Software for the Computation of Markov-Perfect Equilibria in a Dynamic Game of Store Location by Multi-Store Firms

Victor Aguirregabiria and Gustavo Vicentini

University of Toronto, Analysis Group

May 2007

Software for the Computation of Markov-Perfect Equilibria in a Dynamic Game of Store Location by Multi-Store Firms

Victor Aguirregabiria*
University of Toronto

Gustavo Vicentini
Analysis Group - Boston

May 2007

Abstract

This document is a supplement of the paper *"Dynamic Spatial Competition Between Multi-Store Firms"* by Aguirregabiria and Vicentini (2007). It describes in detail the library of programs and procedures, in GAUSS language, that is used in that paper. The program computes an equilibrium of a dynamic game of store location and spatial competition by multi-store firms. The equilibrium of the game is a space-time stochastic process for the network of stores of each firm as well as for prices, markups, profits and consumer welfare at every location in the geographic market. We illustrate the use of the program with an example.

TABLE OF CONTENTS

*Contact Address: Victor Aguirregabiria. Department of Economics, University of Toronto. E-mail: *victor.aguirregabiria@utoronto.ca.* Comments are welcome.

1 Overview

This manual describes a library of programs and procedures that implement an algorithm to compute a Markov Perfect Equilibrium (MPE) of the dynamic game presented in the paper *"Dynamic Spatial Competition Between Multi-Store Firms"* by Aguirregabiria and Vicentini (2007). The programs are written in GAUSS language. The list of programs (*.prg*) and procedures (*.src*) is:

Program / Procedure	**Description**
spatial_main.prg	Main program
spatial_bertrand.src	Compute Nash-Bertrand equilibrium of pricing game
spatial_mpe.src	Computes MPE of dynamic game
spatial_pack.src	Packs primitives of the model into a GAUSS structure
sigma_algebra.src	Creates a matrix with all states of the game
spatial_grid.src	Discretizes the space of consumer locations.
pdf_bn.src	Computes a bivariate Normal density
spatial_ld.src	Computes array of local demands
iv.src	Integrates local demands over space
dist_p.src	Computes distances between consumers and stores
spatial_dp.src	Solves Bellman equation
spatial_ccvalue.src	Calculates choice-specific expected next period values
spatial_tranp.src	Calculates matrix of transition probabilities

The user should create a GAUSS library name *spatial*, where all procedures should be put into and called from. There is one main program, *spatial_main.prg*, where all the primitives of the game are specified and most procedures are called from. Once the primitives are specified, there are two major procedures that are called from this program. One is to compute the Nash-Bertrand equilibrium and variable profits for each possible value of state variables (*spatial_bertrand.src*), and the

other to compute the MPE of the dynamic game of entry-exit and location choice (*spatial_ mpe.src*). There are eight other auxiliary procedures that are called upon throughout the algorithm. The algorithm that we provide is intended to serve as a basic framework, and therefore we normalize some parameters that appear in the paper in order to simplify the understanding and exposition of the program. Including these parameters back in the program should be straightforward. More specifically, we impose restrictions on the following parameters:

Normalization	Description
$\bar{n} = 1$	Firms are allowed to have at most one store per location each
$FC_i = 0$	Fixed costs are normalized to zero

2 Main Program (*spatial_ main.prg*)

The main program where all the primitives are specified and the two major procedures are called from is *spatial_ main.prg*. This program is divided into five sections.

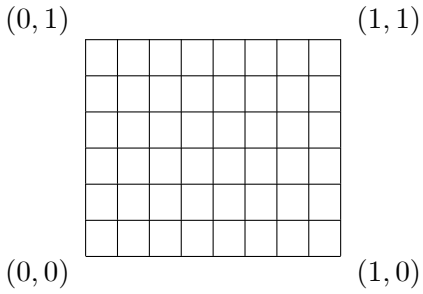
SECTION 1 (Specification of Primitives):

Subsection 1.1: Specification of parameters for the boundaries of the Market $\mathbb{C}$. The user specifies its x- and y-axis boundaries and in how many cells should the market be divided into (for integration):

Paper	Program	Dimension	Description
x-*axis* of $\mathbb{C}$	*xaxis*	1×2	Left boundary ($xaxis$[1,1]) and Right Boundary ($xaxis$[1,2]) of the x-axis for $\mathbb{C}$.
y-*axis* of $\mathbb{C}$	*yaxis*	2×1	Upper boundary ($yaxis$[1,1]) and Lower Boundary ($yaxis$[2,1]) of the y-axis for $\mathbb{C}$.
grid for $\mathbb{C}$	*ncell*	1×2	Number of grid Cells at x-axis ($ncell$[1,1]) and at y-axis ($ncell$[1,2]) of the market $\mathbb{C}$.

For instance, if the user specifies the boundaries as $xaxis = (0, 1)$ and $yaxis = (1, 0)'$, such that $\mathbb{C}$ is the unit square, and chooses $ncell = (8, 6)$, then:

$\mathbb{C}$ =



Subsection 1.2: Specification of the number of feasible business locations (i.e. the submarkets) and their geographic coordinates:

Paper	Program	Dimension	Description
L	$Lnum$	scalar	Number of feasible locations.
z	$zmat$	$L \times 2$	Geographic coordinates of feasible business locations. The 1^{st} row is the (x, y) coordinates for location 1, the 2^{nd} row is for location 2, etc.

Subsection 1.3: Specification of the Utility function primitives $\{\tau, \mu, outs\}$:

Paper	Program	Dimension	Description
τ	tau	scalar	Transportation cost parameter
μ	miu	scalar	Dispersion parameter of consumers' unobserved heterogeneity
outside alternative	$outs$	scalar	Utility from choosing outside alternative

Subsection 1.4: Specification of the aggregate population distribution and evolution (i.e. ϕ_t). We define ϕ_t to be a bivariate normal distribution, and that it evolves according to an exogenous Markov process specified by the $nump \times nump$ transition matrix tr_phi, where $nump$ is the number

of possible different values for ϕ_t.

Paper	Program	Dimension	Description
ϕ_t	*phi*	$nump \times 6$	Parametrization of all possible ϕ_t distributions. 1^{st} row parametrizes the 1^{st} possible realization of ϕ_t, 2^{nd} row parametrizes the 2^{nd} possibility, and so on. The first 2 columns of each row give the coordinates of the population mean for that realization; The 3^{rd} and 4^{th} columns give the variance of the population along the x- and y-axis, respectively; The 5^{th} column gives the variance-covariance of the population along $\mathbb{C}$, and the last column gives the size of the population for that realization of ϕ_t.
Transition Matrix for ϕ_t	*tr_phi*	$nump \times nump$	1^{st}-order Transition Matrix for the possible realizations of ϕ_t.

Subsection 1.5: Specification of the primitives of firms $\left\{I, \beta, \omega_i, c_i, \theta^{EC}, \theta^{EV}\right\}$:

Paper	Program	Dimension	Description
I	*Inum*	scalar	*N*umber of firms that are potentially operative.
β	*beta*	scalar	Discount Factor of Firms.
ω_i	*omeg*	$I \times 1$	Vector with quality level of each firm; First row is quality level of first firm, ω_1, second row is for the second firm, ω_2, and so on. (common across stores/locations).
c_i	*c*	$I \times 1$	Vector with marginal cost of each firm; First row is marginal cost of first firm, c_1, and so on. (common across stores/locations).
θ^{EC}	*ec*	$I \times L$	Matrix with *Entry Costs* of each Firm at each Location; First row is Entry Costs of firm 1, $\left\{\theta_{11}^{EC}, \theta_{12}^{EC}, ..., \theta_{1L}^{EC}\right\}$, second row is for firm 2, $\left\{\theta_{21}^{EC}, \theta_{22}^{EC}, ..., \theta_{2L}^{EC}\right\}$, and so on.
θ^{EV}	*ev*	$I \times L$	Matrix with *Exit Values* of each Firm at each Location; First row is Exit Values of firm 1, $\left\{\theta_{11}^{EV}, \theta_{12}^{EV}, ..., \theta_{1L}^{EV}\right\}$, second row is for firm 2, $\left\{\theta_{21}^{EV}, \theta_{22}^{EV}, ..., \theta_{2L}^{EV}\right\}$, and so on.

