Solution Methods for Markov Perfect Nash Equilibria of Continuous-time, Finite-state Stochastic Games

Ulrich Doraszelski
Hoover Institution

Kenneth L. Judd
Hoover Institution
National Bureau of Economic Research

April, 2004 - Very Preliminary; do not quote without permission

1 Introduction

Discrete-time stochastic games with discrete states have been usefully applied in industrial organization, often using the algorithm presented in Pakes-McGuire (1994) (PM1) or more recent adaptations. However, the range of applications of these methods is limited by the large computational burden of computing Markov perfect equilibria (MPE) of these dynamic games. Pakes-McGuire (2001) (PM2) points out that computing the players' expectations over future states is subject to a "curse of dimensionality" since the computational costs are exponential in the number of states. This paper examines the alternative of continuous-time stochastic games with discrete states, and shows that continuous-time games have several advantages over discrete-time games. First, there is no curse of dimensionality with continuous-time games. The key fact is that there are no simultaneous changes in states in continuous-time whereas there are in discrete-time models, and this implies that the multidimensional integrals in discrete-time games are replaced by much smaller sums in continuous-time games. Furthermore, space management, a topic of considerable importance for stochastic games, can also be made more efficient for continuous-time games, resulting in a further 20% to 40% increase in speed. The absence of a curse of dimensionality implies that we can examine far more complex problems than is feasible with discrete-time models. For example, we solve a 13-state example, a case where our continuous-time approach uses 1.76 seconds per iteration whereas the corresponding discrete-time example uses over 35 hours per iteration, more than 36,000 times slower. Second, continuous-time games have many conceptual advantages. In particular, continuous-time games avoid unnatural and artificial features that arise in models with both discrete time and discrete states. Finally, there is no reason to not use continuous-time games since they are no more complex to describe than discrete-time models once one uses techniques for controlled continuous-time, finite-state Markov processes.

The key fact is easy to see. Consider a discrete-time game with N players and where each player can be in one of K states tomorrow. Computing the expected value of a player will require the summation of K^N numbers. In contrast, in a continuous-time process, this large sum is replaced by a finite sum of $N(K-1)$ numbers, a dramatically smaller sum. Furthermore, one will need to look up each of the K^N numbers using a complex indexing scheme; see Gowrisankaran (1999) for a discussion of this problem. Furthermore, since $(K-1)\,N$ is often small, one can precompute and store their locations, avoiding the costly index computations executed in PM1, PM2, and Gowrisankaran (1999). Because of the curse

of dimensionality, PM1 choose small values for N and/or K; in contrast, the efficiencies of the continuous-time case allows more freedom in specifying K, the transition processes of players, and the number of players.

Discrete-time finite-state stochastic games have been used frequently in industrial organization. Ericson and Pakes (1995) (EP) develop a general dynamic model of oligopoly. In each period, incumbent firms compete in the product market, and decide how much to invest and even whether to remain in the industry, and potential entrants decide whether to enter the industry. The EP model captures two key findings of the empirical literature on industry dynamics (e.g., see Dunne et al, 1988), and Davis and Haltiwanger, 1999). First, entry and exit occur simultaneously. Second, there is heterogeneity among firms, and this heterogeneity evolves endogenously in response to random occurrences, for example, in the investment process. Since the stochastic games are too complex to be solved analytically, (PM1) and (PM2) provide algorithms to compute a Markov perfect equilibrium (MPE) of this stochastic game.

More recent work has used stochastic games to model a wide range of topics in industrial organization, including advertising (Doraszelski, 2003) capacity accumulation (Besanko and Doraszelski), collusion (Fershtman and Pakes, 2000, de Roos, 2004), competitive convergence (Langhor, 2003), consumer learning (Ching, 2002) learning by doing (Benkard, 2000, Besanko et al., 2003), mergers (Gowrisankaran, 1999), network externalities (Markovich, 1999), and R&D (Auerswald, 2001, Song, 2002). Most recently, stochastic games have been applied in fields other than industrial organization, e.g., international trade (Erdem and Tybout, 2003) and finance (Goetler et al., 2004).[1]

Since the computational burden of a discrete-time model increases exponentially in the number of state variables, existing applications are limited to a handful of firms and restrict firm heterogeneity to a few dimensions. For example, in the typical application firms may differ from each other either in terms of their production capacity or in product quality, but not in both. Breaking the curse of dimensionality opens the way to compute the equilibria of much richer stochastic games. Moreover, the reduced computational costs gives the researcher the ability to compute equilibria for many different parameterizations and learn how equilibrium varies with the parameters. Moreover, if the goal is to conduct policy experiments based on the parameter estimates, then the researcher again needs to compute the equilibrium as fast as possible.

