Implementing the Pakes–McGuire Algorithm for Computing Markov Perfect Equilibria in Gauss

Ariel Pakes, Gautam Gowrisankaran and Paul McGuire

Department of Economics, Yale University

June 7, 1993

• Overview

This paper has been written to accompany computer programs which provide Gauss code to compute Markov Perfect Nash (MPN) Equilibria. These programs can accessed by ftping to "econ.yale.edu", entering "anonymous" as login and the actual login name for password, and retrieving all files from the folder "pub/mrkv–eqm". We ask that anyone accessing that program send his or her name and adress by electronic mail to "mrkv–eqm@econ.yale.edu"; users may also send any questions or comments to this address. This will allow us to keep track of how useful our experiment in providing computer code has been, and, in addition, inform users so that they may update their copies when major refinements in our program become available.

The program computes the MPN equilibria (Maskin and Tirole, 1988a and b) generated by the restrictions enumerated in Ericson and Pakes (1993). In particular, it implements the algorithm developed to compute those equilibria in Pakes and McGuire (1993), and the user might find a copy of the Pakes and McGuire paper to be helpful. The program provided here has been developed with ease of use, in contrast to speed, in mind. The original Pakes and McGuire paper used a program written in C and used a series of computational shortcuts not used here. Consequently, the computation times reported in that article are one to two orders of magnitude smaller than those the user should expect from the program made available here.

If we start the program provided here from "scratch" (at the default initial condition for the recursive fixed point calculation) on a Sun SPARCStation 1 (and probably also a 486), it typically takes five to ten minutes to compute equilibria when the maximum number of active firms is 2, 4 hours when the maximum number of firms is 3, and 30 to 40 hours when the maximum number of firms is 4. However, in many cases, the user will be interested in comparing the equilibrium generated by different parameter values. For this type of problem, we allow the user to start the computation at the solved values for another set of parameter values and this can decrease computational times dramatically (we explain how this is done in Section 7).

The major reason for the huge increase in computation time over that reported in Pakes–McGuire is that we are using the Gauss matrix–based programming language, which is much slower than a compiled language such as C. Additionally we have not incorporated optimization methods which could either; i) detract from the clarity of the programming (and thus deter the user from being able to understand the programs), or ii) which we have found that sometimes lead to convergence problems (even though in the majority of cases they decrease computational times without leading to convergence problems; see below).

We note that we are currently working on algorithms that should be simpler, faster, and much less demanding in terms of memory; we will provide them when available and appropriately debugged. However, the large volume of requests for this program, together with our track record in completing projects on time have convinced us to distribute the program at this time using the current versions of the algorithms. In spite of the performance detriment to using Gauss, we have chosen to distribute the Gauss version of the program because the Gauss language is relatively easy to understand, and is used by many economists. Thus, we hope that by providing a program which can be easily understood and modified, we are allowing students to investigate many rich and interesting details of Industrial Organization structures.

• Section 1: The Model

As in Ericson–Pakes and Pakes–McGuire, a firm's profits are a function of its own level of efficiency and a vector specifying the efficiency levels of its competitors. The vector containing the number of firms at each different efficiency level will be termed the "industry structure". In principle "efficiency" may be a vector valued concept, but currently the algorithm is only set up to handle scalar valued efficiencies. The efficiencies of all competitors evolve over time with the stochastic outcomes of their investment expenditures.

Investment, entry and exit decisions are chosen to maximize the expected discounted value (EDV) of future net cash flows conditonal on the current information set. To make its decisions, each firm requires perceptions of the distribution of the future efficiencies of its competitors, or equivalently of the future industry structures, conditional on the current information set. A given perception determines policies, which in turn determine the true distribution of industry structures for the next period. Equilibrium occurs when the perceived distribution of the efficiencies of its competitors in the future is, in fact, the distribution generated by those perceptions. Thus in equilibrium, the perceived distribution of future industry structures is equal to the actual distribution of future industry structures generated by the actions of all (potential and actual) competitors when those actions are chosen to maximize the EDV of future net cash flows resulting from those perceptions. The algorithm is, therefore, computing a subgame–perfect Nash equilibrium where the choice variables are the amount of investment combined with the binary exit and entry options.

The algorithm takes as input a one–period static profit function and outputs the optimal policies (entry, exit and investment decisions) and the value function at each possible state the firm may find itself in. The main components of the algorithm are discussed in Section 2 of this paper. The possible states consist of couples, the first element

of which is a vector providing the industry structure, and the second element of which is a scalar which lists which one of the firms that we have listed we are considering. The algorithm allows for different profit functions to be used, in order to allow the modeling of different industrial environments.

Also, separate programs are developed which input the profit function from an analogous technological environment but assuming; i) a social planner, and ii) a perfect cartel, makes all decisions. These single–agent problems are discussed in Section 5. These programs allow the user to compare the planner and the cartel solutions to the MPN equilibria.

Finally we provide programs designed to simulate the evolution of the industry given any one of these environments, and analyze both the descriptive and welfare aspects of the outputs from those simulations.

PROFIT FUNCTIONS.

We have, to date, used three basic profit functions in our calculations. They are:

i) a homogeneous products industry where equilibrium is Nash in quantities and firms have differing marginal costs (here the marginal cost differences become our "efficiency" differences; a particular parameterization of this profit function is used as the example in Ericson–Pakes);

ii) a homogeneous products industry where equilibrium is Nash in quantities and firms have identical marginal cost up to a capacity which differs across firms (here the capacities becomes our efficiency difference; a particular parameterization of this profit function is used as the example in Berry and Pakes, 1993);

iii) a differentiated products industry in which equilibrium is Nash in prices and the quality of the products marketed differs across firms but marginal costs do not (here the qualities become our efficiency differences; different parameterizations of this profit function are

used extensively in Pakes–McGuire).

These profit functions have been used intensively in the I.O. literature. We provide programs which construct types i) and iii) for different parameter values in Section 4. It should be easy for the user to construct other examples of profit functions and input them into our algorithm.

BASIC STRUCTURE.

Ericson–Pakes provide conditions which insure that:

i) there are a finite number of efficiency levels that any firm will ever find itself at, and

ii) only a finite number of firms that will ever be simultaneously active.

This provides us with a finite state space (if $\overline{w}$ is the number of efficiency levels, and N is the maximum number of active firms, then $\overline{w}^N$ is an upper bound to the cardinality of the set of distinct industry structures).

We assume that the horizon is infinite (though firms will not be infinitely lived) and that time is discrete. As specified by the Markov Perfect criterion, strategies are restricted to be functions of the current values of the "payoff relevant" random variables (see Maskin and Tirole, 1988a and b). For each distinct industry structure the program outputs the entry, exit, and investment decisions of all incumbents and potential entrants, and the EDV of the future net cash flows of agents who follow these policies.

We start by defining the industry structures that firms may find themselves at.

DEFINITIONS.

(1) W is the set of efficiency values for an individual firm. Typically, W = $\{0,...,\overline{w}\}$. The definition of "0" and of "$\overline{w}$" depend on the parameters of the problem, and we will show how to compute them below.

(2) N is the maximum number of firms that can simultaneously be active in an

industry. N depends on the parameters of the problem and we will also show how to compute it below.

(3) A state [w,n] consists of a $w \in W^*$, $n \in N$, where:

$$W^* = \{(w_1, ..., w_N) \mid w_i \in W, \ w_1 \geq w_2 \geq ... \geq w_N\}.^1$$

For any firm, w represents the industry structure vector faced by the firm, while n represents which element of this vector is the efficiency level of that firm. We can represent the industry structure as a weakly descending N–tuple (as in W^*) because two industries are equivalent regardless of the order in which the firms in that industry are enumerated. Thus, the state w = [(7,5,5,2),1] and w' = [(2,5,7,5),3] are equivalent as the firm at each state is faced with the exact same present conditions (the firm does not care in what order its competitors are listed). Thus, we do not have to calculate separate value and policy functions for w and w', because any firm at either of these two states faces the same economic environment and will choose the same policies. Formally, the value function is exchangeable in the components of the state vector of the competitors, as is discussed in more detail in Pakes (forthcoming). We use this exchangeability to reduce the dimensionality of the fixed point calculation needed for the computation of optimal policies. We have developed a coding method based on binomial coefficients that uniquely maps from W^* into the natural numbers $\mathbb{N}$. We describe and motivate this map and its

[1]An alternate definition of a state is a pair [k,w], where $k \in W$, and w is a weakly descending *N–1 tuple*. In this definition, k is the own firm's efficiency level, while w is the competitors' efficiency levels. This formulation actually results in fewer states (the savings are about 20% for sizes we have used), as states such as [(9,7,7,2),2] and [(9,7,7,2),3] are combined into one state [7,(9,7,2)]; the exposition of the algorithm is also notationally easier with this method. The problem with this formulation is that for some specifications it seemed to have trouble with convergence when N was large. We postulate that this might be because we are forcing these two states to have the same exit decisions at each iteration (which causes cyclic behavior with both being active and both being inactive in alternate iterations). Conditional on remaining active, the two firms must invest the same, but we allow them to adopt different exit decisions (see below).

inverse in Section 3.

We also note that our definition of a state is different from the definition used in Pakes–McGuire. In that paper, a state is defined as a $\overline{w}$–dimensional vector, specifying the number of active firms at each of the $\overline{w}$ efficiency levels. As we will see in Section 3, if $N < \overline{w}$, then the definition used here will result in fewer states than the definition used in Pakes–McGuire. It is for this reason (to minimize the number of states) that we use an alternate definition in this paper.