SECTION 2 (Packing all Primitives Together): This section takes all the parameters that were specified in Section 1 and "packs" them into a GAUSS *structure* called *theta*. It does so by

calling the procedure *spatial_pack.src*. The user does not need to specify any parameters in this section, just call the procedure. So:

$$theta = \theta = \left\{xaxis, yaxis, ncell, L, z, \tau, \mu, outs, phi, tr\_phi, I, beta, omeg, c, \theta^{EC}, \theta^{EV}\right\}$$

The structure *theta* will then be passed on to procedures, where the parameters are then "unpacked" as needed.

SECTION 3 (Creating the State Space): This section calls the procedure *sigma_algebra.src* to create the $nums \times (1 + IL)$ matrix *state*. This matrix lists the entire state space of the economy. The first column lists the *index* of which of the possible *nump* realizations of ϕ_t the state is at. The 2^{nd} through the $(1 + IL)^{th}$ columns list which *spatial market structure* $\mathbf{n}_t$ the state is at. Note that the size of the state space is $nums = nump * 2^{IL}$. The user does not need to specify any parameters in this section, just call the procedure.

Example: Let $nump = 2$, $I = 2$, and $L = 3$. *(2 possible values for* ϕ_t*, 2 firms, and 3 feasible business locations). Then:*

$$state = \begin{pmatrix} & & \textit{Firm 1} & & & \textit{Firm 2} & \\ \phi_t & \textit{Location 1} & \textit{Loc. 2} & \textit{Loc. 3} & \textit{Loc. 1} & \textit{Loc. 2} & \textit{Loc. 3} \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 2 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

and $nums = 128$. ■

SECTION 4 (Computing the Price Equilibrium(a)): This section calls the procedure *spatial_ bertrand.src* to compute the price equilibrium(a) at each possible realization of the state space. The user does not need to specify any parameters in this section, just call the procedure. The procedure will call other auxiliary procedures throughout the computation of equilibrium prices. The default initial prices are set equal to marginal costs for each firm ($\mathbf{p}^0 = c$), although this specification can be easily changed inside the procedure. Besides computing the $nums \times IL$ matrix of equilibrium prices $\mathbf{p}$ at each store, the procedure returns other features of the equilibrium, such as the $nums \times IL$ matrices of aggregate demands (Ad) and variable profits (vp) at each active store, the $nums \times 1$ vectors of consumer surplus (cs) and total transportation costs (tc) at each state, and the $nums \times IL \times IL$ array of elasticities of demand with respect to price across active stores (Edp). More details on *spatial_ bertrand.src* are given below.

SECTION 5 (Computing the MPE in Location): This section calls the procedure *spatial_ mpe.src* to compute the Markov Perfect Equilibrium (MPE) in the location game. The user does not need to specify any parameters in this section, just call the procedure. The procedure will call other auxiliary procedures throughout the computation of the MPE. The procedure returns the $I \times nums \times IL$ array of equilibrium conditional choice probabilities ($Palpha = P^*$) with the decision rule in probability space, the $nums \times nums$ matrix of equilibrium transition matrix (tr_state) for the state, and the $nums \times 1$ vector of probabilistic steady-state of the industry ($psteady$). More details on *spatial_ mpe.src* are given below.

3 Price Equilibrium Procedure (*spatial_ bertrand.src*)

This procedure has the following format:

$$\{p, ad, vp, cs, tc, edp\} = spatial\_bertrand(theta, state)$$

Therefore its inputs are the structure of primitives *theta* and the state space *state*, and the outputs are equilibrium prices, aggregate demands, variable profits, total consumer surplus, total trans-

portation costs, and elasticities of demand with respect to price across active stores. Although the user does not need to specify any parameter inside the procedure, the parameters used at the procedure are related to the paper as follows:

Paper	Program	Dimension	Description
$\mathbf{p}^0$	$p0$	$nums \times IL$	Matrix of initial prices for fixed-point
$\sigma(z, \mathbf{n}_t, \mathbf{p}_t)$	Ld	array	Array of Local Demands at a state
$s(\mathbf{n}_t, \mathbf{p}_t)$	Ad	$nums \times IL$	Matrix of Aggregate Demands at a state
$\Lambda_{i\ell}^{i'\ell'}$	m_up	$L \times L$	Used at $\Delta(\mathbf{p})$ to compute price mark-up
$\mathbf{p}^*$	p	$nums \times IL$	Matrix of equilibrium prices
$R_i(\mathbf{n}, \phi)$	vp	$nums \times IL$	Matrix of equilibrium Variable Profits
$CS(\mathbf{n}_t, \mathbf{p}_t, \phi_t)$	cs	$nums \times 1$	Vector of Consumer Surplus at each state
$TC(\mathbf{n}_t, \mathbf{p}_t, \phi_t)$	tc	$nums \times 1$	Vector of Transport Costs at each state
Elasticity of Demand	Edp	$nums \times IL \times IL$	Array of Elast. of Subst. among stores

After "unpacking" the necessary parameters, the procedure is divided into 3 sections.

SECTION 1 (Nodes and Weights Used at Integration over $\mathbb{C}$): This section calls the procedure *spatial_grid.src* which returns the node points and the weights from the specification of market $\mathbb{C}$ to be used at integration. More specifically, it returns the $1 \times ncell[1,1]$ vector of the x-coordinates for the node points ($xnod$), the $ncell[1,2] \times 1$ vector of the y-coordinates for the node points ($ynod$), and the $ncell[1,2] \times ncell[1,1]$ matrix of weights for each node point from crossing $xnod$ with $ynod$. In the process this procedure also calls another procedure, *pdf_bn.src*, which computes the bivariate normal density at the specified node points (which are in turn used to compute the weights).

SECTION 2 (Computation of $\mathbf{p}^*$): Given an initial $nums \times IL$ matrix of prices for the state space ($\mathbf{p}^0$), this section computes the price equilibrium using a Gauss-Seidel method. The default value for initial prices is marginal costs ($\mathbf{p}^0 = c$). The user may change this initial condition, as well

as the convergence criterion (*convc*). This section calls two auxiliary procedures: (i) *spatial_ Ld.src* for computing an array of local demands ($\sigma(z, \mathbf{n}_t, \mathbf{p}_t)$) given a market structure and price vector; and (ii) *iv.src* for integrating over the Market $\mathbb{C}$. Finally, the procedure *spatial_ Ld.src* calls a third procedure, *dist_ p.src*, used to compute linear distances between a store and each representative consumer across the market $\mathbb{C}$.

SECTION 3 (Computation of Equilibrium Features): Given the equilibrium price vector $\mathbf{p}^*$, this section computes the $nums \times IL$ matrices of aggregate demands (Ad) and variable profits (vp) for each active store, the $nums \times 1$ vectors of total consumer surplus (cs) and total transportation costs (tc) at each state, and the $nums \times IL \times IL$ array of elasticities of demand with respect to price across active stores (Edp) at each state.

4 MPE Procedure (*spatial_ mpe.src*)

This procedure has the following format:

$$\{Palpha, ccvalue, tr\_state, psteady\} = spatial\_mpe(theta, state, vp)$$

Therefore its inputs are the structure of primitives *theta*, the state space *state*, and the equilibrium variable profits *vp*. The outputs are the $I \times nums \times IL$ array of equilibrium conditional choice probabilities $\left(Palpha = P^{\alpha^*}\right)$ as the decision rule in probability space, the $nums \times nums$ matrix of equilibrium transition matrix (tr_state) for the state, and the probabilistic $nums \times 1$ equilibrium vector for the steady-state of the industry ($psteady$). Although the user does not need to specify

any parameter inside the procedure, the parameters used therein are related to the paper as follows:

Paper	Program	Dimension	Description
$\pi_i(a_{it}, \mathbf{n}_t, \phi_t, .)$	*cp*	$I \times nums \times (1+2L)$	Array with observable portion of the Contemporaneous Choice Profit
A	A	$L \times (1+2L)$	Choice set for firms
R	R	scalar	Number of simulations
T	T	scalar	Number of time periods for simulation
ϕ_t^r	*phie*	$nump \times (1+T) \times R$	R simulated processes of length T for ϕ_t
ε_{it}^r	*eps*	$I \times R \times T \times (1+2L)$	R simulated processes of length T for ε_{it}
v^{α^*}	v	$I \times nums \times (1+2L)$	Equilibrium Value functions
P^{α^*}	*Palpha*	$I \times nums \times (1+2L)$	Equil. decision rule in probability space
Transition for state	*tr_state*	$nums \times nums$	Equilibrium transition matrix for *state*
Probabilistic steady-state	*psteady*	$nums \times 1$	Probabilistic steady-state for the industry

After "unpacking" the necessary parameters, the procedure is divided into 8 sections.

