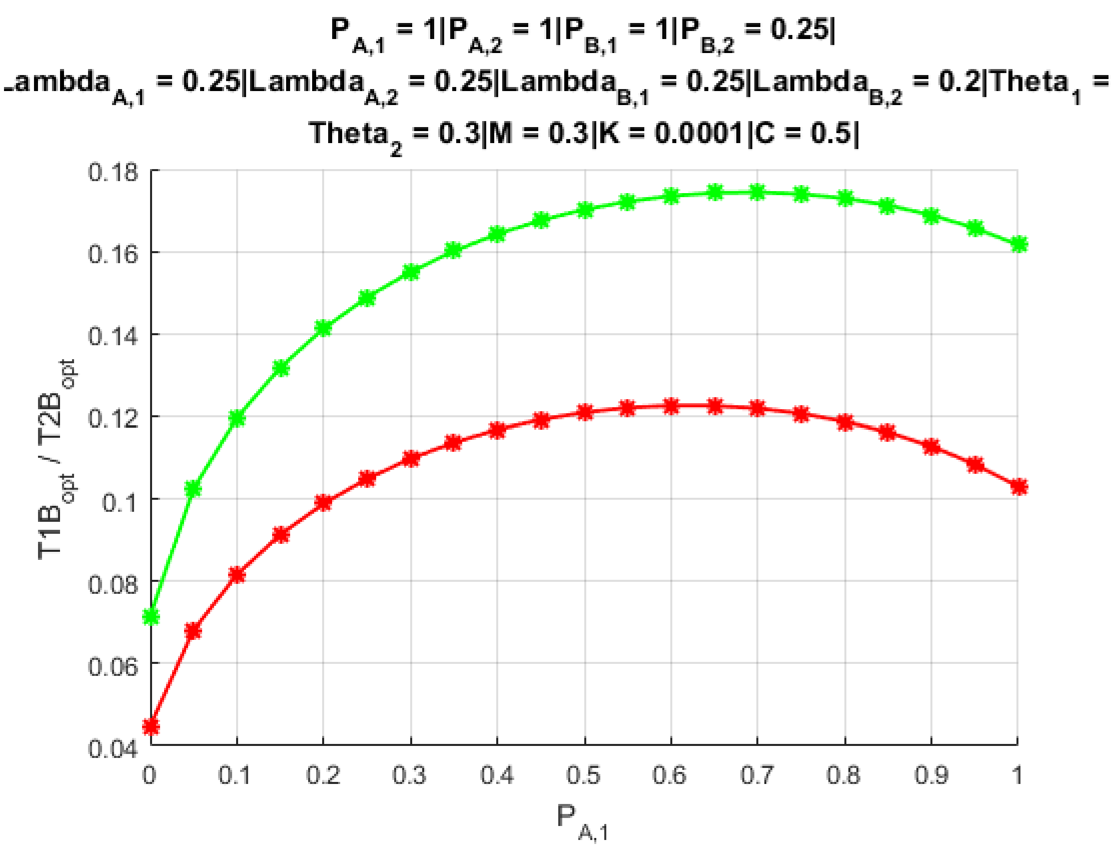
```
                                        %end;
                                    %end;
                                end;
                            end;
                        end;
                    end;
                end;
            end;
        end;

% Get the previously computed optimal influence levels for the two
 firms
% for each consumer.
% % T1_A = Topts(:,1);
% % T2_A = Topts(:,2);
% % T1_B = Topts(:,1);
% % T2_B = Topts(:,2);

% Get the previously computed optimal beliefs (XA and XB) for the two
 firms
% A and B.
% % XA = Xopts(:,1);
% % XB = Xopts(:,2);
% Get the optimal price levels (pA and pB) for the two firms A and B.
% % [pA,pB] = OligopolisticPOptimal(XA,XB,C,K,M);
% Get the optimal quantity levels (Q_A and Q_B) for the two firms A
 and B.
% % [Q_A,Q_B] = OligopolisticQOptimal(XA,XB,C,K,M);
% Get the optimal profit levels (F_A and F_B) for the two firms A and
 B as
% well as their associated costituent quantities
 (F_A_Rev_opt,F_A_Cost_opt,
% F_B_Rev_opt,F_B_Cost_opt).
% % [F_A,F_B,F_A_Rev,F_A_Cost,F_B_Rev,F_B_Cost] =
 OligopolisticFOptimal(T1_A,T2_A,T1_B,T2_B,pA,pB,Q_A,Q_B,C,Gamma);

% Form matrices Popts, Qopts and Fopts.
% % Popts = [pA,pB];
% % Qopts = [Q_A,Q_B];
% % Fopts = [F_A,F_B,F_A_Rev,F_A_Cost,F_B_Rev,F_B_Cost];

% Set the varying parameter index.
ParamIndex = 1;
% Set parameters' indices and corresponding values for the parameters
 that
% remain constant.
ConstIndices = setdiff(1:length(params),ParamIndex);
ConstValues = params(ConstIndices);
% Perform plotting operations.
plot_parameters_tuples(Topts,Sopts,Xopts,Popts,Qopts,Fopts,Params,ParamIndex,Const
```