IMPACT OF INVESTMENT. At any time period, each firm finds itself at a particular efficiency level. The firm's efficiency level next period is generated by a Markov process which depends on the firm's efficiency level in the current period, the firm's investment level in the current period, and exogenous factors. Let $k \in W$ be the efficiency level that a firm is at in the current period and $k' \in W$ be the efficiency level next period, and let the current period investment be x. Then, the program, at present, uses the following controlled Markov process for the evolution of k':

$$(1) \qquad k' = k + \tau^*,$$

where

$$\tau^* = \tau - v,$$

and

$$p(\tau) = \begin{cases} \dfrac{ax}{1 + ax}, & \text{if } \tau = 1 \\ \dfrac{1}{1 + ax}, & \text{if } \tau = 0 \end{cases}, \quad \text{and}$$

$$p(v) = \begin{cases} \delta, & \text{if } v = 1 \\ 1 - \delta, & \text{if } v = 0. \end{cases}$$

Conceptually, τ represents the output of the firm's own investment process,

while v represents factors which erode the profits of all firms in the industry. Depending on the profit functions this evolution can be a result of improvements in the products marketed outside the industry, or increases in the price of labor or other variable factors of production (see below). Note that the value of v is the *same* for all firms active at that time. This generates positive correlation among the profits of the firms within the industry and follows quite naturally from more primitive specifications of profit functions (see the discussion in Ericson–Pakes, and in Pakes–McGuire). Note also that although this specification allows for only limited movement in k over a single decision period, we have leeway in determining how many decision periods there should be in any real time period (say one year). By allowing for many decision periods per year we can get much richer distributions for the increments in k in any year (if we do this we have to adjust the discount rate to reflect the fact that a decision period is less than a year).

Given the simple nature of our family of distributions, we can explicitly write out the probabilities of changes in k. For instance, if we do not condition on the outside world,

$$\text{Prob. (k rises by 1} \mid x) = \frac{(1 - \delta)\ ax}{1 + ax}.$$

Similarly, the other two alternatives, that k remains the same and that k falls by 1 all have easy to compute values in terms of the constant parameters and x. Upon inspection, we can see that the probability a firm has of having its efficiency level rise is a monotonically increasing concave function of the investment level, while the probability of its efficiency falling is a monotonically decreasing convex function of x. This insures that there will be a unique solution to the first order conditions which will be used to determine optimal investment, and hence simplifies computation. Also, as we show below, this particular functional form makes it easy to solve analytically for the unique solution to the first order conditions for investment. However the user can replace this family of distribution

functions for τ^* given x with any other family with finite support and which is stochastically increasing in x (in the first order dominance sense) and the algorithm should still work. (For a more detailed discussion of the possibilities see Ericson–Pakes.)

FINDING BOUNDS FOR W. Theory tells us that there always exists an efficiency level sufficiently high that firms do not want to invest at this efficiency level (no matter what the industry structure is). Since our specifications insure that efficiency cannot improve without some investment, and that the increment in efficiency levels that results from investment in any given period is bounded with probability one, it follows that the efficiency level of a firm will never be higher than some value, which we have called $\overline{w}$. Typically, we compute $\overline{w}$ as the maximum efficiency level that a monopolist would ever reach: i.e. we start out with a very large $\overline{w}$ and compute the monopolist problem (this is an easy computation, see below), and then use the w at which the monopolist stops investment as $\overline{w}$ in the subsequent calculations. In the examples we have tried this has always been at least as high as the maximum efficiency level ever achieved by a firm when there can be more than one firm active. However, we have no proof that this must be true, so one must check whether there is zero investment when $w=\overline{w}$ at the end of the algorithm.

In determining a lowerbound to efficiency levels, theory is more helpful. First of all, theory insures that if a firm is at a sufficiently low efficiency level, which we call 0, it will always want to choose the exit option (which we will discuss shortly). Moreover, theory also tells us that the w at which a monopolist exits is low enough to induce exit no matter how many firms are active. So we choose efficiency level 0 as that efficiency level that would induce a monopolist to exit. Note that it may be feasible to shorten the length of W when we allow for a large number of active firms, and this would reduce the computational burden of the algorithm.

INVESTMENT COSTS. Firms attempt to maximize the EDV of all future net cash flows

(profits minus investment costs) using a discount factor β. Investment costs are cx, where x is the amount of investment.

EXIT. Firms cease production if the EDV of future net cash flow given optimal future behavior is less then the scrap value of the firm. The scrap value is ϕ, which is received immediately, and a firm which takes it *does not* earn current period profits.

ENTRY. Potential entrants calculate the EDV of entering and compare it to a one time sunk entry cost; entering if the EDV exceeds the costs. If they do enter they receive *no* profits in an initial startup year, and then enter at either W_E or W_E − 1 in the following period, depending on what happened to the outside alternative in the interim (they enter at W_E − 1 if v=1). In general, the sunk cost of entry, X_E, is a random variable, which, for simplicity, we assume is distributed uniformly between X_EL to X_EH (one can replace this with any other distribution if one wishes). The incumbent firms know only the distribution of X_E at the time of their decision process (thus the realization of X_E is not part of their state). However, a potential entrant knows its draw of X_E at the time when it decides whether or not to enter. Note that when X_EL = X_EH, entry is deterministically specified by the EDV of future net cash flows. In general, however, at each incumbent industry structure, there is a probability of there being an entrant equal to the probability that the random entry cost is less than the EDV of a new entrant. Random entry costs tend to smooth out the relationship between the EDV of future net cash flows of incumbents and the value of entry, and this tends to eliminate convergence problems in our iterative algorithm (in the deterministic case there is a discontinuity in the value function of incumbents to changes in the value function of the entrant when that value function passes a critical level, and this can generate cycles in the algorithm used to compute equilibria, see Pakes–McGuire).

UNIQUENESS. The MPN equilibrium strategies at any stage may not be unique. In particular in some equilibria a higher efficiency incumbent may exit while a lower efficiency incumbent remains active. As we do not want to find this kind of equilibria, we force the algorithm to converge only to the type of equilibrium where the lower efficiency incumbent *always* exits when the higher efficiency incumbent is exiting. This ensures that all equilibria that result from our algorithm will be of the latter type (it does not insure that there is only one such equilibrium).

CONSTANTS IN THE ALGORITHM. We have implicitly defined several constants. Here we list their function and give typical values for the examples we have computed:

Table 1:

Const	Use	Typical Value
a	Const. used in state increase / investment fn.	3
c	Cost in dollars of a dollars worth of investment	1 (unless tax credi
N	Number of firms (maximum ever allowed active)	≤6
$\overline{w}$	Highest efficiency level attainable	19
W_E	Efficiency level at which new firms enter	4
X_E	Sunk cost of entry (deterministic entry)	0.2
X_EL	Lowest sunk entry cost (w/ random entry costs)	0.15
X_EH	Highest sunk entry cost (w/ random entry costs)	0.25
β	Discount factor	0.925
δ	Probability of outside alternative rising	0.7
ϕ	Scrap value at exit	0.1

• **Section 2: The Algorithm**

MATRICES CARRIED IN ALGORITHM. Profits, Π, is an invariant matrix (it does not change from iteration to iteration), inputted into the algorithm, which enumerates the profits of the underlying one–shot game that we are using, for each possible industry structure. Each iteration of the algorithm starts with a matrix X which contains the

investments, and a matrix V, which contains the value functions, outputted from the last iteration. X, V, and Π are defined for every state [w,n]. For instance, if N=5, one element of V is V[(12,9,9,2,1),4], which provides the value of a firm with efficiency level 2 (the fourth one) at this industry structure. Note that though the first index here is a vector, we use the binomial encoding scheme we mentioned above in order to code descending 5–tuples (in this case) into unique integers; so V will have two integer indices, the first of which maps into vectors.

ITERATIVE PROCEDURE. The algorithm iterates on the V and X matrices, until the maximum of the element–by–element difference between successive iterations in these matrices is below a tolerance level. The calculations in each iteration are performed separately for each row (industry structure) using only the old values of the matrices V and X. Thus, each row of the updated matrices at any iteration is a function only of the previous iteration's matrices. We note that if each element of V and X has converged, then we are assured of having computed a MPN equilibrium to the dynamic game. This can be checked by showing that the converged policies and value functions satisfy the list of conditions that define an equilibrium in Ericson–Pakes.

We now describe the process that provides us with new V and X matrices at every iteration. The computation is done separately for each element of V and X. Thus we describe what the algorithm does to V[w,n] and X[w,n] for every [w,n] ε (W^*,N). Keep in mind that, although we illustrate the transformation process for the typical element, this process is done to all possible states [w,n] ε (W^*,N). The purpose of the iteration is to calculate the optimal X[w,n] and V[w,n], given that all other rows in V and X are as specified in last iteration's output. Within an industry structure, we start with the highest efficiency level firm, update its choices using last iteration's values, then replace its investment and entry/exit choices (but not its value function) with the current iteration choices for use in calculating policies for the second–highest efficiency firm. The

second–highest firm's investment and entry/exit choices (along with the highest) are replaced with current iteration choices for the third–highest firm's calculations etc.

For a firm to make its investment decision (and hence for us to update x[w,n]) that firm must have a perception of the distribution of its competitors' efficiency levels in the next period. It forms this perception by using the V matrix to determine which of its incumbent competitors will remain active and whether a new entrant will appear. It then uses the X matrix, the locations of the incumbents that continue, and the exogenous process determining the location a new entrant will enter at (if there is a new entrant) in order to form its perception of where its competitors will be in the next period. Recall that for all states outside its industry structure, the firm looks at last iteration's values, while for states in its industry structure, the choice is different, as described above.

UPDATING NUMBER OF INCUMBENTS. A firm determines whether an incumbent competitor exits by looking at the value function of its competitor. This is done by looking at the value of different firms within the given industry structure, and then determining if the value function at a competitor's state equals ϕ. If it does, then the given firm will perceive that the competitor will go out of business at the start of this period (it can obtain ϕ simply by shutting down production).

In order to ensure that we are not computing an equilibrium where a firm with some efficiency level remains active and a firm with lower efficiency level exits, we further define the strategy set so that if a firm perceives that a competitor with higher efficiency level than its own has exited, then the firm must also exit. (As above, in our version of the program, this computation is a function of current iteration's values.)

