```
new ;
closeall ;

//  ************************************************************************
//
//  incae_tutorial_03_solution_dynamic_game.gss
//
//  Program for the solution of dynamic game of market entry-exit with 2 firms
//
//  Given parameters fixed by user, the program:
//      (a) computes an equilibrium (Markov Perfect equilibrium) of the model
//
//  by Victor Aguirregabiria
//
//  ************************************************************************

library pgraph ;

// --------------
//  OUTPUT FILE
// --------------
wdir = "C:\\Users\\victo\\Dropbox\\documents\\COURSES\\2022_INCAE\\TUTORIALS\\" ;
// Working directory
buff = changedir(wdir) ;
fileout  = "incae_tutorial_03_solution_dynamic_game.out" ; // Name of output file
output file = ^fileout reset ;
format /mb1 /ros 16,4 ;

// ----------------------------------------------------------------
//  Specification of Profit Function
//
//  i = index for firm. i belongs to {1, 2}
//  Firms' actions = a_1 and a_2
//
//      a_i = 0 ----> Firm i is not active in market
//      a_i = 1 ----> Firm i is active in market
//
// ----------------------------------------------------------------
//
//  The game is dynamic because there is a sunk cost of entry
//
//  The only state variables of this game are the indicators of whether
//  the firms were active or not in the market at previous period.
//  These are payoff relevant state variables because they determine
//  whether a firm has to pay an entry cost to operate in the market.
//
//          x_i = a_i[t-1]
//
//          x_i = 0 ----> Firm i was not active in market at previous period
//              = 1 ----> Firm i was active in market at previous period
```

```
//
// ---------------------------------------------------------------
//
//  Parameters:
//      ecost =   Entry cost
//      pi_mon =   Operating profit of a monopolist
//      pi_duo =   Operating profit of a duopolist
//
// ---------------------------------------------------------------
//
//  Profit function for firm 1 (the symmetric for firm 2)
//
//      Profit_1(a_1 = 0) = eps_1(0)
//
//      Profit_1(a_1 = 1) = (1-a_2) * pi_mon + a_2 * pi_duo
//
//                        - (1-a_1[t-1]) * ecost + eps_1(1)
//
//  where eps_1(0) and eps_1(1) are private information shocks that are
//  i.i.d. over time and over players, and independent of each other,
//  with Standard Normal distribution.
//
// ---------------------------------------------------------------

// ------------------------------------------------
//  1. Values of parameters and other constants
// ------------------------------------------------
ecost  = 1.0 ;    // Entry cost
pi_mon = 1.0 ;     // Monopoly  profit
pi_duo = 0.4 ;     // Duopoly profit

dfact   = 0.9 ;    // Discount factor

nplayer = 2 ;       // Number of players

trueparam = (ecost | pi_mon | pi_duo) ;  // vector with true values of parameters
namesb = "Entry Cost"
       | "Monopoly Prof"
       | "Duopoly Prof" ; // Vector with names of parameters
kparam  = rows(trueparam) ;   //  Number of parameters to estimate

// 'vstate' is a matrix with all the possible values of
//  the state variables of this dynamic game.