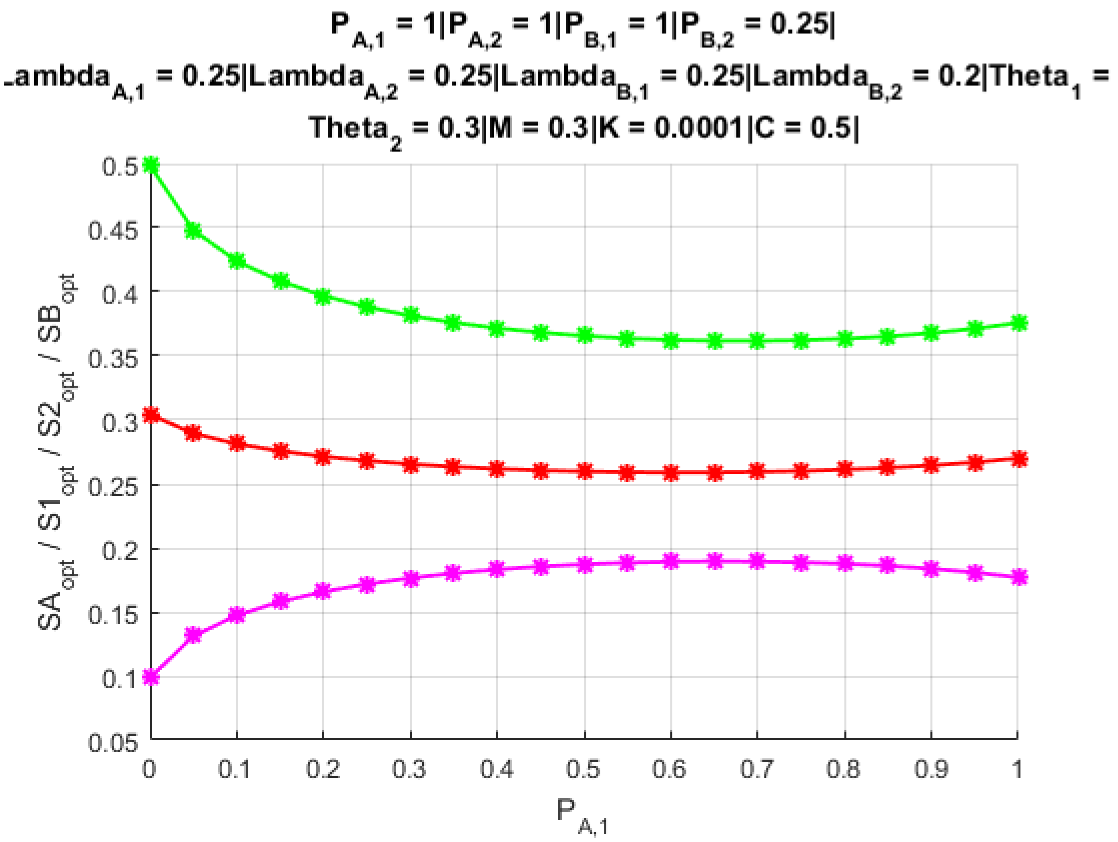
P_A_1 =

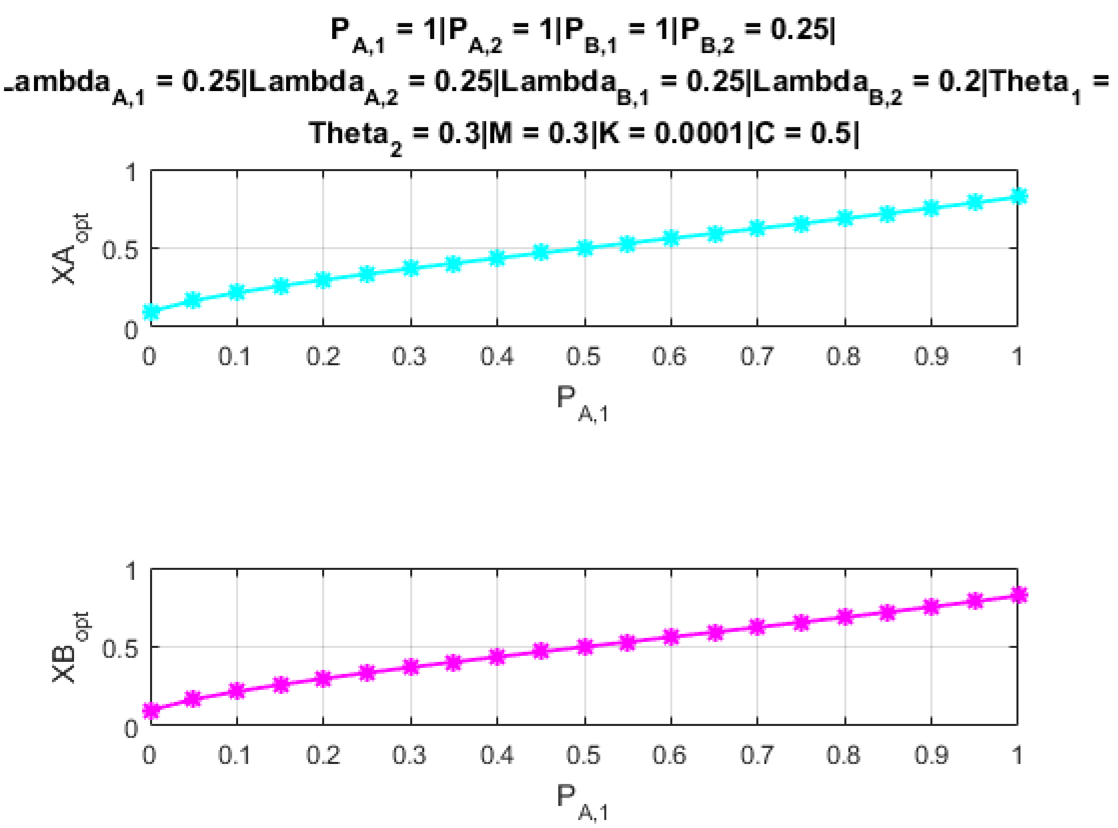
```
     0

P_A_1 =

   0.050000000000000

P_A_1 =

   0.100000000000000

P_A_1 =

   0.150000000000000

P_A_1 =

   0.200000000000000

P_A_1 =

   0.250000000000000

P_A_1 =

   0.300000000000000

P_A_1 =

   0.350000000000000

P_A_1 =

   0.400000000000000

P_A_1 =

   0.450000000000000

P_A_1 =

   0.500000000000000

P_A_1 =
```