To illustrate how we determine which firms are perceived to be active, we give an example. Consider the state [w,n] = [(9,8,7,2,1),3]. Then, to find out which of its incumbent competitors it perceives to be active, we begin by examining whether V[(9,8,7,2,1),1] is greater than ϕ. If the firm perceives that V[(9,8,7,2,1),1] = ϕ, then the

firm is forced to exit (as there is a competitor with efficiency level 9 who is exiting, while the firm only has efficiency level 7). Similarly, the firm computes the activity of all of its other competitors. If either of the last two competitors are inactive, this does not force the firm to liquidate.

As indicated earlier, this computational convention resolves one possible source of nonuniqueness in the results; when we can establish an equilibrium if either one of two firms exit we will always select the equilibrium in which the firm with the lower value of the efficiency parameter will exit. Of course there may also be an equilibrium in which both firms make the same exit decision.

For any [w,n], we define w' as that industry structure that results after exit has been accounted for, and m as the number of active firms (firms with positive efficiency levels) in w'. We will use this definition below.

UPDATING ENTRY. After iterating on which of a firm's incumbent competitors remain active, we iterate on whether there will be entry. Recall that the algorithm only allows for entry if $m < N$, i.e. if the number of active firms is less than the preset upper bound. (Theory tells us that if we keep pushing N up we will eventually reach a situation where there never will be entry when $m=N-1$).

Entry is determined in the following manner. For any industry structure, we examine how many active firms there are, by counting those firms with efficiency level greater than zero. If the number of active firms is less than N, we compute the EDV of entering, and then calculate the probability of it being greater than the random entry fee X_ E. (We will discuss the calculation of EDVs of entering later on in this section.)

The incumbent making the investment decision proceeds as follows: after accounting for exit by other incumbents as described in the last subsection, the incumbent observes whether or not there is space for entry, by looking at whether the number of incumbents is less than N. If there is space for entry, the incumbent perceives that a new

competitor will enter with probability given by the probability that the EDV of entry is greater than X_E, and that there is no entry with one minus this probability. Letting [w′,n] be the state vector of incumbents after the exit decisions of competitors are made as before, we will denote the probability of entry by $\lambda(w',n)$. Recall that only one new entrant is allowed at any state in any iteration.

If there is entry, the entrant does not earn any profits and cannot invest in the current period and becomes like any other firm at the beginning of the next period with efficiency of either W_E or W_E − 1, depending on whether v in the interim is 1.

To illustrate how we find the firm's perceptions of its future competitors, we revive the example used previously (of w = (9,8,7,2,1)), and assume that the firms with values 1 and 2 will be inactive, using last iteration's values. Then, to determine entry, the first four firms (assuming that their predecessors are all active) look at the EDV of entering for [(9,8,7,W_E,0),4]. (The fourth firm should look at the EDV of entering for [(9,8,7,2,W_E),5], but this will be the same as the above as the firm at 2 exits last iteration.) The last firm always assesses that there will be no entry, as if it is remaining active, all its competitors are also active. If the probability of X_E being less than this EDV is $\lambda(w',n)$, then for purposes of determining investment and calculating value functions, the (first three) incumbent firms effectively see their industry structures as (9,8,7,W_E,0) with probability $\lambda(k,w')$, and (9,8,7,0,0) with probability $1-\lambda(w',n)$. Note that regardless of whether there is an entrant or not the firm evaluates that there is no investment from the fourth slot, i.e. that X[(9,8,7,W_E,0),4] = 0 (regardless of what the value of x that was calculated in the last iteration for the "fourth" w). Similarly, in both cases of entry and no entry, the firm sees its competitors investing what last iteration's values indicate that they would invest at states [(9,8,7,2,1),·], as in equilibrium this will yield the correct value. Finally, we note that the first three firms evaluate their current period static profit functions using Π[(9,8,7,0,0),·], while the fourth firm uses Π[(9,8,7,2,0),4] and the fifth firm, Π[(9,8,7,2,1),5].

UPDATING INVESTMENT. We now assume that the firm's perception of future competitors has been formed in the manner described above and then illustrate how it chooses its optimal investment policy. Again this calculation is done separately for every $[w,n] \in (W^*,N)$.

Notationally let:

- w' be the incumbent efficiency levels *after* updating for exit (as above);
- $m(w')$ be the number of active competitors at $w=w'$ [i.e. $m(w')$ is just the number of elements in w' that are strictly positive] (as above);
- $\lambda(w',n)$ be the probability of entry as described above — we provide the details of how $\lambda(w',n)$ is calculated below;
- $e(j)$ be a vector all of whose elements are zero except for the j^{th} element which is one — the dimension of $e()$ will be obvious from the context;
- i be a vector all of whose elements are one — again its dimension will be obvious from the context;
- τ be the vector containing the random τ of competitors;
- n_e be the position of the entrant for any industry structure; $n_e = m(w')+1$ unless the permutation cycle has been reordered.
- $w'_1,\ldots,w'_N$ be the elements of the vector w';
- $x_1,\ldots,x_N$ (except for x_n) be the investments of the N−1 competitors at w' (these are obtained by permuting the last iteration's X matrix and reading off $x[(w_1,\ldots,w_N),i]$, $i = 1,\ldots,n-1,n+1,\ldots,N$);
- a hat ($\hat{}$) in a summation mean to omit that element; and
- define a permutation cycle of industry structures (vectors or points) to be all points obtained by permuting the elements of a given $[w,n]$. If a permutation cycle contains an element that is not in descending order, we reorder it.

We begin by writing down the value function at the i^{th} iteration as:

$$(2)\quad V^{i}(w,n) = \max\{\phi, \sup_{x\geq 0}[\Pi(w',n) - cx + \beta\lambda(w',n)$$
$$\{\Sigma_{\tau_1=0}^{1}\ldots\Sigma_{\tau_N=0}^{1}\Sigma_{v=0}^{1}V^{i-1}[w' + W\_E\,e(n_e) + \tau - iv, n]Pr[\tau_1|x_1^{i-1},v]\cdots Pr[\tau_n|x,v]\cdots Pr[\tau_N|x_N^{i-1},v]p(v)\}$$
$$+\beta[1-\lambda(w',n)]$$
$$\{\Sigma_{\tau_1=0}^{1}\ldots\Sigma_{\tau_N=0}^{1}\Sigma_{v=0}^{1}V^{i-1}[w' + \tau - iv, n]Pr[\tau_1|x_1^{i-1},v]\cdots Pr[\tau_n|x,v]\cdots Pr[\tau_N|x_N^{i-1},v]p(v)\}.$$

Note that each of the probabilities of the τ being one or zero is an easily calculated function of the investment for that agent, as described in equation (1).

In order to ease notation, we let:

$$(3)\quad Calcval(w',n) = \lambda(w',n)\{\Sigma_{\tau_1=0}^{1}\ldots\Sigma_{\tau_n=0}^{1}\ldots\Sigma_{\tau_N=0}^{1}\Sigma_{v=0}^{1}V^{i-1}[w' + W\_E\,e(n_e) + \tau - iv, n]$$
$$Pr[\tau_1|x_1^{i-1},v]\cdots Pr[\tau_n|x,v]\cdots Pr[\tau_N|x_N^{i-1},v]p(v)\}$$
$$+\ [1-\lambda(w',n)]\{\Sigma_{\tau_1=0}^{1}\ldots\Sigma_{\tau_n=0}^{1}\ldots\Sigma_{\tau_N=0}^{1}\Sigma_{v=0}^{1}V^{i-1}[w' + \tau - iv, n]$$
$$Pr[\tau_1|x_N^{i-1},v]\cdots Pr[\tau_n|x,v]\cdots Pr[\tau_N|x_N^{i-1},v]p(v)\}.$$

$Calcval(\cdot)$ sums out over the probability weighted average of the possible states of future competitors, but not over the investing firm's own future states. Thus, $Calcval(\cdot)$ indicates the firm's EDV for each of the two possible realizations of the own firm's investment process, τ. Substituting (3) into (2) we find that

$$(4)\quad V^{i}(w,n) = \max\{\phi, \sup_{x\geq 0}[\Pi[w',n] - cx +$$
$$\beta[ax/(1+ax)]Calcval(w'+e(n),n) + \beta[1/(1+ax)]Calcval(w',n)]\}.$$

We let $x^{i}[w',n]$ be the investment level that solves (4). To compute it we first

solve for the firm's optimal investment level given that this investment is not zero and that the firm remains active; we call this solution $x[w',n]$. We then set investment to be either this number or zero; zero is our solution for investment if either $x[w',n]$ is negative, or if, once we have calculated the optimal investment and substituted it back into (4), we find that it is optimal for the firm to exit. More formally, letting D_x denote the derivative with respect to x, the first stage investment solves:

$$x[w',n] =$$

$$\text{argsolv}_x \{c = \beta\left[D_x\left\{ax/(1+ax)\right\} \text{Calcval}(w'+e(n),n) + D_x\left\{1/(1+ax)\right\} \text{Calcval}(w',n)\right.$$

$$= \text{argsolv}_x \{c = \beta\left[D_x\left\{ax/(1+ax)\right\} \text{Calcval}(w'+e(n),n) - D_x\left\{ax/(1+ax)\right\} \text{Calcval}(w',n)\right].$$

Now note that $D_x\left\{1/(1+ax)\right\} = a/(1+ax)^2 = a[1-p(x)]^2$, where $p(x) = [ax/(1+ax)]$. So, letting $v1 = \text{Calcval}(w'+e(n),n)$ and $v2 = \text{Calcval}(w',n)$, we can write:

$$x = \text{argsolv}_x \{c = \beta[a[1-p(x)]^2 v1 - a[1-p(x)]^2 v2]\}$$

$$\Longrightarrow x = \text{argsolv}_x\{c = \beta\, a\, [1-p(x)]^2 (v1 - v2)\}$$

$$\Longrightarrow x = \text{argsolv}_x\left\{1 - p(x) = \sqrt{\frac{1}{\beta a(v1 - v2)}}\right\}$$

$$(5) \qquad \Longrightarrow p(x) = 1 - \sqrt{\frac{1}{\beta a(v1 - v2)}}$$

Taking the inverse of $p(x)$, we can see that:

$$(6) \qquad x[w',n] = \frac{p(x)}{a - ap(x)}, \text{ where } p(x) \text{ is as in (5).}$$

That is, we have derived an *analytic* formula for the optimal x that a firm chooses at any time period, given that it remains active and $x \geq 0$. In the actual algorithm we obtain x directly from this formula. If we are at a boundary case where the optimal

probability of having the state rise conditional on the outside alternative is less than 0 then we simply replace the p calculated from (6) with 0, as this is the optimal feasible p. We do, however, have to compute the function Calcval. Calcval and x are derived using last iteration's V and X matrix (except for a firm's own cohorts of higher efficiency).