//  That is, a matrix with all the possible values of x_1 and x_2
//
//  - Each row represents a value for the state variables
//  - Column 1 contains the values of x_1, and column 2 the values of x_2

vstate = (0 ~ 0)
       | (0 ~ 1)
```

```
       | (1 ~ 0)
       | (1 ~ 1) ;

nstate = rows(vstate) ; // Number of possible values of the vector of state
variables

//
------------------------------------------------------------------------------
------
//  2. PROCEDURE FOR EVALUATING BEST-RESPONSE MAPPING
//
------------------------------------------------------------------------------
------
//
//  Input of the procedure:
//      prob0 = (nstate x 2) matrix with the CCPs of each player
//
//      Each row corresponds to a state, each column corresponds to the CCP of a
player
//
//  Output of the procedure:
//      newprob = (nstate x 2) matrix with the best response CCPs of each player
//
//      The i-th column is the vector of CCPs which are player i's best response
to prob0
//
//
------------------------------------------------------------------------------
------


proc (1) = equilmap(prob0);
  local ns, newprob, profit1_a0, profit1_a1, profit2_a0, profit2_a1,
        iptran1_a0, iptran1_a1, iptran2_a0, iptran2_a1,
        eprofit1, eprofit2, ftran, value1, value2, vtilda1, vtilda2 ;
  ns = rows(prob0) ;    // number of states
  newprob = prob0 ;

  // ------------------------------
  // a. Vectors of expected profits
  // ------------------------------

  // Expected profit of firms 1 and 2 if own action is 0
  profit1_a0 = zeros(ns,1) ;
  profit2_a0 = zeros(ns,1) ;

  // Expected profit of firms 1 and 2 if own action is 1
  profit1_a1 = (1-prob0[.,2]).*pi_mon + prob0[.,2].*pi_duo -
ecost.*(1-vstate[.,1]) ;
  profit2_a1 = (1-prob0[.,1]).*pi_mon + prob0[.,1].*pi_duo -
```

```
ecost.*(1-vstate[.,2]) ;

  // Ex-ante expected profit of firms 1 and before knowing the own epsilons
  // Assumption about distribution of epsilon: Standard Normal
  eprofit1 = (1-prob0[.,1]).*profit1_a0 + prob0[.,1].*profit1_a1
           + sqrt(2) * pdfn(cdfni(prob0[.,1])) ;

  eprofit2 = (1-prob0[.,2]).*profit2_a0 + prob0[.,2].*profit2_a1
           + sqrt(2) * pdfn(cdfni(prob0[.,2])) ;

  // ------------------------------------------------------
  //  b. Transition probabilities of the state variables
  // ------------------------------------------------------
  //
  //  - (nstate x nstate) matrices of transition probabilities of
  //    state variables
  //
  //  - In these transition matrices:
  //        - Rows represent current value of state variables
  //        - Columns represent next period value of state variables
  //            (the elements of each row should sum to 1)
  //
  //  - Remember: vstate = (0~0) | (0~1) | (1~0) | (1~1)
  //
  //  - Since the only state variables are the lagged values
  //    of firms' actions, all what we need to obtain the
  //    transition probabilities is the choice probabilities
  //
  //  - Transition matrices from the point of view of each firm:
  //    Each firm knows his current action but not the current action
  //    of his competitor
  //
  // ------------------------------------------------------

  // Trasition matrix of firm 1 if current own action = 0
  //  Remember: vstate = (0~0)     | (0~1)          | (1~0)       | (1~1)
  iptran1_a0 =      (1-prob0[.,2]) ~ prob0[.,2]  ~ zeros(ns,1)~ zeros(ns,1) ;

  // Trasition matrix of firm 1 if current own action = 1
  //  Remember: vstate = (0~0)     | (0~1)          | (1~0)       | (1~1)
  iptran1_a1 =      zeros(ns,1)    ~ zeros(ns,1) ~ (1-prob0[.,2]) ~ prob0[.,2] ;

  // Trasition matrix of firm 2 if current own action = 0
  //  Remember: vstate = (0~0)     | (0~1)          | (1~0)       | (1~1)
  iptran2_a0 =      (1-prob0[.,1]) ~ zeros(ns,1)    ~ prob0[.,1] ~ zeros(ns,1) ;

  // Trasition matrix of firm 2 if current own action = 1
  //  Remember: vstate = (0~0)     | (0~1)          | (1~0)       | (1~1)
  iptran2_a1 =      zeros(ns,1)    ~ (1-prob0[.,1]) ~ zeros(ns,1)~ prob0[.,1] ;
```

```
  // Ex-ante transition probability matrix
  //    Not conditional on own action
  //  Remember: vstate = (0~0) | (0~1) | (1~0) | (1~1)

   ftran = (1-prob0[.,1]).*(1-prob0[.,2])
         ~ (1-prob0[.,1]).*prob0[.,2]
         ~ prob0[.,1].*(1-prob0[.,2])
         ~ prob0[.,1].*prob0[.,2] ;

  //  -------------------------------------------------------