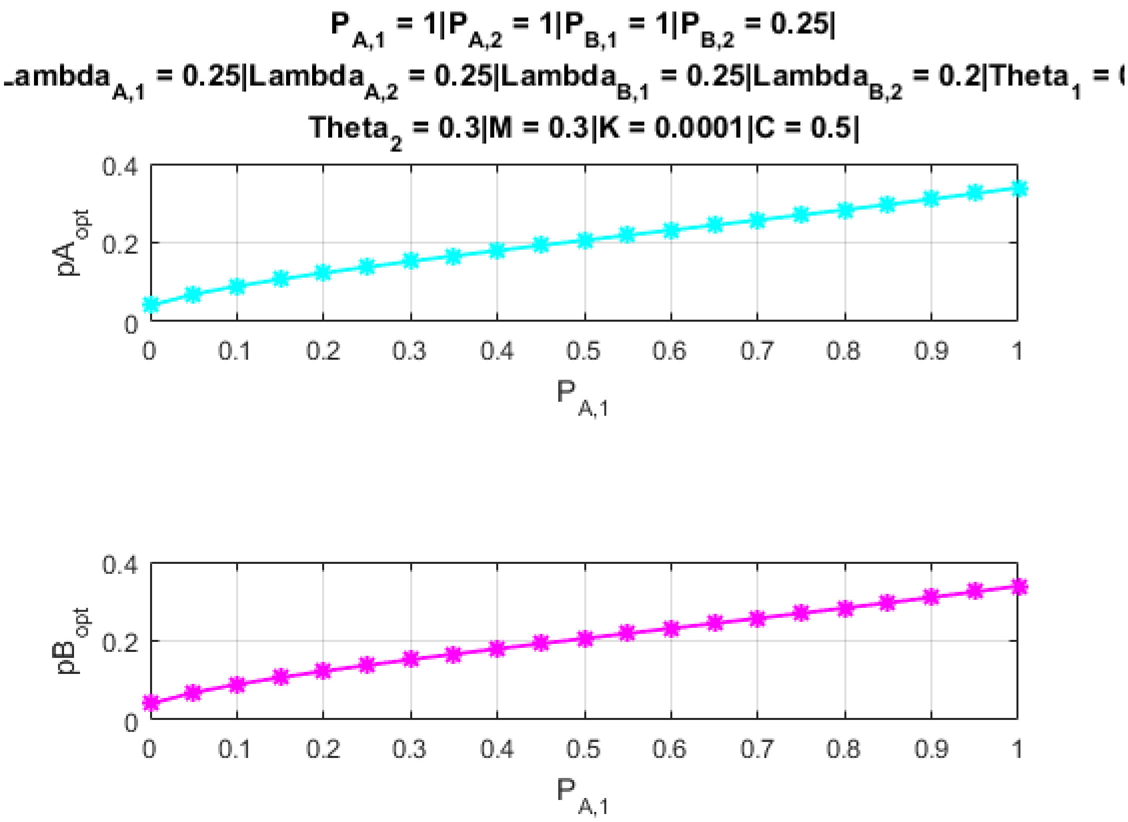
```
   0.550000000000000

P_A_1 =

   0.600000000000000

P_A_1 =

   0.650000000000000

P_A_1 =

   0.700000000000000

P_A_1 =

   0.750000000000000

P_A_1 =

   0.800000000000000

P_A_1 =

   0.850000000000000

P_A_1 =

   0.900000000000000

P_A_1 =

   0.950000000000000

P_A_1 =

     1

Warning: More than one valid solutions were found.
```