It is now straightforward to derive the optimal value function; we need only plug in our optimal x into an equation for the value function, and compute:

$$V^i(w,n) = \max \{ \phi, \Pi(w',n) - cx[w',n] + \beta\{ax[w',n] / (1+ax[w',n])\} Calcval(w'+e(n),n) + \{1 / (1+ax[w',n]) Calcval[w',n]\}.$$

We now come back to the possibility that the firm will choose to exit in the current period. In order to incorporate this, we note that if the value function calculated above has the property that:

$$V^i(w,n) = \phi,$$

then the firm's optimal investment strategy given that it remains in business is no better than simply stopping production, and selling its equipment. The firm should then exit, which implies not investing at all. Thus, if $V^i(w,n) = \phi$ then we set $x=0$ with probability one. Notationally, by substituting from (6), we determine the actual investment $x^i[w',n]$ using the above logic as:

$$(7) \qquad x^i[w',n] = \{V^i(w,n) > \phi\} \, x[w',n],$$

where $\{\cdot\}$ is the indicator function which takes the value of one if the conditon inside it is satisfied, and the value of zero elsewhere. It is these investment values that fill up the i^{th} iteration investment matrix.

CALCULATING THE PROBABILITY OF ENTRY.

Again we let the efficiency levels of the incumbent competitors after exit decisions are made be w'. Thus, if the element we are examining is $[w,n] = [(9,5,4,2,1),2]$, then, if the last three firms are exiting, we would obtain $w' = (9,5,0,0,0)$. A potential entrant can then enter, if and only if $m(w') < N$. In this case, $m(w') = 2$, and $N = 5$, so there is a possibility of entry. We let x be the N–dimensional vector of investments of the incumbent competitors' of X_E (obtained in the same manner as the similar vector used above). Now the value of entering to the potential entrant is simply the value of an incumbent that; i)realized that there would be no entry and ii)was constrained not to have profits or invest in the current period. The EDV of entering is then just:

$$(8) \qquad V^e(w') = \beta\, \text{Calcval}[w' + W\_E\, e[m(w')+1], m(w')+1; \lambda = 0].$$

Recall that by the definition of Calcval, it is understood that the Nth position of x (the entrant's position) is zero, and as noted, this calculation is done setting $\lambda[w' + W\_E\, e[m(w')+1], m(w')+1] = 0$.

A firm would choose to enter if and only if $V^e > X\_E$, the random entry cost. Recall that the random cost of entry is assumed to distribute uniformly between X_EL and X_EH. Consequently the probability of entry to an incumbent with state $[w,n]$, incumbent competitors who remain active specified by w' and $m(w') < N$, is:

$$(9) \qquad \lambda(w',n) = \min\left\{ \max\left[\frac{V^e - X\_EL}{X\_EH - X\_EL}, 0\right], 1 \right\},$$

with the understanding that $V^e = V^e(w')$ as in (8), and that if X_EL = X_EH, so entry is deterministic, then $\lambda(w',n) = 1$ if $X\_EL \le V^e$, and it is zero otherwise. Note that if $X\_EL < X\_EH$, then $\lambda(w',n)$ is a continuous functional of last iteration's elements of the V and X matrices.

The same Calcval function that is used for the current firm we are examining is used to calculate the EDV for the entering firm (evaluated at a different point). In order to avoid repeating the same calculation several times in an iteration, we calculate the entry values for all $w \in W^*$ at the beginning of each iteration, and store these values in a table 'isentry' for use throughout the iteration.

UPDATING N. The algorithm starts with the one–firm problem, and solves for its value function and optimal policies. The one–firm problem is a contraction mapping and hence our iterative procedure is guaranteed to converge [using any bounded map from W to $\mathbb{R}$ as an initial condition; we usually start with $V^0 = \Pi(k)$, and $X^0 = 0$]. As noted above, this allows us to set the lower and upper bounds of W (though we have to insure that there is actually no investment at the upper bound for higher N). We then proceed to the two–firm problem, using as starting values for X and V the fixed point values that we solved for in the one–firm problem. We substitute:

$$V^0[(w_1,w_2),1] = V^\infty(w_1),\ \forall\ w_1,w_2 \in W,$$
$$V^0[(w_1,w_2),2] = V^\infty(w_2),\ \forall\ w_1,w_2 \in W,$$

where $V^\infty(k)$ is the fixed point for the one firm problem. Analogously, for the $N > 2$ firm problem, we use as starting values:

$$V^0[w,n] = \begin{cases} V^\infty[(w_1,\ldots,w_{N-1}),\ n], & \text{if } n < N \\ V^\infty[(w_1,\ldots,w_{N-2},w_N),n-1], & \text{if } n = N \end{cases}.$$

It is understood that the elements of X are updated in the same way as V, which is shown above.

This process is then repeated until $\lambda(w',n) = 0$ for all (w',n) with $m(w') \geq N-1$.

As in Ericson–Pakes we then know that, provided our initial industry structure has less than N firms active, N is an upper bound to the number of firms that will ever remain active.[2]

POSSIBILITY OF VALUE AND POLICY ITERATION. The algorithm just described never uses the current iteration's output to calculate the new X and V values for [w,n]. Two possible variants of the algorithm that do use this output come to mind. One variant uses the technique of "value" iteration (see Pakes and McGuire 1993 a and b). This variant iterates the X matrix to convergence for a *fixed* V, followed by updating the V matrix.[3] Similarly, another variant uses the technique of policy iterations. This variant iterates the V matrix to convergence for a *fixed* X matrix, followed by updating the X matrix. Though both these variants require more calculations per iteration, we would expect them to decrease the number of iterations needed before convergence. However, neither of these variants improved the performance of the algorithm in the examples that we computed, so we do not discuss them further (though it is easy to add them to the algorithm if the user wishes to do so).

• **Section 3: Encoding and Decoding Elements of W^***

Recall that an element of W^* is a weakly descending N–tuple of integers in the

[2]In many of our runs we stop the algorithm after fairly large N, run the output programme (see below) that generates the ergodic class of industry structures, and then ask if there are any points in the ergodic class with N firms active. If there are no (or a very small frequency) of points in the ergodic class with N active firms we stop the algorithm. This is a less stringent criteria then the more formal criteria for stopping the algorithm used above.

[3]Alternatively one could use any other nonlinear equation solver that solves simultaneously for all incumbents investments given last iterations value function. See Judd (forthcoming) for a discussion of the alternatives.

range 0,...,$\overline{w}$. We are interested in coding elements of W^* into integers, in order that we can express V, X, and Π by two–dimensional matrices. This will also allow us both to avoid computing the same state two or more times, and to determine the cardinality of the state space.

Formally what we require is an encoding function that bijectively maps onto a subset of $\mathbb{Z}$ (the integers) that is of the same cardinality (has the same number of elements in it) as does W^*, and a decoding function that maps from this subset of integers back into elements of W^* (by bijective we mean that the map is one to one and onto; so that each element of the subset of $\mathbb{Z}$ can be mapped into one, and only one, element of W^*). We accomplish this by first expressing the cardinality of W^* in terms of binomial coefficients, and then by showing how we can use the same binomial coefficients to define a coding function.

We proceed by describing the encoding function we use. After this description we provide simple proofs which insure that the encoding function we provide has the properties that we claim it has. Those who are not interested in the details of the proofs can skip them and go directly on to the next section of the paper.

A game with fixed N, and a known $\overline{w}$, has a set of possible industry structures (different w's) which can be determined as a function of N and $\overline{w}$. Let this set of possible state vectors be $W^*(N, \overline{w})$. Throughout we will only consider cases where $N \leq \overline{w}$[4].

The following definitions will prove useful.

DEFINITIONS.

(1) We let '# [A]' denote the cardinality of (the number of distinct elements in) any set A.

---

[4]If $m > \overline{w}$, then we would minimize the dimension of the state space by using, as states, the number of firms at each possible $k \in W$; see Pakes–McGuire.

(2) We define a matrix Binom which contains as its elements the binomial coefficients (in the standard order, see below) in the last $\overline{w}$ columns, and a column of zeros in the first column,

$$(10) \qquad \text{Binom} = \begin{bmatrix} 0 & 1 & & & & & & & \\ 0 & 1 & 1 & & & & & & \\ 0 & 1 & 2 & 1 & & & & & \\ 0 & 1 & 3 & 3 & 1 & & & & \\ 0 & 1 & 4 & 6 & 4 & 1 & & & \\ 0 & 1 & 5 & 10 & 10 & 5 & 1 & & \\ 0 & 1 & 6 & 15 & 20 & 15 & 6 & 1 & \\ 0 & 1 & 7 & 21 & 35 & 35 & 21 & 7 & 1 \\ \cdot & \cdot & \cdot & & & & & & \end{bmatrix}.$$

Note that each element of Binom (except for the first row) is obtained by adding the values of the element above it, and the element above and to the left of it. Thus,

$$(11) \qquad \text{Binom}[a,k] = \text{Binom}[a-1,k] + \text{Binom}[a-1,k-1].$$

Also the elements of Binom (except the first column) can be obtained as permutation coefficients from the well-known formula:

$$\text{Binom}[a, k] = {}^{a}P_{k-1} = \frac{a!}{(k-1)!\ (a-k+1)!},$$

where P represents the number of permutations. Note that this implies that the elements of Binom are increasing as we move down any superdiagonal; that is Binom[a,k] ≤ Binom[a+1,k+1], ... etc.