SECTION 1 (Some Constants): This section specifies the constants *kcons* (to prohibit infeasible entry or exit choices), *nump* (# of possible values of ϕ_t), and *nums* (size of state space).

SECTION 2 (Contemporaneous Profit Function): This section specifies the common-knowledge part of the contemporaneous profit function, $\pi_i(a_{it}, \mathbf{n}_t, \phi_t, .)$. It creates an $I \times nums \times (1 + 2L)$ array called *cp* ("Contemporaneous Profit"). The first "face" of the array gives the $\pi_i(a_{it}, \mathbf{n}_t, \phi_t, .)$ payoff for player 1, the second "face" for the second player, etc. The first column of the first "face" is the profit for firm 1 if she chooses to do nothing ($a_{1t} = 0$) at each state. The next L columns of the first "face" is the profit for firm 1 if she chooses to *enter* a new store at either location 1, 2, ..., or L ($a_{1t} = \ell_+$). The final L columns of the first "face" is the profit for firm 1 if she chooses to *exit* an existing store at either location 1, 2, ..., or L ($a_{1t} = \ell_-$). The second "face" of the array are the payoffs for firm 2 under the same order, and so on for the remainder firms.

SECTION 3 (Computation of MPE): This section contains the Gauss-Seidel iterative method that computes the equilibrium choice probabilities. At each Gauss-Seidel iteration two procedures are called: *spatial_dp.src* that solves the dynamic programming problem of a single firm; and *spatial_bestp.src* that obtains the best response probabilities of a single firm based on the solution of its dynamic decision problem.

SECTION 4 (Compute *tr_st* and *psteady*): This section computes the $nums \times nums$ matrix of equilibrium transition matrix (tr_s) of the state variables and, based in this transition, the steady-state probability distribution of the state variables.

5 Example (*spatial_example.e*)

The program *spatial_example.e* provides an example.

Appendix: Gauss code

A.1. Program spatial_main.prg

```
/*
** spatial_main.prg
**
** This is the MAIN program of the paper by Aguirregabiria & Vicentini (2006):
** "Dynamic Spatial Competition Between Multi-Store Firms."
** The user first specifies all the parameters of the City-Economy.
** The algorithm then computes the Bertrand equilibrium prices and
** current profits for spatially differentiated products ('stores').
** It then computes the entry/exit/location Markov Perfect Equilibrium (MPE)
** of a Store Location dynamic game by firms in a 2-dimensional
** market with continuously distributed consumers and logit preferences.
**
**
** by Victor Aguirregabiria and Gustavo Vicentini
** University of Toronto and Analysis-Group, respectively
**
** First version: August 2004
** This revision: May 2007
**
** The researcher specifies five types of Primitives:
**
** - parameters of Market 'C'
** - parameters of Feasible Business Locations
** - parameters of individual Consumers
** - parameters of the aggregate Population process
** - parameters of individual Firms
**
** These parameters are then Packed into the parameter vector
** structure called 'theta', which is then passed on to
** the respective procedures.
**
** The program works backwards: starts from the consumer preference maximization, then
** constructs firms' current equilibrium variable profit functions by computing equilibrium
** prices, and then computes the MPE for store location choice of firms.
**
** The procedure 'spatial_bertrand.src' is called to compute the static price equilibrium
** for all possible state spaces. Given these static payoffs, the procedure
** 'spatial_mpe.src' is called to compute the MPE for the dynamic entry/exit/location game.
** Other Auxiliary procedures are also called throughout the algorithm.
**
** Remark 1 : please see the 'spatial_bertrand.src' and 'spatial_mpe.src' procedures
** for details on consumer preferences, profit functions, set-up of state space, etc.
**
** Remark 2 : This main library where all the procedures should be placed is 'spatial.lib'
**
** Remark 3 : For further details on this algorithm, see the Manual:
** "Software for the Computation of Markov-Perfect Equilibria in
** a Dynamic Game of Store Location by Multi-Store Firms,"
** by Victor Aguirregabiria and Gustavo Vicentini, available at the Authors' websites.
**
```

```
*/
new; closeall;
library pgraph spatial ;
struct PVobj { matrix m; array a; matrix fpoff; };
struct PV { scalar np; matrix type; struct PVobj obj;
 matrix table; string array names; };
wdir = "c:\\MYPAPERS\\SPATIAL\\PROGAU\\spatprog";
buff = changedir(wdir) ;
fileout = "spatial_output.out" ;
output file = ^fileout reset ;
format /mb1 /ros 12,4 ;
 @ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
 @ 1. Input Parameters of the Model @
 @ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
 @ ~~~~~~~~~~~~~~~~~~~~~~~~~~~@
 @ 1.1. Market 'C' @
 @ ~~~~~~~~~~~~~~~~~~~~~~~~~~~@
 xaxis = 0~1 ; // Left and Right boundaries of x-axis of Market 'C', respectively
 yaxis = 1|0 ; // Upper and Lower boundaries of y-axis of Market 'C', respectively
 ncell = 40~40 ; // # of grid cells at x-axis and y-axis of Market 'C', respectively
 // (used at integration)
 @ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
 @ 1.2. Feasible Business Locations @
 @ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
 Lnum = 2 ; // number of Feasible Business Locations (= 'L')
 zmat = ( .2 ~.5 ) |
 ( .8 ~.5 ) ; // x and y Coordinates ('z') of all Business Locations, stacked
 @ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
 @ 1.3. Individual Consumer Parameters @
 @ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
 tau = 1 ; // Coefficient on consumer Transporation Costs
 miu = .25 ; // Dispersion Parameter of Consumer heterogeneity
 outs = 0 ; // Utility of purchasing from Outside alternative
 @ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
 @ 1.4. Aggregate Population Distribution ('Phi'), @
 @ and its transition matrix ('tr_p') @
 @ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
 // We assume that 'Phi' is a Bivariate Normal Distribution
 // Each row specifies parameters for one possible 'Phi':
 // Columns 1 and 2: (x,y) coordinates of population mean
 // Columns 3 and 4: Variance of popu. along x-axis and y-axis, respectively
 // Column 5: Covariance of population along x- and y-axis
 // Column 6: Size of Population.
 phi = ( 0.5 ~0.5 ~0.9 ~0.9 ~0 ~4 )|
 ( 0.5 ~0.5 ~1.8 ~1.8 ~0 ~5 )|
 ( 0.5 ~0.5 ~2.0 ~2.0 ~0 ~6 ) ;
 tr_p = ( 0.60 ~0.30 ~0.10) |
 ( 0.20 ~0.60 ~0.20) |
 ( 0.10 ~0.30 ~0.60) ;
 @ ~~~~~~~~~~~~~~~~~~~~~~~~~~~@
 @ 1.5 Parameters of Firms @
 @ ~~~~~~~~~~~~~~~~~~~~~~~~~~~@
 Inum = 2 ; // Number of Potential Firms (= 'I')
 beta = 1/(1+.05) ; // intertemporal discount factor of firms (= 'Beta')
```

```
omeg = 1|1 ; // firms' observable qualities (= 'omega'), common across locations
c = 1|1 ; // firms' marginal costs, common across locations
ec = ( 1 ~1 ) | // First Firm Entry Cost at each location
( 1 ~1 ) ; // Second Firm Entry Cost at each location
ev = ( .5 ~.5 ) | // First Firm Exit Value at each location
( .5 ~.5 ) ; // Second Firm Exit Value at each location

@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
@ 2. Pack all parameters of the Spatial Economy @
@ specified above into a PV structure @
@ called 'theta' @
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
struct PV theta ;
theta = pvCreate ;
{theta} = spatial_pack( theta,
xaxis,yaxis,ncell,
Lnum,zmat,
tau,miu,outs,
phi,tr_p,
Inum,beta,omeg,c,ec,ev ) ;
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
@ 3. Observable State Space 'state.' @
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
state = sigma_algebra(seqa(0,1,2),Inum*Lnum) ; // State Space of Spatial Market Structure ('n')
state = seqa(1,1,rows(phi)).*.ones(2^(Inum*Lnum),1)
~ones(rows(phi),1).*.state ; // Entire Observable State Space ( 'phi' ~'n' )
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
@ 4. Compute Second-Stage Spatial Bertrand price equil. 'p', @
@ total Demands 'ad', equil. variable profits 'vp', @
@ consumer surplus 'cs', transportation costs 'tc', and @
@ elasticity of Demand with respect to price 'Edp'. @
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
{p,ad,vp,cs,tc,edp} = spatial_bertrand(theta,state) ;
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
@ 5 Computes the MPE in Probability Space ('Palpha') for the @
@ dynamic entry/exit/location game, the transition matrix @
@ for the state 'tr_s' = Pr( state[t+1] | state[t] ), and @
@ the probabilistic steady-state equilibrium 'psteady'. @
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
{pia,v,Palpha,tr_s,psteady} = spatial_mpe(theta,state,vp);
output off;
end;
```