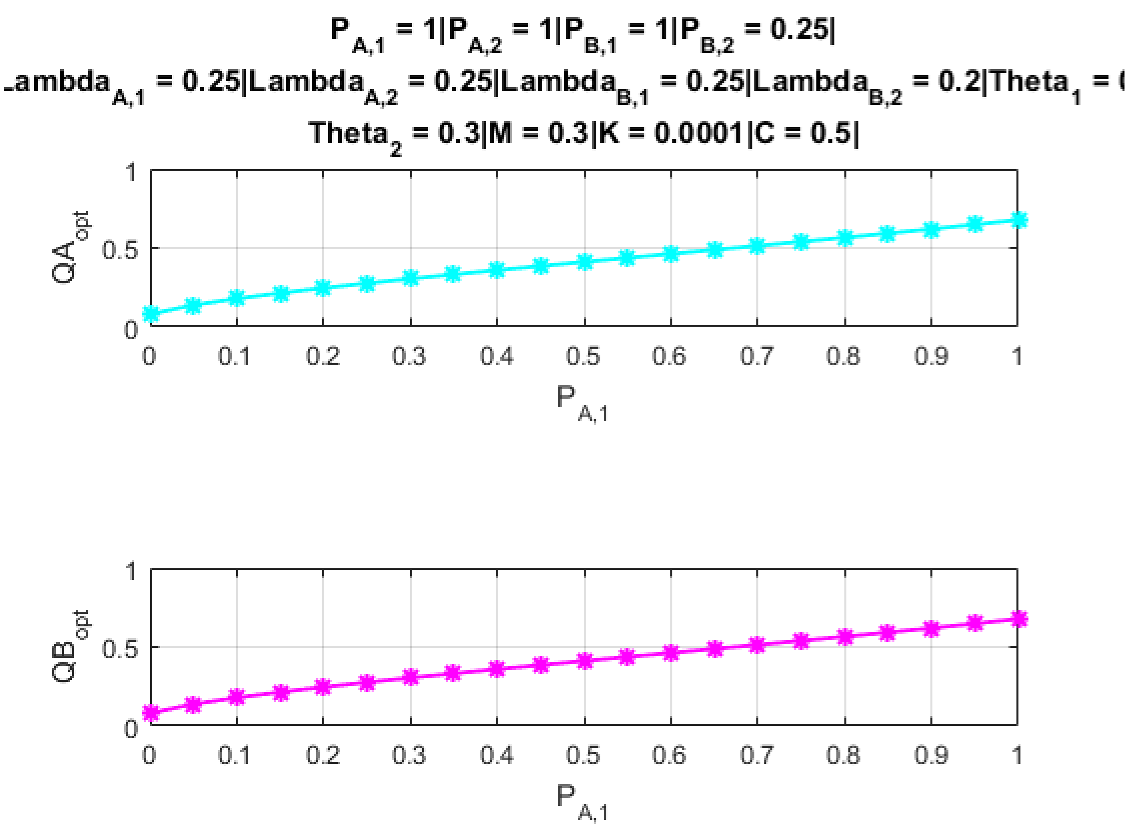

P A,1 = 1|P A,2 = 1|P B,1 = 1|P B,2 = 0.25|
-ambda A,1 = 0.25|Lambda A,2 = 0.25|Lambda B,1 = 0.25|Lambda B,2 = 0.2|Theta 1 = (
Theta 2 = 0.3|M = 0.3|K = 0.0001|C = 0.5|
T1A opt / T2A opt
P A,1