[1] A survey of several applications of stochastic games in industrial organization can be found in Pakes (2000).

Avoiding the curse of dimensionality has several substantive advantages for modelers. allows us to specify a richer range of stochastic outcomes over finite intervals of time. For example, can easily accommodate more than one investment choice per firm. Also, depreciation can be modelled in a more natural manner - constant relative depreciation instead of constant absolute depreciation. Discrete-time models often specify a small discount factor, implicitly assuming that the period of time is on the order of a year or two. We can separate discount rate from period length; no need to use small β; the rate of convergence of the algorithm is naturally related to the stochastic properties of the transition process and not dictated by the discount rate. This is clear since a continuous-time model is equivalent to a β very close to 1, but the continuous-time algorithm does not take forever to converge. easier to parameterize. We propose a computational strategy for solving continuous-time models and describe their massive computational advantages.

The continuous-time model also allows us to pursue coding strategies that are not possible for discrete-time. We present a data storage method which substantially improves speed. This is an example of how there are often synergies in algorithm development: improving one aspect of an algorithm often leads to new opportunities to further improve efficiency.

This paper describes the basic elements of continuous-time finite-state games and compares the computational burdens of discrete-time and continuous-time games. Section 2 reviews the basic discrete-time stochastic game and the Gauss-Jacobi (PM1) and Gauss-Seidel algorithms used to solve such games. Section 3 follows with a description of the basic continuous-time stochastic game and the comparable Gauss-Seidel algorithm. Section 4 presents both the discrete-time and continuous-time formulations of the example used in PM1. Section 6 presents computational details and compares the performance of the two models. Section 7 argues that continuous-time models have several conceptual advantages as well as the computational advantages. Section 8 discusses some generalizations of the model and algorithm. Section 9 concludes.

2 Discrete-time Model

We first review the canonical discrete-time stochastic game with discrete states. This type of game is used in EP and is generally called a "stochastic game" (Filar and Vrieze, 1997, and Basar and Olsder, 1999) in the dynamic game literature. We let Ω be the finite set of possible states; the state of the game in period t is $\omega_t \in \Omega$. We assume that there are N players. Player i's control (action) in period t is $x_t^i \in \mathbb{X}^i(\omega_t)$ where $\mathbb{X}^i(\omega)$ is the set of

feasible actions for player i if the game is in state ω. We make no specific assumptions about $\mathbb{X}^i(\omega)$, which may be a scalar or a vector, continuous or discrete. The collection of player actions at time t is $x_t = \left(x_t^1, \ldots, x_t^N\right)$. We follow the usual convention of letting x_t^{-i} denote $\left(x_t^1, \ldots, x_t^{i-1}, x_t^{i+1}, \ldots, x_t^N\right)$.

The state follows a controlled first-order Markov process. More specifically, if the state is ω and the players choose actions x at time t then the probability that ω' is the time $t+1$ state is $\Pr(\omega'|\omega, x)$. In many applications, the state ω is a vector and is partitioned into $(\omega^1, \ldots, \omega^N)$, where ω^i denotes the (one or more) coordinates of the state vector that are controlled by player i, and that the transition process of each player's states is independent of the other players' states and actions; in this case, the law of motion can be written as

$$\Pr(\omega'|\omega, x) = \prod_{i=1}^{N} \Pr^i\left(\omega'|\omega^i, x^i\right).$$

where $\Pr^i(\omega'|\omega^i, x^i)$ is the transition probability law for the states controlled by player i. The special case of independent transitions allows us to easily illustrate the advantages of continuous over discrete time but, as we will point out below, the insights are not limited to this case.