  //  c. Calculating present values for each firm and state
  //        Given that firms' believe that they will behave
  //        in the future according to the probabilities prob0
  //  -------------------------------------------------------
  ftran = inv(eye(ns)-dfact*ftran) ;    // 'Discounting' matrix
  value1 = ftran * eprofit1 ;   // Present values for firm 1
  value2 = ftran * eprofit2 ;   // Present values for firm 2

  // Threshold value for firm 1:
  //    Value of action 1 minus value of action 0
  vtilda1 = (profit1_a1 + dfact * iptran1_a1 * value1)
          - (profit1_a0 + dfact * iptran1_a0 * value1) ;

  // Threshold value for firm 2:
  //    Value of action 1 minus value of action 0
  vtilda2 = (profit2_a1 + dfact * iptran2_a1 * value2)
          - (profit2_a0 + dfact * iptran2_a0 * value2) ;

  //  -------------------------------------------------------
  //  d. Best response probabilities
  //  -------------------------------------------------------
  newprob = cdfn(vtilda1/sqrt(2)) ~ cdfn(vtilda2/sqrt(2)) ;
  retp(newprob) ;
endp ;
// -------------- End procedure equimap ------------------


// ------------------------------------------------------------------------
//  3. COMPUTING AN EQUILIBRIUM (MPE)
// ------------------------------------------------------------------------
//
//  Here the algorithm for computing an equilibrium
//  is fixed point iterations in the best response equilibrium mapping
//
// ------------------------------------------------------------------------

prob0 = rndu(nstate,nplayer) ; // Initial value for the vector of players' CCPs

cconv = 1e-12 ;  // Convergence constant
criter = 1000 ;  // Initial value for the convergence criterion
```

```
maxiter = 100 ;  // Maximum number of iterations to stop
                 // Remember, the equilibrium mapping is not a contraction
                 // such that fixed point iterations might not converge

iter=1 ;
do while (criter>cconv).and(iter<=maxiter) ;
  "         Best response mapping iteration  =";; iter ;
  "         Convergence criterion =";; criter ;
  "" ;
  prob1 = equilmap(prob0);
  criter = maxc(maxc(abs(prob1-prob0))) ;
  prob0 = prob1 ;
  iter=iter+1 ;
endo ;
pequil = prob0 ;

if (iter<maxiter) ;
  "-------------------------------------------------------------------------" ;
  "         CONVERGENCE ACHIEVED AFTER";; iter;; "BEST RESPONSE ITERATIONS";
  "-------------------------------------------------------------------------" ;
  "         EQUILIBRIUM PROBABILITIES" ;
  pequil ;
  "" ;
  "     Remember that" ;
  "" ;
  "     vstate  = (0 ~ 0)" ;
  "             | (0 ~ 1)" ;
  "             | (1 ~ 0)" ;
  "             | (1 ~ 1)" ;
  "-------------------------------------------------------------------------" ;
endif ;


// ----------------------------------------------------------------
// 4. COMPUTING THE STEADY-STATE DISTRIBUTION OF THE STATE VARIABLES
//      for THE COMPUTED EQUILIBRIUM
//
//      This is the distribution of "market structure" in steady state
// ----------------------------------------------------------------


// Transition probability matrix for the state variables
ftran = (1-pequil[.,1]).*(1-pequil[.,2])
      ~ (1-pequil[.,1]).*pequil[.,2]
      ~ pequil[.,1].*(1-pequil[.,2])
      ~ pequil[.,1].*pequil[.,2] ;

// By definition, the steady-state distribution 'psteady'
//  is such that:
//
```

```
//      psteady = Ftran * psteady
//
//  where Ftran is the transition probability matrix of the state vars
//
//   We can calculate psteady by fixed point iterations in the vector
//   system:  psteady = Ftran * psteady
// ------------------------------------------------------------------------

cconv = 1e-6 ;
criter = 1000 ;
psteady = (1/nstate)*ones(nstate,1) ;
do while criter>cconv ;
  "Criter =";; criter ;
  pbuff = ftran'*psteady ;
  criter = maxc(abs(pbuff-psteady)) ;
  psteady = pbuff ;
endo ;

"-------------------------------------------------------------------------" ;
"       Ergodic Distribution of the State Variables" ;
"" ;
psteady ;

"" ;
"-------------------------------------------------------------------------" ;
"" ;
"       Remember that" ;
"" ;
"           vstate  =   (0 ~ 0)" ;
"                     | (0 ~ 1)" ;
"                     | (1 ~ 0)" ;
"                     | (1 ~ 1)" ;
"" ;
"-------------------------------------------------------------------------" ;


output off ;

end ;
```