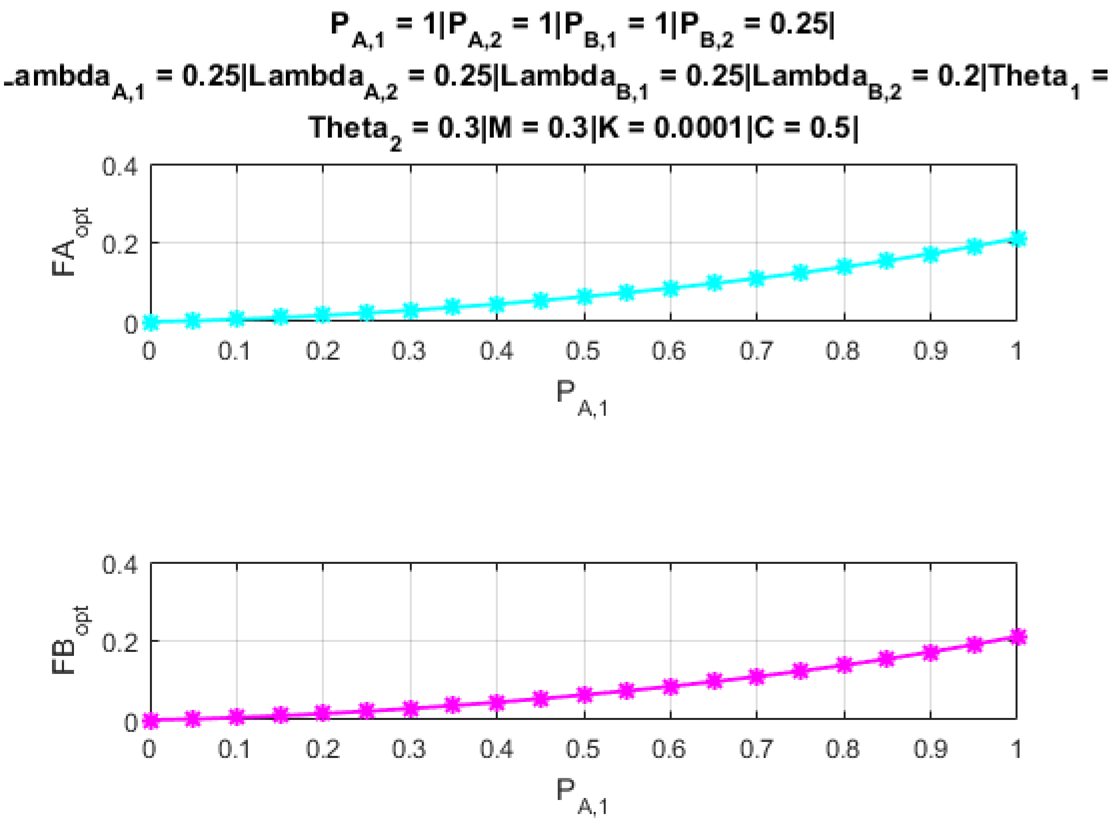

P A,1 = 1|P A,2 = 1|P B,1 = 1|P B,2 = 0.25|
-ambda A,1 = 0.25|Lambda A,2 = 0.25|Lambda B,1 = 0.25|Lambda B,2 = 0.2|Theta 1 = (
Theta 2 = 0.3|M = 0.3|K = 0.0001|C = 0.5|
T1B opt / T2B opt
P A,1