A.2. Procedure spatial_bertrand.src

```
/*
** spatial_bertrand.src
**
** Procedure that computes the Price equilibrium 'p' in a static
** Bertrand game with spatially differentiated products and
** random-field type of demand. It also computes the integrated Aggregate
** Demand 'D' and Variable Profits 'R' of each firm, as well as the
** total Consumer Welfare 'cs' and Transportation Costs 'tc' at
** each state space, and the cross Elasticities of Demand
** w.r.t. Price 'Edp' for each store.
** A Gauss-Siedel Fixed-Point search method is used to find the optimal
** prices 'p' given an initial price matrix 'p0'.
**
** by Victor Aguirregabiria and Gustavo Vicentini
** University of Toronto and Boston University, respectively
**
** First version: August 2004
** This revision: November 2006
**
**
** ____________________________________________
** MODEL
** ____________________________________________
**
** i = Firm index;
** n = current network of stores of a firm
**
** Assumption: All Players are "Global" in the city C
** Assumption: Firms may open, close or relocate at most one store per period
** Assumption: Consumers are truncated Normally at City C
**
** _____
**
** PREFERENCES of a Consumer j with coordinates zj purchasing
** from a firm i with coordinates zi at location l (i.e. A, B, C, or D):
**
** U(zj,zi): omeg[i] - p[i,l] - tau * d(zj,zi) + e[j,i,l]
**
** U(zj,outs): outs + e[j,outs] => Utility from outside alternative
**
** where:
** omeg[i] = firm i's observable quality ('omega')
** p[i,l] = Firm i's price at location l
** tau = consumer's Transportation Costs parameter
** d(zj,zi) = distance between consumer and the firm
** e[j,i,l] = unobservable taste of consumer j for firm i at location l
**
** Remark: this utility specification entails a 'random field'
** type of demand
**
** _____
**
```

```
** VARIABLE PROFITS of Firm i with network n:
**
** Not active: Profit(0) = 0
**
** Active: Profit(i,n) = SUM(l locations) { (p[i,l]-c[i,l]) * D(i,l) }
** = R(i,n)
**
** where:
** p[i,l] = firm i's prices at Location l
** c[i,l] = firm i's marginal cost at Location l
** D(i,l) = total demand for firm i at Location l
** R(i,n) = Total Variable Profits for firm i, as in the paper.
**
** Remark: in this second-state multi-store oligopoly game, firms are
** allowed to charge different prices at different locations
**
** ----------------------------------------------------------------
**
** FORMAT:
** { p,D,R,cs,tc,Edp } = spatial_bertrand(theta,state) ;
**
** INPUTS:
**
** theta - this is a PV structure with all the parameter vectors of the
** City economy packed into it. Its member vectors are:
**
** * Parameters for Market 'C':
** xaxis = (1 x 2) left and right boundaries of x-axis of Market 'C', respectively
** yaxis = (2 x 1) upper and lower boundaries of y-axis of Market 'C', respectively
** ncell = (1 x 2) # of grid cells at x-axis and y-axis of the market, respectively (used at integration)
** * Aggregated Population Primitives:
** phi = (nump x 6) Matrix with parametrization of different possible
** realizations of Population process 'phi'.
** (nump = |phi|).
** (Please see the manual for further details).
** tr_p = (nump x nump) transition matrix for Population process 'phi'.
** * Individual Consumers parameters:
** tau = scalar, consumers' Transportation Costs parameter
** miu = St. dev. of consumers unobserved heterogeneity
** outs = Utility of purchasing from Outside alternative
** * Geographic Locations parameters:
** Lnum = scalar - Number of Feasible Business Locations
** zmat = (Lnum x 2) - x and y coordinates of each location
** * Firms parameters:
** Inum = number of potential firms in the market
** beta = Time discount factor
** omeg = (Inum x 1) vector with firms' observable quality
** c = (Inum x 1) vector with firms' variable costs
** ec = (Inum x Lnum) matrix of firms' Entry Costs at each location
** ev = (Inum x Lnum) matrix of firms' Exit Values at each location
**
** state - (nums x 1+Inum*Lnum) matrix listing the entire state space
** (nums = |state| = size of state space).
** (Please see the manual for further details).
```

```
**
**
** OUTPUTS:
**
** p - ( nums x Inum*Lnum ) Matrix with bertrand equilibrium prices for each
** firm and store at each state space.
** Example with 2 firms (Inum=2), 3 locations (Lnum=3), and nump=2 ;
**
** Location: A B C A B C
** Firm: 1 1 1 2 2 2
** Phi: Phi
**
** Row 1: 1 0 0 0 0 0 0
** Row 2: 1 p[1,A] 0 0 0 0 0
** Row 3: 1 0 p[1,B] 0 0 0 0
** Row 4: 1 p[1,A] p[1,B] 0 0 0 0
** Row 5: 1 0 0 p[1,C] 0 0 0
** Row 6: 1 p[1,A] 0 p[1,C] 0 0 0
** Row 7: 1 0 p[1,B] p[1,C] 0 0 0
** Row 8: 1 p[1,A] p[1,B] p[1,C] 0 0 0
** Row 9: 1 0 0 0 p[2,A] 0 0
** Row 10: 1 p[1,A] 0 0 p[2,A] 0 0
** Row 11: 1 0 p[1,B] 0 p[2,A] 0 0
** Row 12: 1 p[1,A] p[1,B] 0 p[2,A] 0 0
** .
** .
** .
** Row : 2 0 0 0 p[2,A] p[2,B] p[2,C]
** Row : 2 p[1,A] 0 0 p[2,A] p[2,B] p[2,C]
** Row : 2 0 p[1,B] 0 p[2,A] p[2,B] p[2,C]
** Row : 2 p[1,A] p[1,B] 0 p[2,A] p[2,B] p[2,C]
** Row : 2 0 0 p[1,C] p[2,A] p[2,B] p[2,C]
** Row : 2 p[1,A] 0 p[1,C] p[2,A] p[2,B] p[2,C]
** Row : 2 0 p[1,B] p[1,C] p[2,A] p[2,B] p[2,C]
** Row nums: 2 p[1,A] p[1,B] p[1,C] p[2,A] p[2,B] p[2,C]
**
**
** D - (nums x Inum*Lnum) Matrix of equilibrium Total Demand faced
** by each firm and store at the equilibrium prices and state space.
**
** R - (nums x Inum*Lnum) Matrix of equilibrium Variable Profits
** of each firm and store at the equilibrium prices and state space.
**
** cs - (nums x 1) Matrix with total Consumer Surplus for each
** state space, computed using the logit assumption.
**
** tc - (nums x 1) Matrix with Average Transport Costs at each state space
**
** Edp - (nums x Inum*Lnum x Inum*Lnum) Array of own and cross Elasticities of
** Demand w.r.t. Price for each firm at each state space.
**
**
** Remark: For further details on the algorithm, see the Manual:
** "Software for the Computation of Markov-Perfect Equilibria in a Dynamic Model of Spatial Competition,"
```

```
** by Victor Aguirregabiria and Gustavo Vicentini, available at the Authors' websites.
**
*/
#include pv.sdf ;
proc (6) = spatial_bertrand(struct PV theta,state) ;
local ncell,zmat,Lnum,tau,miu,outs,phi,Inum,c,
xnod,ynod,weig,
p0,p,convc,crit,iter,
Ld,D,m_up,t,
R,cs,tc,Edp ;
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
@ 0. Read in parameters of the City-Economy @
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
// Market 'C':
ncell = pvunpack(theta,"ncell");
// Locations:
zmat = pvunpack(theta,"zmat");
Lnum = pvunpack(theta,"Lnum");
// Individual Consumers:
tau = pvunpack(theta,"tau");
miu = pvunpack(theta,"miu");
outs = pvunpack(theta,"outs");
// Population:
phi = pvunpack(theta,"phi");
// Firms:
Inum = pvunpack(theta,"Inum");
c = pvunpack(theta,"c");
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
@ 1. Compute 'weights' (= area*density) of Truncated @
@ Bivariate Normal densities at a grid of 'nodes', @
@ to be used to calculate local Demands ('Ld') @
@ and at integration ('iv.src') @
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
{xnod,ynod,weig} = spatial_grid(theta) ;
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
@ 2. Compute Equilibrium Prices (='p') @
@ It uses a Fixed-Point Gauss-Siedel method @
@ based on the FOC. @
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
p0 = state[.,2:cols(state)].*(c'.*.ones(1,Lnum)) ; // initial guess for prices (= 'mc')
convc = (1e-6) ; // Convergence criteria for price

" ";
" ********************************** ";
" ********************************** ";
" BEGIN Bertrand Price Computation: ";
" ********************************** ";
" "; " Price Optimization (Fixed-Point by Gauss-Siedel): "; " ";
p = p0 ;
for s(1,rows(state),1) ; // loop on state space
crit = 30 ; // Initial criteria for price convergence
iter = 1; // iteration on Prices
do while crit>convc ;
for j(1,Inum,1) ; // loop on firms
```

```
Ld = spatial_Ld(theta,p[s,.],state[s,.],xnod,ynod) ; // Local Demands given p
Ld = Ld[.,(j-1)*Lnum+1:j*Lnum,.,.] ; // Local Demands for firm 'j'
D = iv(Ld,weig[state[s,1],.,.]) ; // Aggregate Demands for firm 'j'
m_up = -(1/miu)*iv(Ld.*(Ld-1),weig[state[s,1],.,.]) ; // to be used at new mark-up (SAME-STORE price effect)
m_up = diagrv(zeros(Lnum,Lnum),m_up) ;
for k(1,Lnum,1) ; // loop on stores of firm 'j'
t = k%Lnum+1 ;
do while t/=k ;
m_up[k,t] = - (1/miu)*iv(Ld[.,k,.,.].*Ld[.,t,.,.],weig[state[s,1],.,.]) ; // OTHER-STORES price effect
t = t%Lnum+1 ;
endo;
endfor;
p[s,(j-1)*Lnum+1:j*Lnum] = c[j]*(D.>0) + D*invswp(m_up)' ; // new prices for firm 'j' ('Delta' mapping)
endfor; // end loop on firms
crit = maxc(maxc(abs(p[s,.]-p0[s,.])));
p0[s,.] = p[s,.] ;
iter = iter+1 ;
endo ;
" state= ";; s;; " Converged in ";; iter-1;; "iterations";
endfor; // end loop on state
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
@ 3. Compute Per-Capita 'D', 'R', 'cs', 'tc', 'Edp'. @
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
Ld = spatial_Ld(theta,p,state,xnod,ynod) ; // Equilibrium (per capita) Local Demands
D = iv(Ld,weig) ; // Equilibrium Aggregate Demands
R = (p-c'.*.ones(1,Lnum)).*D ; // Equilibrium Variable Profits
cs = reshape(state[.,2:1+Inum*Lnum],prodc(ncell'),rows(state)*Inum*Lnum)' ; // Consumer Surplus
cs = areshape(cs,rows(state)|Inum*Lnum|ncell[2]|ncell[1]) ;
cs = (1-asum(cs.*Ld,3))./exp(outs/miu);
cs = iv(miu*ln(1./cs),weig) ;
tc = 0*R ; // Transportation Costs
for s(1,rows(state),1); // loop on state space
for j(1,Inum*Lnum,1); // loop on firms and stores
Edp = tau * dist_p( zmat[((j-1)%Lnum)+1,1],zmat[((j-1)%Lnum)+1,2],
xnod,ynod ) ;
Edp = areshape(Edp,1|1|ncell[2]|ncell[1]) ;
tc[s,j] = iv(Ld[s,j,.,.].*Edp,weig[state[s,1],.,.]) ;
endfor;
endfor;
tc = sumc(tc') ; // Equilibrium Transportation Costs
Edp = arrayinit(Inum*Lnum|Inum*Lnum|rows(state)|1,0); // Elast. Demand w.r.t. Price
for s(1,Inum*Lnum,1);
Edp[s,s,.,.] = -(1/miu)*p[.,s].*(1./D[.,s])
.*iv((1-Ld[.,s,.,.]).*Ld[.,s,.,.],weig) ;
t=s%(Inum*Lnum)+1;
do while t/=s ;
Edp[s,t,.,.] = (1/miu).*p[.,t].*(1./D[.,s])
.*iv(Ld[.,t,.,.].*Ld[.,s,.,.],weig) ;
t=t%(Inum*Lnum)+1;
endo;
endfor;
Edp = missrv(Edp,0) ;
Edp = atranspose(Edp,3|4|1|2);
Edp = areshape(Edp,rows(state)|Inum*Lnum|Inum*Lnum); // Equilibrium Elasticity Demand w.r.t. Price
```

```
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
@ 4. Compute NON-Per-Capita 'D', 'R', 'cs', 'tc'. @
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
D = D.*phi[state[.,1],6] ; // Equilibrium Aggregate Demands
R = R.*phi[state[.,1],6] ; // Equilibrium Variable Profits
cs = cs.*phi[state[.,1],6] ; // Equilibrium Consumer Surplus
tc = tc.*phi[state[.,1],6] ; // Equilibrium Transportation Costs
" ";
" ********************************** ";
" END of Bertrand Price Computation. ";
" ********************************** ";
" ********************************** ";
" ";
clear weig,p0,Ld ;
retp(p,D,R,cs,tc,Edp) ;
endp ;
```