We decompose payoffs into two components. First, player i receives a per-period payoff equal to $\pi^i(x_t, \omega_t)$ when the current state is ω_t and the players choose actions x_t. The term $\pi^i(x_t, \omega_t)$ captures the flow of profits that depend solely on the current state and current controls. For example, if ω_t is a list of firms' capacities and x_t are their output decisions and investment efforts, then $\pi^i(x_t, \omega_t)$ represents earnings from product market competition net of ongoing investment expenses that don't depend on whether or not a transition has occurred. Sometimes, a payoff will depend on the lagged state and payoff. Therefore, we assume a second part to the payoff, $\Phi^i(x_t, \omega_t, \omega_{t+1})$, that equals the change in the wealth of player i at the beginning of period $t+1$ if the state moves from ω_t to ω_{t+1} and controls were x_t. The $\Phi^i(x_t, \omega_t, \omega_{t+1})$ term allows us to model events such as the sale of equipment. For example, a firm may spend money searching for a buyer for equipment it wants to sell; this cost does not depend on whether a buyer is found and is included in π. When it succeeds in finding a buyer and the state changes, firm i receives a payment equal to $\Phi^i(x_t, \omega_t, \omega_{t+1})$. In general, the payment $\Phi^i(x_t, \omega_t, \omega_{t+1})$ depends on the nature of the transition (e.g., selling some or all capacity) and may be affected by the sales effort of the incumbent firm, x_t^i, just prior to the sale. Our representation of the model's payoffs allows us to model many features of a dynamic model, including entry and exit.

Players discount future payoffs using a discount factor $\beta \in [0,1)$. The objective of player i is to maximize the discounted sum of cash flows

$$E\left\{\sum_{t=0}^{\infty}\beta^{t}\pi^{i}\left(x_{t},\omega_{t}\right)+\sum_{m=1}^{\infty}\beta^{T_{m}}\Phi^{i}\left(x_{T_{m}-1},\omega_{T_{m}-1},\omega_{T_{m}}\right)\right\} \tag{1}$$

where T_m is the random time of the m'th change in the state, x_{T_m-1} is the players' strategy choice vector just before the m'th change, ω_{T_m-1} is the state just before the m'th change, and ω_{T_m} is the state just after the m'th change.

Following the tradition in dynamic games, we will assume that each player's actions depends only on the current state of the game. Let $V^i(\omega)$ denote the expected net present value to player i if the current state is ω. Suppose that player i's opponents use the strategies $X^{-i}(\omega)$. Then the Bellman equation for player i is

$$V^{i}\left(\omega\right)=\max_{x^{i}}\pi^{i}\left(x^{i},X^{-i}\left(\omega\right),\omega\right)+\beta E_{\omega'}\left\{\Phi^{i}\left(x^{i},X^{-i}\left(\omega\right),\omega,\omega'\right)+V^{i}\left(\omega'\right)|\omega,x^{i},X^{-i}\left(\omega\right)\right\}$$

where ω denotes the current state. The Bellman equation for player i adds his current cash flow $\pi^i(x^i, X^{-i}(\omega), \omega)$ to his discounted expected future cash flow,

$$E_{\omega'}\left\{\Phi^{i}(x^{i},X^{-i}\left(\omega\right),\omega,\omega')+V^{i}\left(\omega'\right)|\omega,x^{i},X^{-i}\left(\omega\right)\right\},$$

over all possible future states ω'. Player i's strategy is the solution to

$$X^{i}\left(\omega\right)=\max_{x^{i}}\pi^{i}\left(x^{i},X^{-i}\left(\omega\right),\omega\right)+\beta E_{\omega'}\left\{\Phi^{i}\left(x^{i},X^{-i}\left(\omega\right),\omega,\omega'\right)+V^{i}\left(\omega'\right)|\omega,x^{i},X^{-i}\left(\omega\right)\right\}, \tag{2}$$

Each player has his own version of (1) and (2). The system of equations defined by the collection of (1) and (2) over all players defines a Markov Perfect Equilibrium (MPE). See Doraszelski and Satterthwaite (2001) for conditions under which a MPE exists.