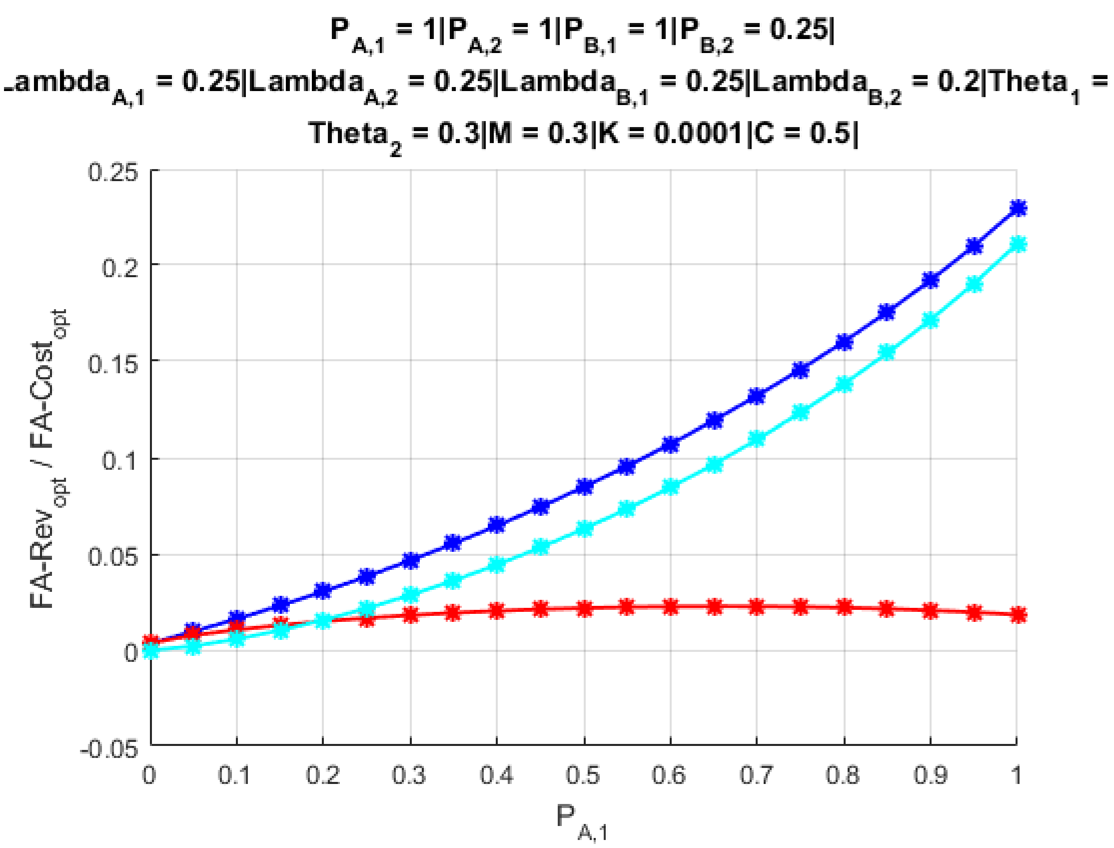

P_{A,1} = 1|P_{A,2} = 1|P_{B,1} = 1|P_{B,2} = 0.25|
.ambda_{A,1} = 0.25|Lambda_{A,2} = 0.25|Lambda_{B,1} = 0.25|Lambda_{B,2} = 0.2|Theta_1 = (
Theta_2 = 0.3|M = 0.3|K = 0.0001|C = 0.5|
SA_opt / S1_opt / S2_opt / SB_opt
P_{A,1}