A.3. Procedure spatial_pack.src

```
/*
** This procedure packs all the parameters from the Spatial
** Economy into a single PV Structure called 'theta'
**
** Written by: Gustavo Vicentini
** March 2005
**
** Format: {theta} = spatial_pack(theta, all parameters listed ) ;
**
** Input: an empty PV structure called 'theta' and all the
** parameters from the spatial economy listed.
**
** Output: the PV structure 'theta' with all parameters of
** the spatial economy packed and named.
**
**
*/
 #include pv.sdf ;
 proc(1) = spatial_pack(struct PV theta,
 xaxis,yaxis,ncell,
 Lnum,zmat,
 tau,miu,outs,
 phi,tr_p,
 Inum,beta,omeg,c,ec,ev ) ;
 // Market 'C' boundaries and cells:
 theta = pvpack(theta,xaxis,"xaxis");
 theta = pvpack(theta,yaxis,"yaxis");
 theta = pvpack(theta,ncell,"ncell");
 // Business Location Parameters:
 theta = pvpack(theta,Lnum,"Lnum");
 theta = pvpack(theta,zmat,"zmat");
 // Individual Consumer Parameters:
 theta = pvpack(theta,tau,"tau");
 theta = pvpack(theta,miu,"miu");
 theta = pvpack(theta,outs,"outs");
 // Aggregate Population Parameters:
 theta = pvpack(theta,phi,"phi");
 theta = pvpack(theta,tr_p,"tr_p");
 // Firms Parameters:
 theta = pvpack(theta,Inum,"Inum"); theta = pvpack(theta,beta,"beta");
 theta = pvpack(theta,omeg,"omeg"); theta = pvpack(theta,c,"c");
 theta = pvpack(theta,ec,"ec"); theta = pvpack(theta,ev,"ev");
 retp(theta);
 endp;
```

A.4. Procedure sigma_algebra.src

```
/*
** sigma_algebra.src
** This procedure takes an input (Kx1) vector 'x' and computes
** a ( K^j x j ) matrix where each row is all possible
** j-combinations between each element in 'x' to all its
** other elements ==> its 'sigma algebra'.
**
** Written by Gustavo Vicentini, August 2005.
** Department of Economics, Boston University.
**
** Format: {y} = sigma_algebra(x,ncol) ;
**
** Inputs: x = (Kx1) input vector to be expanded into its sigma algebra
** ncol = # of columns (='j') for the expansion of vector 'x'
**
** Output: y = 'sigma algebra' of vector 'x', expanded 'ncol' ('j') times.
**
*/

 proc(1) = sigma_algebra(x,ncol);
 local nrow, y ;
 nrow = rows(x) ;
 y = {} ;
 for i(1,ncol,1);
 y = ones(nrow^(i-1),1).*.x.*.ones(nrow^(ncol-i),1)
 ~y ;
 endfor;
 retp(y);
 endp;
```