2.1 Symmetry, Anonymity and Storage Issues

Many applications of dynamic games assume symmetric payoffs and costs. Symmetric games assume that the payoff functions π^i and Φ^i and the transition law $\Pr(\omega_i'|\omega, x)$ for firm i depends only on firm i's state and the distribution of the other firms' states. Many authors then assume that there exists a similarly symmetric Markov Perfect Equilibrium, and search for policy and value functions with the same symmetries. Symmetric games are easier to solve, a fact emphasized in PM1 and related work, but is not essential for most of our results.

The curse of dimensionality is present in any discrete-time model but not in the continuous-time model we describe below. Our examples will be symmetric but only to make easy comparisons with the existing literature.

However, we will also note below that continuous-time models can better exploit symmetry. Therefore, we need to discuss some details for solving symmetric games. In a symmetric MPE, if $V^1(\omega)$ denotes firm 1's value in state ω, then firm i's value is given by

$$V^i(\omega^1, \ldots, \omega^{i-1}, \omega^i, \omega^{i+1}, \ldots, \omega^N) = V^1(\omega^i, \ldots, \omega^{i-1}, \omega^1, \omega^{i+1}, \ldots, \omega^N)$$

and similarly for firm i's policy. Therefore, symmetry and anonymity allow us to focus on the problem of firm 1. Furthermore, anonymity says that firm 1 does not care about the identity of its competitors, only their distribution. Hence, for all permutations π on $\{2, 3, \ldots, N\}$

$$V^1(\omega^1, \omega^{\pi(2)}, \omega^{\pi(3)}, \ldots, \omega^{\pi(N)}) = V^1(\omega^1, \omega^2, \omega^3, \ldots, \omega^N)$$

and similarly for the policy function. Therefore, we need only examine states of the form $\left(\omega^1, \omega^2, \omega^3, \ldots, \omega^N\right)$ where $\omega^1 \leq \omega^2 \leq \ldots \leq \omega^N$. This means that there are far fewer distinct states in symmetric games than in the corresponding asymmetric games. More precisely, the number of distinct states is $(N + M - 1)!/(N!(M - 1)!)$, whereas the number of distinct states is M^N without symmetry.

To understand the importance of this we need to discuss the problem of computer storage. Any algorithm solving for the Nash equilibrium values of $V^i(\omega)$ and $X^i(\omega)$ must store these values in some table that we denote $\mathbb{M}$. Table 1 displays such a table. Each row of $\mathbb{M}$ stores the information related to a state ω; let $\psi(\omega)$ denote that row. Table 1 displays a simple example for a three-firm game. Row $\psi(\omega)$ contains the information for state ω, with the first three columns listing the components of ω, the second triple of columns listing the three values, $(V^1(\omega), V^2(\omega), V^3(\omega))$, and the last three rows listing the strategies $(X^1(\omega), X^2(\omega), X^3(\omega))$.

Table 1: Canonical Tabular Storage Scheme in $\mathbb{M}$

	Column								
	1	2	3	4	5	6	7	8	9
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
row $\psi(\omega)$:	ω_1	ω_2	ω_3	$V^1(\omega)$	$V^2(\omega)$	$V^3(\omega)$	$X^1(\omega)$	$X^2(\omega)$	$X^3(\omega)$
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Any algorithm needs to know how to find the information for an arbitrary state ω. For example, the integral $E\left\{V^1\left(\omega'\right)|\omega,x\right\}$ needs to access all values of $V^j\left(\omega\right)$ that could correspond to firm one's value in the successor states ω'. More precisely, $E\left\{V^j\left(\omega'\right)|\omega,x\right\}$ is the sum

$$\sum_{\omega'\in\Omega(\omega,x)}\Pr\left(\omega'|\omega,x\right)V^j\left(\omega'\right)$$

where

$$\Omega\left(\omega,x\right)\equiv\left\{\omega'|\Pr\left(\omega'|\omega,x\right)>0\right\}$$

is the set of states ω' that occur with positive probability in the next period if the current state is ω and the current players' actions are x. To compute this sum, the algorithm must find the rows and columns with the relevant information, implying that the sum is really

$$\sum_{\omega'\in\Omega(\omega,x)}\Pr\left(\omega'|\omega,x\right)\mathbb{M}\left(\psi\left(\omega'\right),J\left(1,\omega',\omega\right)\right) \tag{3}$$

where $J\left(1,\omega',\omega\right)$ is the column in row $\psi\left(\omega'\right)$ that contains the next period's value for firm 1 if the current state is ω and ω' occurs next. This expression displays all the computation that must occur.[2]