THE CODING METHODS. We now provide # $[W^{*}(N,\overline{w})]$ in terms of a binomial

coefficient (again the proof of this claim is given below). This is a first step to coding elements of W^*, as it allows us to see how many states there are for any problem.

Claim 1: $\#\ [W^*(N,\overline{w})] = \mathrm{Binom}[N+\overline{w}+1,\overline{w}+2]$.

Next we define a coding function that maps $W^*(N,\overline{w})$ into (a connected) subset of $\mathbb{Z}$ of cardinality $\#\ [W^*(N,\overline{w})] = \mathrm{Binom}[N+\overline{w}+1,\overline{w}+2]$. Below we will show that this coding function is bijective. The connected subset will be all non–negative integers with values $< \#\ [W^*(N,\overline{w})]$, so that there will be precisely one element of the subset for each element $W^*(N,\overline{w})$. The coding function, which we denote by $\mathrm{enc}_{N,\overline{w}}(w)$: $W^* \longrightarrow Z$, is given by

$$(12) \qquad \mathrm{enc}_{N,\overline{w}}(w) = \sum_{i=1}^{N} \mathrm{Binom}[w_i+N-i+1,w_i+1],$$

where we are writing $w \in W^*$ in component form as $w = (w_1,\ldots,w_N)$. Note that we may think of enc graphically, as being obtained by adding an element of the principal diagonal of Binom for $i=N$, with an element of the second diagonal down for $i=N-1$, and with an element from the third diagonal for $i=N-2$ and so on. The element chosen from each diagonal is the w_i+1^{th} element, counting from the top of the matrix.

The encoding function that we have just defined is used whenever we are referring to any element of the matrices V, X, and Π, all of which have dimensionality $\#\ [W^*(N,\overline{w})]$ by N. The decoding function is the inverse of the encoding function (which exists by virtue of the fact that the encoding function is a bijection). To decode $z \in Z$ into $w \in W^*$, the program starts with the Nth horizontal diagonal, and finds the largest element that is no greater than z; we let w_1 be this element number. The program then takes the remainder, and repeat the process with the N–1th horizontal diagonal; this provides a w_2.

Repeating this process N times yields w, which satisfies $w = enc^{-1}(z)$.

In order to lower computation time, the computer performs this decoding process only *once* for each N (for every $z \in \mathbb{Z}$ s.t. $0 \leq z < \#\ [W^*(m,\overline{w})]$), and then uses a table lookup to reduce computation time. The decoded values are stored in a matrix Dtable, of dimensionality N by $\#\ [W(N,\overline{w})]$. To find the stored decoding of $z \in Z$, noting that $enc^{-1}(z) = Dtable[\cdot,z]$, we simply look at the zth column of the matrix Dtable. There are other, less intuitive schemes, to store the encoded values in a table, though they tend use more memory than the basic method from (12).

THEOREMS AND PROOFS. We now come back to the proofs of the claims that we used in building our coding function. Again we have

Claim 1: $\#\ [W^*(N,\overline{w})] = Binom[N+\overline{w}+1,\overline{w}+2]$.

Proof of Claim 1: We use induction to prove this theorem. We start with a vector of initial conditions; in particular we show directly that the theorem is true for the second column of Binom and for its dominant diagonal elements. We will then use an inductive step which will prove that given this vector of initial conditions, the theorem holds true for every element below the dominant diagonal. (Note that by our claim, any value of Binom that is referenced will be a superdiagonal or diagonal element.)

For the initial condition, suppose that $\overline{w} = 0$. Then, no matter what N is, $W^*(N,0)$ has only one element, namely, $\{(0,\ldots,0)\}$. Thus, no matter N, provided $\overline{w}=0$,

$$\#\ [W^*(N,0)] = 1 = Binom\ [N+1,2] = Binom[N+\overline{w}+1,\overline{w}+2],$$

Now suppose that $N = 1$. Then, no matter what $\overline{w}$ is, $W^*(1,\overline{w})$ has $\overline{w}+1$ elements, namely $\{(0),(1),\ldots,(\overline{w})\}$. Thus provided $N=1$, we have, no matter $\overline{w}$,

$$\#\ [W^*(1,\overline{w})] = \overline{w}+1 = \text{Binom}[\overline{w}+2,\overline{w}+2] = \text{Binom}[N+\overline{w}+1,\overline{w}+2].$$

That gives us the vector of initial conditions for the inductive argument. To complete the proof of the theorem it will suffice to show that if the theorem holds for the element directly above the one we are examining, and for the one directly above and to the left, it holds also for the element we are examining (a quick look at the matrix will suffice to convince the reader that this is the required inductive step for the theorem). For this to be true all we need is to show that $\#\ [W^*(N,\overline{w})]$ follows a recursion analogous to that given for binomial coefficeints above, i.e. that:

$$\#\ [W^*(N,\overline{w})] = \#\ [W^*(N,\overline{w}-1)] + \#\ [W^*(N-1,\overline{w})].$$

To see this, we note that an element in $W^*(N,\overline{w})$ can either have its first component be $\overline{w}$ or not. The number of elements which do not have a first element of $\overline{w}$ is $\#\ [W^*(N,\overline{w}-1)]$, by definition. The number of elements which do have a first element of $\overline{w}$ is simply the number of weakly descending N−1 dimensional tuples, where each element ranges between $\overline{w}$ and 0; or $\#\ W^*(N-1,\overline{w})$. That completes the inductive argument, and hence the proof of the theorem. •

We now show that $\text{enc}_{N,\overline{w}}$ provides a bijective map from $W^*(N,\overline{w})$ to $[0,\ldots,\#\ [W^*(N,\overline{w})-1]$. We start with definitions and a lemma.

Definitions:

(1) Let $D(N,\overline{w}) = \{z \in \mathbb{Z} \mid z \geq 0,\ z \leq \#\ [W^*(N,\overline{w})]-1\}$.

(2) We say that our function $\text{enc}_{N,\overline{w}}$ maps surjectively from W^* to $D(N,\overline{w})$, if for every $z \in D(N,\overline{w})$, $\exists\ w \in W^*$ such that $\text{enc}_{N,\overline{w}}(w) = z$ (that is the map is "onto").

Lemma: If $enc_{N,\overline{w}}$ maps surjectively from W^* to $D(N,\overline{w})$, then, $enc_{N,\overline{w}}(\cdot)$ is bijective with domain $W^*(N, \overline{w})$ and range $D(N,\overline{w})$.

Proof of Lemma: This follows directly from the fact that the cardinality of $W^*(N,\overline{w})$ equals the cardinality of $D(N,\overline{w})$ and is finite. (If the claim were not true then there exists a $z \in D(N,\overline{w})$ that maps into two points in $W^*(N,\overline{w})$; which, given that the map is surjective, implies that $\#\ [W^*(N,\overline{w})] - 1 \leq \#\ [D(N,\overline{w})]$, a contradiction). •

It follows that to show that our function is bijective we need only show that it is surjective onto $D(N,\overline{w})$.

Claim 2: $enc_{N,\overline{w}}$ is a bijective function, with range $D(N,\overline{w})$.

Proof of Claim 2: We prove the theorem by induction on N for all finite values of $\overline{w}$.

For the initial condition of the inductive argument assume $N = 1$. Then, for any $\overline{w}$, $enc_{1,\overline{w}}(w) = Binom[w+1,w+1] = w$. Thus, in this case the encoding is simply the identity map, which is clearly a bijection from $W^*(1,\overline{w}) = \{0,\ldots,\overline{w}\}$, onto itself.

Our inductive assumption is that for any m and for all $\overline{w}$, $enc_{m-1,\overline{w}}$ is a bijection with range $D(m-1,\overline{w})$. For the inductive step what we need to show is that this inductive assumption implies that for any $z \in D(m,\overline{w})$, $\exists\ w \in W^*(m,\overline{w})$ such that $enc_{m,\overline{w}}(w) = z$ (that is, that $enc_{m,\overline{w}}$ is surjective). To do this we will take an arbitrary z and construct a $w^* \in W^*(m,\overline{w})$ such that $enc_{m,\overline{w}}(w^*) = z$.

We start by finding the largest element of the mth horizontal diagonal of Binom such that this element is no greater than z. Formally, let:

$$w_1 = \text{the largest } x \in \mathbb{I}, \text{ such that } Binom[x+m,x+1] \leq z.$$

Note that the encoding function implies that when we encode w, w_1 will add Binom[x+m,x+1] to the encoded value. Now, let

$$y = z - \text{Binom}[w_1+m,w_1+1].$$

Subclaim: $y \in D(m-1,w_1)$.

Proof of subclaim: Clearly, $y \geq 0$. So all we have to show is that $y < \#\ [W^*(m-1,w_1)]$. But,

$$y < \text{Binom}[m+w_1+1,w_1+2] - \text{Binom}[m+w_1,w_1+1],$$

(by the fact Binom increases as we move down any diagonal),

$$= \text{Binom}[m+w_1,w_1+2],$$

(by definition of the binomial coefficients),

$$= \#\ [W^*(m-1,w_1)],$$

(by Claim 1).

End of Subclaim.

Using this claim and our inductive step then, $\exists\ v \in W^*(m-1,w_1)$ such that $\text{enc}_{m-1,w_1}(v) = y$. Let $w_2 = v_1$, $w_3 = v_2$, ..., $w_m = v_{m-1}$. Note that $w \in W^*$, as $w_2 \geq \ldots \geq w_m$, as this holds from v, and $w_1 \geq w_2$, as $v \in W^*(m-1,w_1)$.