A.5. Procedure spatial_grid.src

```
/*
**
** spatial_grid.src
**
** Procedure that computes Weights ('weig') based on the weighted
** AREA and DENSITY of each of the Partitions of the City-Market C
** used in the dynamic network location game from 'Spatial_game.prg'.
** It also returns the center coordinates (the 'nodes') of each
** of the partitions, 'xnod' (x-axis) and 'ynod' (y-axis).
**
**
** by Gustavo Vicentini, Boston University
** First version: August, 2005
**
**
** ----------------------------------------------------------------
**
** FORMAT:
** { xnod,ynod,weig } = spatial_grid(theta) ;
**
** INPUTS:
**
** theta - this is a PV structure with all the parameter vectors of the
** City economy packed into it. Its member vectors are:
** See 'bertrand_spatial.src' for a list of these parameters.
**
** OUTPUTS:
**
** xnod - ( numv x ngr[1] ) Matrix of rows with X-axis coordinates
** of the center of each partition, for each type of Var-Cov.
** ynod - ( ngr[2] x numv ) Matrix of colums with Y-axis coordinates
** of the center of each partition, for each type of Var-Cov.
** weig - ( numv x ngr[2] x ngr[1] ) Array with 'weights' (= area*density)
** of the Partitions of the City-Market C.
**
**
*/
 #include pv.sdf ;
 proc (3) = spatial_grid(struct PV theta) ;
 local xaxis,yaxis,ncell,phi,
 xgr,ygr,area,xnod,ynod,
 weig,den ;
 @ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
 @ 1. Read in parameters of the City-Economy @
 @ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
 // Aggregate Population:
 xaxis = pvunpack(theta,"xaxis"); yaxis = pvunpack(theta,"yaxis");
 ncell = pvunpack(theta,"ncell"); phi = pvunpack(theta,"phi");
 @ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
 @ 2. Compute Weights ('weig') to be used at @
```

```
@ integration of the Trunc. Biva. Normal densities. @
@ Also compute the node points ('xnod' & 'ynod'). @
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
xgr = seqa(xaxis[1],(xaxis[2]-xaxis[1])/ncell[1],ncell[1]+1)' ; // grid for partitions at x-axis
ygr = seqa(yaxis[1],(yaxis[2]-yaxis[1])/ncell[2],ncell[2]+1) ; // grid for partitions at y-axis
area = (xgr[2]-xgr[1])*(ygr[1]-ygr[2]) ; // area of each partition cell
xnod = meanc(trimr(xgr',0,1)'|trimr(xgr',1,0)')' ; // nodes at x-axis for each partition cell
ynod = meanc(trimr(ygr,0,1)'|trimr(ygr,1,0)') ; // nodes at y-axis for each partition cell
weig = arrayinit(rows(phi)|rev(ncell'),0); // 'weights' used at integration
for j(1,rows(phi),1);
den = pdf_bn(xnod,ynod,phi[j,1:2]',phi[j,3:4]',phi[j,5]) ; // Bivariate Normal Density at each node
weig[j,.,.] = (area*den)/sumc(sumc(area*den)) ; // weights at each node (used at integration)
endfor;

clear xgr,ygr,area,den;
retp(xnod,ynod,weig);
endp;
```

A.6. Procedure pdf_bn.src

```
/*
* pdf_bn.src This procedure returns the density values for the
* Bivariate Normal distribution given specified
* mean and Variance-Covariance parameters.
*
* Written by Gustavo Vicentini
* First version: March 2006
* This revision: March 2006
*
* Format: {den} = pdf_bn (xref,yref,mean,var,cov) ;
*
* Inputs:
* xgr - row grid of reference points at x-axis
* ygr - column grid of reference points at y-axis
* mean - (2x1) coordinates of mean of the distribution
* var - (2x1) Variances of x- and y-axis, respectively
* cov - scalar - Covariance of the distribution
*
* Outputs:
* den - density values at the reference points
*
*
*/
 proc(1) = pdf_bn(xref,yref,mean,var,cov) ;
 local sd,cor,den ;
 sd = sqrt(var) ; // standard deviations of x and y
 cor = cov/prodc(sd) ; // correlation between x and y
 xref = (xref-mean[1])/sd[1] ; // standardized x variables
 yref = (yref-mean[2])/sd[2] ; // standardized y variables
 den = (1/(1-cor^2)) * ( xref.^2 + yref.^2 - 2*cor*yref.*xref ) ; // quadratic form
 den = 1/(2*pi*prodc(sd)*sqrt(1-cor^2))*exp(-1/2*den) ; // density points
 retp(den);
 endp;
```

A.7. Procedure spatial_ld.src

```
/*
** spatial_ld.src Procedure that computes the equilibrium LOCAL
** DEMANDS 'Ld' of each representative consumer in the
** city-market C for each firm's branches, given the price vector
** of all firms and the Geographic profile of firms'
** networks (i.e. the locational state space). Firms are
** competing in a Bertrand Spatial competition model in this
** City-Economy.
**
**
** by Gustavo Vicentini
** First version: August, 2004
** This revision: August 2006
**
** ____________________________________________
** MODEL
** ____________________________________________
**
** Please see 'bertrand_spatial.src' for the Economy Model.
**
** ____________________________________________-
** ____________________________________________-
**
**
** FORMAT:
** {Ld} = Ld_spatial (p,aval,theta,xnod,ynod) ;
**
**
** INPUTS:
**
** p - (numa*numv x nf*L) matrix with prices charged at each firm/branch
** aval - (numa x nf*L) matrix with possible network state spaces
** theta - see 'bertrandSpatial.src' for a description of this PV structure
** xnod - points of reference of x-axis
** ynod - points of reference of y-axis
**
**
** OUTPUT:
**
** Ld - (numa*numv x nf*L x ngr[2] x ngr[1]) Array with bertrand equilibrium
** local Demands for each firm (given the state space and a price vector p).
**
**
** Note: when a firm-outlet is inactive, its demand is 0.
**
*/
 #include pv.sdf ;
 proc (1) = spatial_Ld (struct PV theta,p,state,xnod,ynod) ;
 local tau,miu,outs,zmat,Lnum,
 Inum,omeg,Ld,dis ;
```

```
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
@ 1. Read in parameters of the City-Economy @
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
// Individual Consumers:
tau = pvunpack(theta,"tau");
miu = pvunpack(theta,"miu");
outs = pvunpack(theta,"outs");
// Locations:
zmat = pvunpack(theta,"zmat");
Lnum = pvunpack(theta,"Lnum");
// Firms:
Inum = pvunpack(theta,"Inum");
omeg = pvunpack(theta,"omeg");
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
@ 2. Compute local Demands 'Ld' for each state space, @
@ given the price matrix p @
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@

Ld = arrayinit(rows(state)|Inum*Lnum|rows(ynod)|cols(xnod),0); // matrix of local Demands
for s(1,rows(state),1); // loop for State Space
for j(1,Inum*Lnum,1); // loop for each firm and location
if state[s,1+j]==0 ;
dis = zeros(rows(ynod),cols(xnod)); // 'dis' is a Disposable matrix
else;
dis = exp ( 1/miu * ( omeg[ceil(j/Lnum)] - p[s,j] // Consumers indirect utility
- tau * dist_p(zmat[(j-1)%Lnum+1,1],zmat[(j-1)%Lnum+1,2],
xnod,ynod))) ;
endif;
setarray Ld,s|j,dis ;
endfor;
dis = asum(Ld[s,.,.,.],3)+exp(outs/miu);
dis = areshape(dis,1|Inum*Lnum|rows(ynod)|cols(xnod));
Ld[s,.,.,.] = Ld[s,.,.,.]./dis;
endfor;
clear p,state,xnod,ynod,dis ;

retp(Ld);
endp;
```

A.8. Procedure dist_p.src

```
/*
** dist_p.src This procedure computes the linear distance ON A PLANE
** between a given point 'z' and the plane point(s) in 'xnod' and 'ynod'.
** Note that 'z' is a single point, while 'nod' can be many points.
**
** Written by Gustavo Vicentini
** Department of Economics, Boston University
** August 2005
**
** Inputs: Zx = x-coordinates of point Z
** Zy = y-coordinates of point Z
** xnod = x-coordinates of point(s) 'nod' ==> a row vector (x grid)
** ynod = y-coordinates of point(s) 'nod' ==> a column vector (y grid)
**
** Output: distance 'dist2' between 'Z' and the 'nods'
**
*/
 proc (1) = dist_p(Zx,Zy,xnod,ynod) ;
 local dist2 ;

 dist2 = (Zx-xnod).^2 ;
 dist2 = dist2 + (Zy-ynod).^2 ;
 dist2 = sqrt(dist2) ; // linear distance
 clear Zx,Zy,xnod,ynod ;
 retp(dist2) ;
endp ;
```