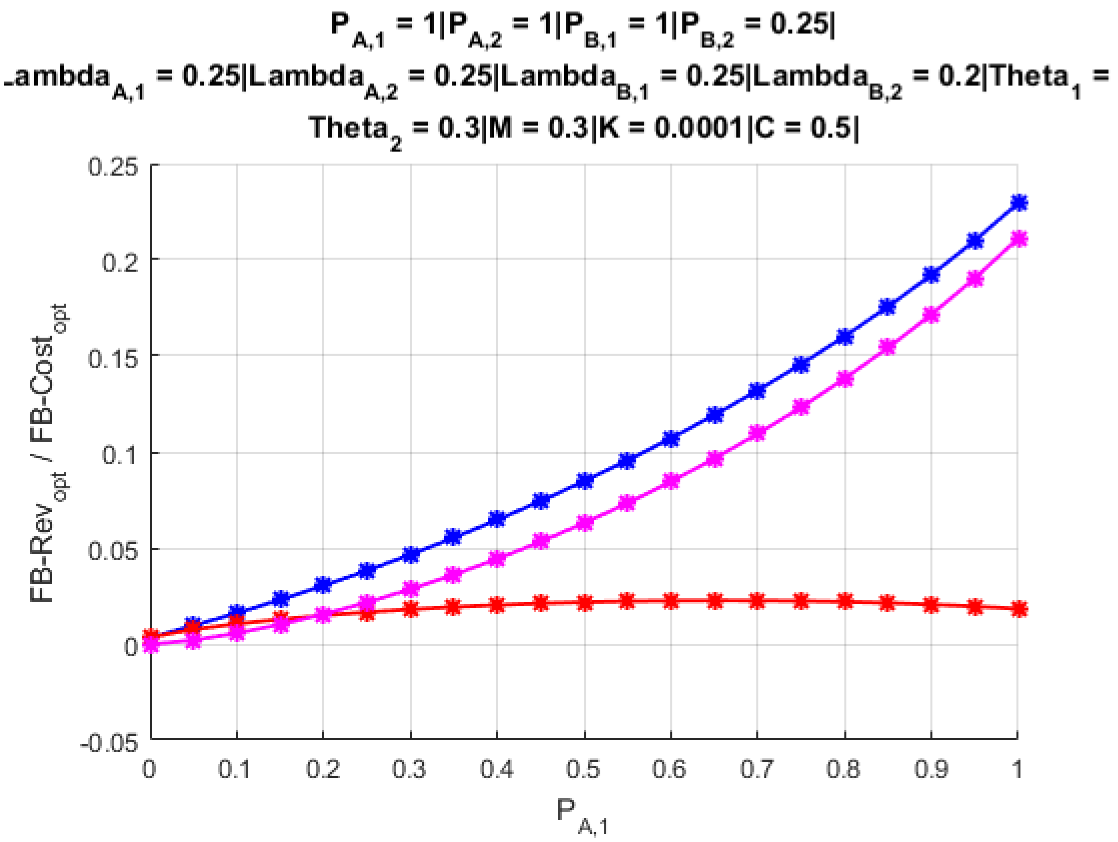

P_{A,1} = 1|P_{A,2} = 1|P_{B,1} = 1|P_{B,2} = 0.25|
.ambda_{A,1} = 0.25|Lambda_{A,2} = 0.25|Lambda_{B,1} = 0.25|Lambda_{B,2} = 0.2|Theta_1 = (
Theta_2 = 0.3|M = 0.3|K = 0.0001|C = 0.5|
XA_opt
P_{A,1}
XB_opt
P_{A,1}


P A,1 = 1|P A,2 = 1|P B,1 = 1|P B,2 = 0.25|
.ambda A,1 = 0.25|Lambda A,2 = 0.25|Lambda B,1 = 0.25|Lambda B,2 = 0.2|Theta 1 = (
Theta 2 = 0.3|M = 0.3|K = 0.0001|C = 0.5|
pA opt
P A,1
pB opt
P A,1


P A,1 = 1|P A,2 = 1|P B,1 = 1|P B,2 = 0.25|
.ambda A,1 = 0.25|Lambda A,2 = 0.25|Lambda B,1 = 0.25|Lambda B,2 = 0.2|Theta 1 = (
Theta 2 = 0.3|M = 0.3|K = 0.0001|C = 0.5|
QA opt
P A,1
QB opt
P A,1


P A,1 = 1|P A,2 = 1|P B,1 = 1|P B,2 = 0.25|
Lambda A,1 = 0.25|Lambda A,2 = 0.25|Lambda B,1 = 0.25|Lambda B,2 = 0.2|Theta 1 =
Theta 2 = 0.3|M = 0.3|K = 0.0001|C = 0.5|
FA opt
P A,1
FB opt
P A,1


P A,1 = 1|P A,2 = 1|P B,1 = 1|P B,2 = 0.25|
Lambda A,1 = 0.25|Lambda A,2 = 0.25|Lambda B,1 = 0.25|Lambda B,2 = 0.2|Theta 1 =
Theta 2 = 0.3|M = 0.3|K = 0.0001|C = 0.5|
FA-Rev opt / FA-Cost opt
P A,1



Published with MATLAB® R2016a