Taking this value of w as w* we have $\text{enc}_{m,\overline{w}}(w^*) = z$. To see this note that the coding of the last m−1 digits are the same in both $W^*(m,\overline{w})$, and $W^*(m-1,\overline{w})$:

$$\text{enc}_{m,\overline{w}}(w) = \sum_{i=1}^{m} \text{Binom}[w_i+m-i+1,w_i+1]$$

$$= \text{Binom}[w_1+m, w_1+1] + \sum_{i=2}^{m} \text{Binom}[w_i+m-i+1, w_i+1]$$

$$= \text{Binom}[w_1+m, w_1+1] + \sum_{j=1}^{m-1} \text{Binom}[w_{j+1}+m-(j+1)+1, w_{j+1}+1]$$

$$= \text{Binom}[w_1+m, w_1+1] + \sum_{j=1}^{m-1} \text{Binom}[v_j+(m-1)-j+1, v_j+1]$$

$$= \text{Binom}[w_1+m, w_1+1] + \text{enc}_{m-1, w_1}(v)$$

$$= \text{Binom}[w_1+m, w_1+1] + y = z.$$

Thus, we have shown that $\text{enc}_{m,\overline{w}}$ maps surjectively from W^* to $D(m,\overline{w})$, for all m. This then completes the proof that $\text{enc}_{N,\overline{w}}$ is a bijection with range $D(N,\overline{w})$ for all finite $(N,\overline{w})$ couples. •

• Section 4: Some Profit Functions

We noted in Section 2 that one can input any one–shot profit function into the program that satisfies our regularity conditions. This section shows how to construct and compute two one–shot profit functions that have appeared repeatedly in the theoretical literature. It is easy to (and we have in the past) construct alternative profit functions and substitute them into the algorithm instead. The one–shot profit functions that we provide here are:

i) a homogeneous products Nash in quantities (Cournot) market where efficiency differences among firms are interpreted as differences in marginal costs; and

ii) a differentiated products Nash in prices (Bertrand) market where efficiency differences are interpreted as differences in the mean of the distribution of utilities assigned to the product by different consumers.

We briefly discuss the features of the variants of these algorithms that we used in order to generate the one–shot profits necessary for input into the algorithm.

DIFFERENTIATED PRODUCTS, NASH IN PRICES PROFITS MODEL. For a more detailed discussion of differentiated product models like the one we use here see Anderson De Palma and Thisse (1991), Berry Levinsohn and Pakes (1992), and the literature cited in these articles.

In this model, good "0" is the outside good, and goods 1,...,N are the goods produced by the firms competing in the industry. Each consumer purchases at most one good from the industry. The utility consumer "i" derives from purchasing and consuming good "n" is given by

$$U(i,n) = v_n - p^*_n + \varepsilon(i,n),$$

where v_n is the quality or efficiency index, and p^*_n is the price, of the good, and i= 1,...,M. Consumer "i" chooses good "n" if and only if she prefers it over all the alternatives, that is, if for all j=0,1,...,N,

$$\begin{aligned} \varepsilon(i,n) - \varepsilon(i,j) &\geq [v_j - v_n] + [p^*_n - p^*_j] \\ &= [v_j - v_0] - [v_n - v_0] + [p^*_n - p^*_0] - [p^*_j - p^*_0] \\ &= g[w_j] - g[w_n] + p_n - p_j, \end{aligned}$$

where $g(\cdot)$ is increasing and concave, efficiency level $w_n = g^{-1}[v_n - v_0]$, and $p_n = p^*_n - p^*_0$. Thus, g is the function that maps a firm efficiency level to a product quality level. All we need for the regularity conditions is for $g(\cdot)$ to have an upper bound, as this will generate

the needed upper bound to the profit function.[5] The function g that we provide makes use of a constant w^*, and then sets:

$$(13) \qquad \exp[g(w_n)] = \begin{cases} \exp(w_n), & \text{if } w_n \leq w^* \\ \exp(w_n)[2-\exp(w^*-w_n)], & \text{otherwise.} \end{cases}$$

Note that this implies that consumer choices are determined entirely by the quality and prices of the goods marketed in this industry relative to the quality and price of the outside alternative (only relative qualities and prices count, and we have chosen the outside alternative as our "numéraire"). Therefore when we refer to an increase in the quality of the product marketed by the firm we will mean an increase relative to the outside alternative, and we will say quality decreases only when the improvements to the firm's own product are not as great as the improvements in the outside alternative. Note also that movements in ν (the outside alternative change) will cause synchronized movements in the relative efficiencies (in the w_n's) of all firms in the industry, which in turn will generate a *positive* correlation in their profits. Since movements in the ν_j will tend to generate negative correlations in the profits of the firm's in an industry, dealing explicitly with the outside alternative allows us to rationalize the positive correlations in the profits of firms within an industry that we often observe in the data.

Let the set $C[w,n;p]$, where w is a set of w_n's (efficiency levels) and p is the price vector, be the set of ε's that satisfy the set of inequalities above, and hence which induce the choice of good n. Assume that the distribution of ε is multivariate extreme value (logit), and denote that distribution function by $G(\cdot)$. Then, the probability that a

[5] g generates decreasing marginal utility to increments in the relative quality of the goods marketed in this industry, and this, in turn, generates an upper bound to profits. A more detailed model of consumers would explicitly incorporate a distribution of income constraints and decreasing marginal utility of income (see, for example, Berry, Levinsohn and Pakes (1992)), and this would have similar effects on demand patterns (though it would also be computationally more burdensome).

random consumer will choose good "n" is

$$(14) \qquad \sigma[w,n;p] = \int_{\varepsilon \in C[w,n;p]} dG(\varepsilon) = \exp[g(w_n) - p_n] / \{1 + \sum_j \exp[g(w_j) - p_j]\}.$$

If there are N firms in the market, no fixed costs of production and constant marginal costs equal to mc, then it can be shown that if firms choose prices to maximize profits a unique Nash equilibrium exists (see Caplin and Nalebuff (1991)) and satisfies the vector of first order conditions

$$(15) \qquad -[p_n - mc]\sigma_n[1 - \sigma_n] + \sigma_n = 0$$

for n = 1,...,N. Profits are given by

$$(16) \qquad \pi[w,n] = \{p[w,n] - mc\}M\sigma[w,n],$$

where it is understood that the price and share vectors are calculated from the spot market equilibrium conditions, and M is the number (or measure) of consumers.

We are also interested in analyzing how a single–agent actor in this industry would set prices, and what market shares and profits would accrue from this choice. This will allow us to discuss the perfect colluder and social planner's solutions, which are discussed in Section 5.

Let us first consider a social planner. A social planner seeks to maximize the sum of producer and consumer surplus; hence she would set p = mc, for all firms. As demand remains unchanged, we could use (14) to determine the demand shares that would result from this choice of price. Note that total firm profits are zero for the social planner problem.

Let us now consider a perfect cartel. A perfect cartel sets prices so as to maximize joint profits. The cartel calculates demand for each product exactly as in (14). The cartel would then look at joint profits, and find the vector of prices that maximizes them. Profits, for any price vector, are given by

$$(17) \qquad \pi[w] = \Sigma_{n=1}^{N} \{p[w,n]-mc\}M\sigma[w,n].$$

Hence the price vector that maximizes profits must satisfy the vector of first order conditions

$$(18) \qquad -[p_n-mc]\sigma_n[1-\sigma_n] + \sigma_n + \sum_{j\neq n} \sigma_n\sigma_j(p_j-mc) = 0,$$

for n = 1,...,N. This characterizes both single agent problems in which we are interested.

In order to provide descriptive statistics, which we will discuss in Section 6, we will be interested in calculating the consumer surplus results from any choice prices. As discussed in Pakes–McGuire, consumer surplus for this model has an analytic form of

$$\ln \{ \sum_{n} \exp(g(w_n) - p_n) \},$$

where w is again a vector of efficiency levels, so $g(w_n)$ is a quality level. Note that this formula will apply regardless of the structure of the industry.

Table 2:

Const.	Use	Typical Value
c	Marginal cost to production for firm	5
M	Market size; meausure of consumers in market	5
n	Number of firms	≤6

w^*	Constant used to determine quality of goods[6]	12

We provide a program which calculates the Nash prices and then profits, consumer surplus and other statistics for every industry structure (for alternative sets of parameter values), and other programs that calculate the analogous variables for the single–agent problems. These solutions are obtained using a non–linear search method on (15) for the Nash case, a non–linear search on (18) for the perfect cartel case, and through simple analytic substitution for the social planner case. The parameters we set in the calculation are listed in Table 2. Note that when we calculate profits we must calculate the profits of each industry structure separately. Thus, we perform the static equilibrium computation $W^*(N,\overline{w})$ times, and fill in a row of the matrix Π with each such computation.

Note that the measure of consumers is in the same units as marginal cost. That is if M were thousands of consumers, then c is the marginal cost of producing an additional thousand units.

HOMOGENEOUS PRODUCTS, NASH IN QUANTITIES PROFITS MODEL. The homogeneous product Cournot models has producers with different, but constant, marginal costs. Marginal costs, given by $\theta(w_n)$, are determined by the multiple of a firm specific efficiency index and a common factor price index. That is if $s\tau$ and sv are the logarithms of the factor price index and of the firm's efficiency index respectively, then $w_n \equiv s\tau - sv$ and $\theta(w_n) = \gamma \exp(-w_n)$. Firms' R&D investments are directed at improving their efficiency of production (increasing their $s\tau$). Factor prices (sv) are a nondecreasing stochastic process generating a correlated negative drift in the state of the firms in the industry.

---

[6]In computing the static profit functions, we have not let $w \in \{0,\ldots,\overline{w}\}$. We found that these parameters worked properly for another set of w, so we used these, and then translated into the above scale. The set of w's be have used is $w \in \{-7,-4,-1,\ldots,51\}$. We then translated these for use in the equilibrium generation program, by mapping –7 to 0, –4 to 1, –1 to 0, 2 to 3, etc.