A.9. Procedure iv.src

```
/*
* iv.src This procedure returns the Integrated Value (='intv')
* for a function 'funct' in the city C. It applies a
* simple Gaussian Quadrature method by multiplying
* 'funct' (the function to be integrated) by the
* 'weight' (=area*density) at specified Node points.
* See Judd (1995, page 257)
*
* Written by Gustavo Vicentini
* First version: August 2004
* This revision: March 2006
*
* Format: {intv} = iv (funct,weight) ;
*
* Inputs:
* funct - ((numa*numv) x (nf*L) x ngr[2] x ngr[1]) -
* values of the function to be integrated in the city
* weig - ( numv x ngr[2] x ngr[1] ) Array with 'weights' (= area*density)
* of the Partitions of the City-Market C.
*
* Outputs:
* intv - ((numa*numv) x (nf*L)) - integrated value at each state
* space and firm/location
*
*
*/
 proc (1) = iv(funct,weig);
 local a,b,intv ;
 a = getorders(funct) ;
 b = getorders(weig) ;
 weig = areshape(weig,a[1]*a[2]/b[1]|b[1]|a[3]|a[4]) ;
 weig = atranspose(weig,2|1|3|4);
 weig = areshape(weig,a[1]|a[2]|a[3]|a[4]);
 intv = asum(asum(funct.*weig,2),1);
 intv = arraytomat(areshape(intv,a[1]|a[2]));
 clear funct,weig ;
 retp (intv);
 endp;
```

A.10. Procedure spatial_mpe.src

```
/*
** spatial_mpe.src
**
** This procedure computes players' Markov Perfect Equilibrium (MPE)
** strategies in Probability Space, and the corresponding Value Functions.
** The MPE is for firms' entry/exit/location choice under incomplete information.
** The game is one of firms' entry, exit and location choice under incomplete
** information, combined with Spatial Bertrand Competition in
** differentiated products, in a city-market C with continuous population.
** It uses Gauss-Jacobi fixed-point iteration procedure for the
** computation of value functions.
**
** by Victor Aguirregabiria and Gustavo Vicentini
** University of Toronto and Analysis Group, respectively
**
** First version: December 2005
** This revision: May 2007
**
** ____________________________________________
** DETAILS OF PROGRAM
** ____________________________________________
**
** FORMAT:
**
** { pia, ccvalue, Palpha, tr_s, psteady }
** = spatial_mpe ( theta, state, R ) ;
**
**
** INPUTS:
**
** theta - this is a PV structure with all the parameter vectors of the
** City economy packed into it. Its member vectors are:
**
** * Parameters for Market 'C':
** xaxis = (1 x 2) left and right boundaries of x-axis of Market 'C', respectively
** yaxis = (2 x 1) upper and lower boundaries of y-axis of Market 'C', respectively
** ncell = (1 x 2) # of grid cells at x-axis and y-axis of the market, respectively (used at integration)
** * Aggregated Population Primitives:
** phi = (nump x 6) Matrix with parametrization of different possible
** realizations of Population process 'phi'.
** (nump = |phi|); (Please see the manual for further details).
** tr_p = (nump x nump) transition matrix for Population process 'phi'.
** * Individual Consumers parameters:
** tau = scalar, consumers' Transportation Costs parameter
** miu = St. dev. of consumers unobserved heterogeneity
** outs = Utility of purchasing from Outside alternative
** * Geographic Locations parameters:
** L = scalar - Number of Feasible Business Locations
** z = (L x 2) - x and y coordinates of each location
** * Firms parameters:
```

```
** I = number of potential firms in the market
** beta = Time discount factor
** omeg = (I x 1) vector with firms' observable quality
** c = (I x 1) vector with firms' variable costs
** ec = (I x L) matrix of firms' Entry Costs at each location
** ev = (I x L) matrix of firms' Exit Values at each location
**
** state - (nums x 1+I*L) matrix listing the entire state space
** (nums = |state| = size of state space).
** (Please see the manual for further details).
**
** R - (nums x I*L) Matrix of equilibrium Variable Profits
** of each store at each state space.
**
**
** OUTPUTS:
**
** pia - (I x nums x (1+2*L)) array with Obervable Portion of Current
** Profit function of Firms (the 'pi(a)' function in the paper,
** for each Player, State and Action
**
** ccvalue - (I x nums x (1+2*L)) array with conditional choice Value Functions
** for each Player, State and Action
**
** Palpha - (I x nums x (1+2*L)) array with MPE Conditional Choice
** Probability strategies (CCP) for each firm, state, and action.
** The first 'face' of the array reflects the optimal strategies for
** firm 1 for either doing nothing, opening a store somewhere,
** or closing an active store.
** (please see 'Spatial_Manual.pdf' for further details.)
**
** tr_s - (nums x nums) matrix with equilibrium transition matrix
** for the states { phi[t], Market Structure[t] }
**
** psteady - (nums x 1) vector with steady-state distribution
** of states { phi[t], Market Structure[t] }
**
**
** REMARK: For further details on the algorithm, see the Manual:
** "Software for the Computation of Markov-Perfect Equilibria in a Dynamic Model of Spatial Competition,"
** by Victor Aguirregabiria and Gustavo Vicentini, available at the Authors' websites.
**
**
*/
#include pv.sdf ;

proc (5) = spatial_mpe(struct PV theta,state,R) ;

local Lnum, tr_p, Inum, beta, ec, ev,
kcons, nump, nums, numa, pia,
Palpha, crit, iter, Palpha0, tr_s,
i, ccvalue, buffpi, psteady0, psteady ;
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
@ 0. Read in parameters of the City-Economy @
```

```
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
// # of Locations:
Lnum = pvunpack(theta,"Lnum");
// Markov Transition Matrix for Aggregate Population Distribution ('phi'):
tr_p = pvunpack(theta,"tr_p");
// Firms:
Inum = pvunpack(theta,"Inum"); beta = pvunpack(theta,"beta");
ec = pvunpack(theta,"ec"); ev = pvunpack(theta,"ev");
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
@ 1. Some Constants & Parameters @
@ used through the procedure @
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
kcons = -(1 + abs(maxc(maxc(R))))*1e6 ; // payoff for infeasible choices
nump = rows(tr_p) ; // # of possible realizations of 'phi'
nums = rows(state) ; // size of state space
numa = 1 + 2*Lnum ; // Number of choice alternatives
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
@ 2. Conditional-Choice Current Profits (= 'Pi(a)'). @
@ This is the 'Pi(a)' profit in the paper. @
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
R = R * (eye(Inum).*.ones(Lnum,1)) ; // Equilibrium Variable Profits aggregate over locations
R = reshape(R',nums*Inum,1) ;
pia = areshape(state[.,2:1+Inum*Lnum]',Inum|Lnum|nums) ;
pia = atranspose(aconcat(pia,pia,2),1|3|2) ;
pia = arraytomat(areshape(pia,1|Inum*nums|2*Lnum)) ;
pia = (pia.==(1~0).*.ones(1,Lnum))*kcons ; // impose penalty for non-feasible actions
pia = pia + (-ec.*.ones(nums,1)~ev.*.ones(nums,1)) ; // entry costs and exit values
pia = areshape(R~(R+pia),Inum|nums|2*Lnum+1) ; // Add Equilibrium Variable Profits
@ ~~~~~~~~~~~~~~~~~~~~@
@ 3. Computation of MPE @
@ ~~~~~~~~~~~~~~~~~~~~@
"**********************************" ;
"**********************************" ;
" BEGIN MPE Computation:" ;
"**********************************" ;
// Initial choice probabilities
Palpha = arrayinit((Inum|nums|(1+2*Lnum)),0);
i=1 ;
do while i<=Inum ;
Palpha[i,.,.] = ones(nums,1)~zeros(nums,2*Lnum) ;
i=i+1 ;
endo ;
crit = 1000 ; // Initial Criterium
iter = 1 ;
do while crit>1e-6 ; // Outer loop
Palpha0 = Palpha ;
i=1 ;
do while i<=Inum ; // Inner loop
// Calling procedure to compute conditional choice values
ccvalue = spatial_dp(i,arraytomat(pia[i,.,.]),beta,state,Palpha,tr_p) ;
// Calling procedure to compute best response probabilities
buffpi = spatial_bestp(ccvalue) ;
Palpha[i,.,.] = buffpi ; // Updating probabilities
i=i+1 ;
```

```
endo ;
// Check for Convergence of 'Palpha'
crit = arraytomat(maxc(areshape(abs(Palpha-Palpha0),Inum*nums*numa|1))) ;
" iter:";; iter;; " Criterion = ";; crit;
iter=iter+1;
endo;
"***********************************" ;
"***********************************" ;
" END MPE Computation:" ;
"***********************************" ;

@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
@ 4. Vector of Equilibrium Steady-State 'psteady'. @
@ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~@
tr_s = spatial_tranp(Palpha,tr_p,state) ;
crit = 30 ;
psteady0 = (1/nums) * ones(nums,1) ;
do while crit>(1e-5) ;
psteady = tr_s' * psteady0 ;
crit = maxc(abs(psteady-psteady0)) ;
psteady0 = psteady ;
endo ;
retp(pia,ccvalue,Palpha,tr_s,psteady) ;
endp ;
```