The spot market equilibrium in this market is assumed to be Nash in quantities. Consequently market shares and profits (gross of fixed costs) are inversely related to marginal cost, and will be increasing in efficiency k. More formally, letting q_n be firm n's output, $Q = \sum q_n$, and f be the fixed cost of production, the profits of our classic Cournot oligopolists are given by

$$\pi_n = p(Q)q_n - \theta(w_n)q_n - f$$

where $$p(Q) = D - Q.$$

It is straightforward to show that the unique Nash equilibrium for this problem gives quantities and price as

$$q^*_{w_n} = \max\{0, p^* - \theta_i\}, \quad \text{and} \quad p^* = \frac{1}{n^*+1}\left[D + \sum_{j=1}^{n^*}\theta(w_j)\right]$$

where n* is the number of firms with $q^* > 0$. Current profits can therefore be written as

$$\pi(w,n) = \max\{-f, [p^*(w,n) - \theta(w_n)]^2 - f\},$$

where $p^*(w,n) = \frac{1}{n^*+1}\left[D + \sum_{j=1}^{N}\theta(w_j)\right]$.

As in the Bertrand case, we are interested in the single agent solutions for the social planner and perfect cartel. The social planner will again set p = mc for that firm with the lowest marginal cost (highest efficiency level), and produce until supply equals demand. The perfect cartel solution is similar: it will choose to produce only at the the lowest marginal cost firm. However, it will then act like a simple monopolist, and set mr = mc. To find the consumer surplus from any outcome, we need only take the area

under the demand curve; this can be found easily, as the demand curve is linear.

In order to use this model for a one–shot game, we proceed as follows. As noted above, the marginal cost for each firm is specified by $\theta(w_n) = \gamma \exp(-w_n)$, where w_n is its efficiency level; efficiency is exactly marginal cost. Similarly, we set the probabilities of $s\tau$ and sv advancing to be the same as the probabilities of τ and v advancing in our general model. Just as in the Bertrand case, we provide a program that calculate the profits, consumer surplus, and other statistics for the competitive model, as well as programs that calculate the analogous variables for the single–agent models. The parameter values that we set in calculation are described in Table 3.

Table 3:

Const.	Use	Typical Value
D	Vertical intercept of the demand curve	5
f	Fixed cost of production to all firms	2
γ	Capital–to–cost constant scale parameter	1

We note that this current profit function is, in many senses, an extreme alternative to the profit function used in the differentiated product model. In that case all firms have the *same* marginal costs but are differentiated by the quality of the product they produce; a quality which increases with successful research activity, and equilibrium is Nash in *prices*.

• **Section 5: Perfect Cartel and Social Planner Problems**

In order to characterize the equilibrium levels that we solve for in the

algorithm, it is useful to compare the decisions that result in the equilibrium with the decisions that would result in a similar industry controlled by perfectly collusive firms (such as one that might arise from a cartel with perfect monitoring), or alternatively, by a benevolent social planner. In both these cases, we are interested in finding out how the appropriate single agent would react when the technology is exactly the same as that faced by the MPN competitors, i.e., when the single agent faces exactly the same random entry costs, the same family of distributions for increments from investment, and the same scrap values for selling off plants. This will allow us to characterize the values of various variables, such as price–cost margins, consumer surplus, etc., under the single agent industry setups, and to compare these to the MPN case.

As discussed in Pakes–McGuire, both the cartel and the social planner problems are similar in the solution methodology used to find their equilibria. In both cases, we may think of the agent as maximizing a stochastic "profit" function. We define their profit goals as follows: the cartel seeks to maximize the EDV of producer surplus, while the planner seeks to maximize the EDV of social (producer and consumer) surplus. Thus, for both problems, we may define a static "profit" function $B[w]$, where $w \varepsilon W^{*}(N,\overline{w})$ is an industry structure, and then write down the value function in a manner similar to (2). The function $B[w]$ should represent the maximized static gain to surplus for any state; thus for the perfect cartel case, $B[w]$ is the one–shot industry profits, while for the social planner case, $B[w]$ is the one–shot consumer surplus. Recall that in Section 4, we described how to compute planer and cartel industry profits and consumer surplus for both of the static games that we discussed. Thus, we have already shown how to compute $B[w]$ for both of these one–shot games.

DIFFERENCES BETWEEN MPN AND SINGLE–AGENT SOLUTIONS.

We may now look at the computation of both single agent problems together, as

the equilibria can be computed with the same program, after substituting in different one–period return functions. There are three major computational differences between the single–agent equilibria and the MPN equilibria. These are: we have a different state space, we must look at exit and entry differently, and (since we do not need to form conditional probabilities of other people's behavior based on perceptions that are correct in equilibrium) the single agent computation can be set up as a contraction with modulus $\beta < 1$ (and hence must converge to a unique fixed point)

First of all, we note that in the MPN models, the state space was $W^* \times N$, as we were looking at each possible firm in each industry structure. In the single–agent models, the state space is W^*, as the agent controls the entire industry, so any industry structure (any $w \in W^*$) results in only one state, not in N states. However, in the single–agent models, any state results in an N–vector of investments and entry/exit decisions.

Secondly, we look at entry and exit. In the single–agent models, the agent chooses which firms it wants to have enter or exit and thus does not need to form perceptions about entry or exit, but rather controls these decisions herself, paying the entry fees and receiving the scrap value ϕ. In programming the MPN equilibria, we have used efficiency level 0 to represent both a firm that is at the lowest efficiency and an empty slot with no firm. For the single–agent problems, we cannot do this, because the agent receives scrap value ϕ for any active firm (even one at the lowest efficiency level) but not for an empty space with no firm. (In the MPN model, we did not care which of the 0's received scrap values, as they were going to be inactive from that point on.) Thus, we must code the lowest efficiency level for an active firm as efficiency level 1. This results in $\overline{w}$ for the single–agent case being one higher than $\overline{w}$ for the MPN case.

Thirdly, in the single–agent problem, we do not need to look at perceptions of the behavior of cohorts, as the agent controls all active firms at any time. As illustrated in Pakes–McGuire the value function for the planner and the cartel are calculated from the

following recursion. Consider any industry structure w, and suppose that firms w_1 through w_n are active. If $V^{i-1}(w_1,...,w_n)$ is the i—1th iteration of this function, and we have ordered the ω so that $w_i \leq w_{i-1}$, then we calculate the i^{th} iteration value function as

$$(19) \qquad V^i(w_1,...,w_n) = \max_{(1\leq q\leq n)} \overline{V}^i(w_1,...,w_q)$$

where

$$\overline{V}^i(w_1,...,w_i) = B(w_1,...,w_q) + (n-q)\phi + \max$$
$$\{\sup_{(x_1,...x_q)} -c\Sigma_j x_j + \Sigma_j V^{i-1}(w_1',...,w_q')p(w_1'|w_1,x_1,\nu)...p(w_q'|w_q,x_q,\nu)p(\nu);$$
$$\sup_{(x_1,...,x_q)} -c\Sigma_j x_j - x_e +$$
$$\Sigma_j V^{i-1}(w_1',...,w_q',W_- E)p(w_1'|w_1',x_1,\nu)...p(w_q'|w_q,x_q,\nu)p(w_e|\nu)p(\nu)\ \}.$$

If the first max over q for $q\in\{1,...,n\}$ is q*, then (n—q*) of the incumbents exit. The second max operator determines whether or not there is entry. That is the agent must decide which of the currently active firms to keep active and whether to allow entry, as a function of the random entry cost. Through the use of q, we are, in effect, forcing the agent to keep a higher efficiency firm active if it is keeping a lower efficiency firm active. The reader can verify that the operator defined in (19) is a contraction mapping with contraction modulus β (< 1).

The problem with applying standard recursive methods to solve for the fixed point of the operator in (19), is that at any iteration the solution is a complicated function of q, $x_1,...,x_N$. One can attempt to apply a non—linear search method, but note that by the nature of the problem the boundary constraints $[x_j \geq 0]$ will be binding for some variables. While elaborate non—linear search methods can be used to solve systems of equations with inequality constraints, they are very time—consuming and cumbersome to use. Thus, we modify the recursive method implicit in (19) to actually solve for our fixed point.

In order to avoid the above problem of having to solve a non—linear system of equations with binding boundary conditions, we proceed to solve this system in a similar

fashion to the solution for the MPN case. Recall the Calcval($\cdot$) function from (3). For any state w, and any *given choice* of q (the number of active firms) and given choice of entry, one can define w' for that choice as

$$w' = \left[w[1{:}q], \{0[q+1{:}N] \text{or} [W\_E]\}\right],$$

where $0[\cdot]$ represents a vector of zeros. Then, one can define an analogous function Calcval(w',n), which again measures firm n's EDV for each of the two possible realizations of the firm's investment process, using last iteration's values for the other actors' investments.

One can verify that for any [w',n], the first order conditions for the single agent are the same as for the MPN case, given the different definitions of Calcval. Thus, we can write the optimal x[w',n], using the first order conditions similar to (4), as

$$x[w',n] = \text{argmax}_{x \geq 0} [-cx + \beta[ax/(1+ax)]\text{Calcval}(w'+e(n),n) + \beta[1/(1+ax)]\text{Calcval}(w',n)],$$

assuming that the max exists by compactness and continuity. Thus, we can solve for the optimal x, using exactly equations (5) and (6). Recall that this x is optimal conditional on the choice of q (or equivalently, of w'). The agent will then find the optimum of these solutions over all possible q and entry rules. The q and entry rule that are optimal will then define the optimal policy at this state. Thus, we have implicitly defined a mapping T, which maps investment, entry and exit decisions and a value function, into new values of each of these variables.

Note that, by using this solution methodology, by have re–introduced into the problem the conditional distributions of other firms states, which is dependent on perceptions of their investment. Thus, T need not be a contraction mapping, unlike the

mapping defined in (19). However, a fixed point of T will maximize the EDV of future profits and thus be the choice of the single agent. One can verify this property by noting that because at a fixed point, perceptions and hence conditional distributions are realized, a fixed–point to T will also be a fixed–point to (19). However, since T is not a contraction mapping, there is no guarantee that it will converge to a fixed–point. For the values we have tried, T has converged. Note that if T fails to converge, the user may always return to using a contraction mapping technique such as (19), which will probably take a lot longer to solve each iteration, but which is guaranteed to converge.