A.11. Procedure spatial_dp.src

```
/*
** spatial_dp.src
**
** This procedure takes as inputs of firms' profits and
** choice probabilities and it solves the Bellman equation
** of an individual firm
**
** by Victor Aguirregabiria and Gustavo Vicentini
** (University of Toronto) (Analysis-Group)
**
** This revision: May 2007
**
** ____________________________________________
** MODEL
** ____________________________________________
** See Aguirregabiria and Vicentini (2007) "Dynamic Spatial Competition
** Between Multi-Store Firms"
**
** ____________________________________________
** DETAILS OF PROGRAM
** ____________________________________________
**
** FORMAT:
**
** ccv = spatial_dp(i,profit,beta,state,Palpha,tr_p) ;
**
** INPUTS:
**
** i - Firm index (an integer between 1 and Inum)
**
** profit - nums x numa matrix of current profits
** (nums = |state| = size of state space)
** (numa = 1 + 2*Lnum = size of action space)
**
** beta - Discount factor
**
** state - nums x (1 + Inum*Lnum) matrix with the value of the
** state variables (columns) at every state (rows).
**
** Palpha - Inum x nums x numa array of conditional choice
** transition probabilities
** (nums = |state| = size of state space)
** (numa = 1 + 2*Lnum = size of action space)
**
** tr_p - nump x nump matrix of transition probabilities of phi
**
** OUTPUTS:
**
** ccv - nums x numa matrix of choice specific values
** (nums = |state| = size of state space)
```

```
** (numa = 1 + 2*Lnum = size of action space)
**
*/

 proc (1) = spatial_dp(i,profit,beta,state,Palpha,tr_p) ;

 local ordprob, Inum, nums, numa, value0, ccval, criter,
 cconv, j, value1 ;
 ordprob = getorders(Palpha) ;
 Inum = ordprob[1] ;
 nums = ordprob[2] ;
 numa = ordprob[3] ;
 value0 = zeros(nums,1) ;
 ccval = zeros(nums,numa) ;
 criter = 1000 ;
 cconv = 1e-6 ;
 do while criter>=cconv ;
 "Firm";; i;; "Value Function itertion" ;
 // Calling procedure for expected next period value
 ccval = spatial_ccvalue(i,state,Palpha,tr_p,value0) ;
 // Value function iteration
 value1 = ln(sumc(exp(profit + beta*ccval)')) ;
 // Check for convergence
 criter = maxc(abs(value1-value0)) ;
 value0 = value1 ;
 endo ;
 retp(ccval) ;
 endp ;
```

A.12. Procedure spatial_ccvalue.src

```
/*
** spatial_ccvalue.src
**
** This procedure takes as inputs firms' choice probabilities
** and a value function and it returns expected next period
** values conditional on current decision.
**
** by Victor Aguirregabiria and Gustavo Vicentini
** (University of Toronto) (Analysis-Group)
**
** This revision: May 2007
**
** ____________________________________________
** MODEL
** ____________________________________________
** See Aguirregabiria and Vicentini (2007) "Dynamic Spatial Competition
** Between Multi-Store Firms"
**
** ____________________________________________
** DETAILS OF PROGRAM
** ____________________________________________
**
** FORMAT:
**
** ccv = spatial_ccvalue(i,state,Palpha,tr_p,value0) ;
**
** INPUTS:
**
** i - Firm index (an integer between 1 and Inum)
**
** state - nums x (1 + Inum*Lnum) matrix with the value of the
** state variables (columns) at every state (rows).
**
** Palpha - Inum x nums x numa array of conditional choice
** transition probabilities
** (nums = |state| = size of state space)
** (numa = 1 + 2*Lnum = size of action space)
**
** tr_p - nump x nump matrix of transition probabilities of phi
**
** value0 - nums x 1 vector of values
**
** OUTPUTS:
**
** ccv - nums x numa matrix of expected next period values
** conditional of current decision.
** (nums = |state| = size of state space)
** (numa = 1 + 2*Lnum = size of action space)
**
*/
```

```
proc (1) = spatial_ccvalue(i,state,Palpha,tr_p,value0) ;
local ordprob, Inum, nums, numa,
nump, ccv, aj, pbuff, tr_s ;
ordprob = getorders(Palpha) ;
Inum = ordprob[1] ;
nums = ordprob[2] ;
numa = ordprob[3] ;
nump = rows(tr_p) ;
ccv = zeros(nums,numa) ;
aj = 1 ;
do while aj<=numa ;
pbuff = Palpha ;
// Replace firm i choice probabilities by deterministic choice
if (aj==1) ;
pbuff[i,.,.] = ones(nums,1)~zeros(nums,numa-1) ;
elseif (aj>1)and(aj<numa) ;
pbuff[i,.,.] = zeros(nums,aj-1)~ones(nums,1)~zeros(nums,numa-aj) ;
elseif (aj==numa) ;
pbuff[i,.,.] = zeros(nums,numa-1)~ones(nums,1) ;
endif ;
// Calling procedure to calculate transition probabilities
// Obtain expected next period values
tr_s = spatial_tranp(pbuff,tr_p,state) ;
ccv[.,aj] = tr_s*value0 ;
aj=aj+1 ;
endo ;
retp(ccv) ;
endp ;
```

A.13. Procedure spatial_tranp.src

```
/*
** spatial_tranp.src
**
** This procedure takes as input firms' choice probabilities
** and transition probabilities of phi and it returns a matrix
** of transition probabilities for market structure and phi
**
** by Victor Aguirregabiria and Gustavo Vicentini
** (University of Toronto) (Analysis-Group)
**
** This revision: May 2007
**
** ____________________________________________
** MODEL
** ____________________________________________
** See Aguirregabiria and Vicentini (2007) "Dynamic Spatial Competition
** Between Multi-Store Firms"
**
** ____________________________________________
** DETAILS OF PROGRAM
** ____________________________________________
**
** FORMAT:
**
** tr_s = spatial_tranp(Palpha,tr_p,state)
**
** INPUTS:
**
** Palpha - Inum x nums x (1+2*Lnum) array of choice probabilities
**
** tr_p - nump x nump matrix of transition probabilities of phi
**
** state - nums x (1+Inum*Lnum) matrix with the values of the state
** variables (columns) at every state (rows).
**
** OUTPUTS:
**
** tr_s - nums x nums matrix of transition probabilities for
** phi and market structure.
**
*/

proc (1) = spatial_tranp(Palpha,tr_p,state) ;

local ordprob, Inum, Lnum, nums, numa, nump,
amat, tr_s, s, j, sj, tr_sj, aj, ak ;
ordprob = getorders(Palpha) ;
Inum = ordprob[1] ;
nums = ordprob[2] ;
numa = ordprob[3] ;
```

```
Lnum = (numa-1)/2 ;
nump = rows(tr_p) ;
amat = zeros(1,Lnum)|(1|-1).*.eye(Lnum) ; // choice set of firms
tr_s = ones(nums,nums) ;

s=1 ;
do while s<=nums ; // loop on state
j=1 ;
do while j<=Inum ; // loop on players
sj = state[.,2+(j-1)*Lnum:1+j*Lnum] ; // state of player 'j'
tr_sj = 0 ;
aj=1 ;
do while aj<=(1+2*Lnum) ; // loop on actions
// ak is a nsum x 1 vector with zeros at every positiion
// except at those positions (states) for which the network
// of firm j at t+1 is reached from state s and choice aj
ak = prodc((sj-sj[s,.]==amat[aj,.])') ;
// tr_sj is a 1 x nsum vector with the probabilities that
// that firmj reaches the correspoinding next period state
// given that the current state is s
tr_sj = tr_sj+ak'.*arraytomat(Palpha[j,s,aj]).^(ak') ;
aj=aj+1 ;
endo ;
tr_s[s,.] = tr_s[s,.].*tr_sj ;
j=j+1 ;
endo ;
s=s+1 ;
endo ;
tr_s = tr_s.*(tr_p.*.ones(nums/nump,nums/nump)) ;

retp(tr_s) ;
endp ;
```