• **Section 6: Descriptive Statistics Output Programs**

Once we have solved an industry for MPN and single–agent equilibria, we are interested in characterizing the equilibria, in order to examine how the industry is characterized for different parameter values. We have examined several statistics that are typically considered important determinants of an industry in the industrial organization literature. We can separate these statistics into three categories. The categories and statistics are:

INDIVIDUAL PERIOD STATISTICS.

- Number of periods with n firms active, n= 0,1,2,...;
- number of periods with exit;
- number of periods with entry;
- number of periods with both entry and exit;
- average percentage job creation (average over periods);
- average percentage job destruction (average over periods);
- average total investment(average over periods);
- average price/ marginal cost margin(average over periods); and

• average one–firm concentration ratio (average over periods).

FIRM LEVEL STATISTICS.

• Lifetime of each firm; and

• discounted returns earned by each firm.

AGGREGATE STATISTICS.

• Discounted consumer surplus (mean and variance);

• Discounted producer surplus (mean and variance); and

• Discounted total surplus (mean and variance).

We compute all of these statistics by simulating the industry from a user–specified starting industry structure. In order to find the decisions at each point, we use the equilibrium value functions, investment levels and entry/exit decisions to evaluate the optimal policies at the states resulting at any time period. The industry structure is updated from its user–specified initial value, depending on the realization of random variables whose distributions are conditional on the chosen equilibrium policies. We have separate (though similar) programs that evaluate these statistics for the MPN equilibrium and for the single–agent problems.

We calculate the first and second set of statistics in one program. We simulate the industry for a large number of periods (we typically use 10,000; but the user can specify the length desired). For many of the statistics, such as one–firm concentration ratios, we need to make use of the value of variables from the static game. Thus, in our static profit programs, we also compute the one–firm concentration ratios, consumer surplus, price/marginal cost margins. Except for consumer surplus (which we discussed in Section 4), these variables are all easy to compute, so we will not discuss how to compute them any further. For the firm–level statistics, we must keep track of each firm's position in the

current industry structure, and update these when firms exit or enter the market. Firms' total profits are calculated in discounted value form, counting period 0 for any firm as the period when it entered the market. Firms' positions are also needed to calculate job creation and destruction.

The third set of statistics differs from the first two sets in that it attempts to measure variables that are discounted sums over all periods. Since we are interested in the distributions of these surpluses, we repeatedly simulate the industry from the user–specified starting condition. Furthermore, because the surpluses are measured in aggregate discounted form, we do not simulate as many periods in any given run. Thus, we typically simulate the industry for 100 periods, and repeat the simulation process 200 times restarting it with the initial conditions each time. These welfare statistics are computed in separate programs from the other statistics because of the different nature of the simulation.

• **Section 7: Computer Implementation of the Algorithm**

The algorithm that we have presented in Section 2 is implemented directly in the Gauss computer programs we have developed; recall that users can access these programs by ftp, see the Overview at the beginning of the paper for details on accessing the program. In order to understand the program, though, we list which Gauss functions perform which of the operations described in the algorithm, as well as the Gauss variable names that correspond to the variables described in the algorithm. In this section, we only describe the main multi–agent equilibrium generation program, markov.eqm.

FUNCTIONS USED IN PROGRAM. The body of the main program cycles through, and solves the iterative problem for the different number of firms, starting with 'stfirm' firms, and going up to N firms. The user may specify how many firms to start with by changing

the variable 'stfirm' at the beginning of the program. If the user wishes to start with 2 or more firms, the data needed for the initial condition of the algorithm is taken from the file "markov.?ot" where ? is set equal to stfirm. The program automatically stores the last solved equilibirium for each 'tlnfirms'. Note if you run two programs with different parameter values but the same 'tlnfirm' the programme will only keep the last one. It is up to the user to rename the first program and store it elsewhere if she wishes.

The iterative problem is solved by repeatedly calling the function *contract*. *Contract* cycles through every possible state $(w,n) \varepsilon W^* \times N$, and calls *optimize* once for each state, in order to find the optimal investment level and associated value function for that state. Before calling *optimize* though, *contract* calls *chkentry*, which specifies the probability of a potential entrant wanting to enter, for any given competitors' efficiency levels $w \varepsilon W^*$. The reader may recall, from Section 2, that the values computed by *chkentry* are then used to calculate the optimal investment level, instead of recalculating these values for each element of a permutation cycle (indsutry structure). The actual calculation of the optimal investment level done in *optimize* is performed using the formulas that are derived in Section 2. The function calcval described in Section 2 is actually programmed into the Gauss function *calcval*, which calculates the same sum of value functions indicated in the algorithm description; *calcval* is performed separately for the cases of entry and no entry.

The coding function $enc_{m,\overline{w}}$ is programmed in the function *encode*; while the basic decoding function is programmed in *decode*. The table lookup decoding and encoding that are actually used to increase computational efficiency are programmed in *qdecode* and *qencode*. The tables used by *qdecode* and *qencode* are constructed in the main program, once for every n (number of firms) value. Because Gauss does not allow for arrays to begin with element 0, we must always add 1 when encoding N–tuples, in order that the encoded values can be used as indices for the X and V matrices. Correspondingly, in the function *decode* we first subtract 1 from the coded value before using the decoding algorithm that

we have proposed.

We have, thus far, left out one function from our discussion. This function, *update* is used to copy the value function and investment for the solution to the n firms problem for use as the starting values of the coresponding elements in the n+1 firms problem. This is used if the user specifies a starting firm size ("stfirm") smaller than N, in which case the program will have to compute equilibria for $n < N$ firms.

VARIABLES USED IN PROGRAM. Most of the variables used in the Gauss program correspond closely to the variables discussed in the algorithm description. We store most of the big variables we use as globals, because of the inefficiency of passing parameters in Gauss. In Table 4, we list the major variables used, along with the corresponding variable and constant in the algorithm, and a '*' if further explanation is required. We note that all of the variables below the second bar are parameters of the problem. These variables can be changed by the user, in order to test the effect of different parameter values. In order that they are easily accessible, these variables are all assigned their values at the top of the program, and can thus be accessed easily. In addition, the matrix Π can be changed by changing the parameters of the underlying static profit functions.

Table 4:

Gauss Variable	Algorithm Var. or Const.	Explanation
binom	Binom	
oldvalue / newvalue	V	*
oldx / newx	X	*
profit	Π	
isentry	$\lambda[w',n]$	
mask		*
dtable	Dtable	
etable	Etable	
wmax	# $[W^*]$	
entry_k	W_E	
stfirm / *nfirms / rlnfirms*	N	*
kmax	$\overline{w}$	
x_entryl	X_EL	

x_ entryh	X_EH
beta,delta,phi,a,c	β, δ, ϕ, a, c

We now provide the explanations of the names that may be unclear. First of all, we note that we require '*oldvalue*' and '*newvalue*' instead of just '*value*' and similarly for '*oldx*' and '*newx*' because the old values of these variables are used to determine their new values at any stage; this is the basis of the recursion routine. Thus, the new values are copied back into the old values after each iteration. The variable '*mask*' stores the list of possible increases in competitors' efficiency levels that can happen, conditioning on the outside alternative. So, if n = 3, for instance, *mask* = [0 0, 0 1, 1 0, 1 1]. Finally, we note that '*rlnfirms*' indicates the actual number of firms that we are solving for, while '*nfirms*' shows the number that we are solving for in this round, and 'stfirm' shows the number that we are solving for in the first round. Recall that the algorithm starts by solving the "stfirm" problem, and then using this solution to solve for the "stfirm + 1" firm problem, etc. So, '*rlnfirms*' would be the same throughout the program, while '*nfirms*' starts out equal to 'stfirm', and increases by 1 each round.

Bibliography

Anderson,S., de Palma,A., and J. Thisse (1991); Discrete Choice Theory of Product Differentiation, M.I.T. Press, Cambridge, Mass.

Berry, S. and A.Pakes (1993); "Some Applications and Limitations of Recent Advances in Empirical Industrial Organization: Merger Analysis", A.E.R., papers and proceedings.

Berry, S., Levinsohn, J., and A. Pakes (1992); "Automobile Prices in Market Equilibrium," N.B.E.R. discussion paper no. 4264.

Caplin, A. and B. Nalebuff (1991); "Aggregation and Imperfect Competition: the Existence

of Equilibrium" Econometrica, Vol. 59, pp. 25–59.

Ericson, R. and A. Pakes (1993); "An Alternative Theory of Firm and Industry Dynamics," Discussion Paper No. 1041 Cowles Foundation. Revision of Columbia University Discussion Paper 445 (1989).

Judd, K.(forthcoming); Numerical Methods in Economics, mimeo, the Hoover Institution.

Maskin, E. and J. Tirole (1988a) "A Theory of Dynamic Oligopoly, I: Overview and Quantity Competition with Large Fixed Costs," Econometrica, Vol. 56, pp. 549–569.

Maskin, E. and J. Tirole (1988b); "A Theory of Dynamic Oligopoly, II: Price Competition, Kinked Demand Curves, and Edgeworth Cycles," Econometrica, Vol. 56, pp. 571–599.

Pakes, A. (forthcoming); "The Estimation of Dynamic Structual Model Problems and Prospects Part II: Mixed Continuous–Discrete Control Models and Market Interactions." In Advances in Econometrics (110 page typescript) proceedings of the 6th World Congress of the Econometric Society, edited by J. J. Laffont and C. Sims.

Pakes, A. and P. McGuire (1993a); "Computing Markov Perfect Nash Equilibria: Numerical Implications of a Dynamic Differentiated Product Model," mimeo, Yale University.

Pakes, A. and P. McGuire (1993b); "Computing Markov Perfect Nash Equilibria II: Approximations," mimeo, Yale University.