Any algorithm needs to use an efficient storage scheme; in particular, the function $\psi\left(\omega\right)$ needs to be easy to compute. This is easy in some cases. If, for example, there are M possible states for each firm and we needed to examine all possible permutations of $\{1,2,3,\ldots,M\}$ (as would be the case in asymmetric games) then we could use

$$\psi\left(\omega\right)=\omega_1+\left(\omega_2-1\right)M+\left(\omega_3-1\right)M^2+...+\left(\omega_N-1\right)M^{N-1} \tag{4}$$

where ω is nothing more than the base M representation of $\psi\left(\omega\right)$. In this case, the first row of the table contains the information for state $(1,1,1)$, the second row represents $(2,1,1)$, and the successive rows represent $(3,1,1)$, $(4,1,1)$, ..., $(M,1,1)$, $(1,2,1)$, $(2,2,1)$, ..., $(M,2,1)$, etc. This choice of ψ allows us to easily find the information related to state ω. This is essentially the type of storage scheme used by default for arrays in programming languages like Fortran and C.

[2]One should not that $J\left(\omega',1\right)$ is not a constant. Suppose that the current state is $(1,1,3)$ and that firm one jumps to state 2. The state in the next period is $\omega'=(2,1,3)$ which has its information stored in the state $(1,2,3)$. Therefore, firm one tomorrow has the same value that firm two has in the state $(1,2,3)$. Therefore, $J\left(\omega',1\right)=2$. Keeping track of this indexing adds some complexity to the programming, but this detail is in all algorithms. Therefore, we will focus on other details that are more important nad central to the distinction between discrete- and continuous-time models.

Symmetric games, however, are more difficult to represent. Symmetry can be exploited to reduce the number of distinct states that are examined, but ψ will not be a simple function like the one in (4). We could use (4) but then most rows in $\mathbb{M}$ would be ignored and the space advantages of symmetry would be lost. PM1 use one scheme to store the value functions efficiently whereas Ericson and Pakes (1995) and Gowrisankaran (1999) propose different schemes. All methods have the structure we display above in Table 1 where $\bar{\Omega}$ is the set of distinct states, $\left|\bar{\Omega}\right|$ be the number of such states, and $\psi : \bar{\Omega} \to \left|\bar{\Omega}\right|$.

Equation (3) displays all the computation that must occur in evaluating $E\left\{V^j\left(\omega'\right)|\omega, x\right\}$.and emphasizes that there are two kinds of computational costs involved in computing the integral $E\left\{V^j\left(\omega'\right)|\omega, x\right\}$. The first is the summation of each term and the second is the computation of the addresses. No matter how we store the states, some time will be used to compute addresses, and in complex cases such as the case of symmetric games the address computation will be even more costly than displayed in (4).

2.2 Algorithms for Discrete-time Games

PM1 uses a value function iteration approach to compute the equilibrium value and policy functions; in the mathematics literature, it is called a (block) Gauss-Jacobi method. We will describe the PM1 method, but we will also describe the more popular and often superior block Gauss-Seidel scheme that we use for both continuous- and discrete-time models. First, we make initial guesses for the value functions, $V^i(\omega)$, and policy functions, $X^i(\omega)$, $i = 1, \ldots, N$, at each state ω. Then these guesses are updated as the algorithm proceeds through the state space in some order. At each state $\omega \in \Omega$, we update the value and policy functions for each player i, $i = 1, \ldots, N$, in either a Gauss-Seidel or Gauss-Jacobi fashion. That is, given current guesses V^i and x^i for the value and policy functions, we compute updated guesses, which we label $\hat{V}^i$ and $\hat{x}^i$, as follows

$$\begin{aligned}
\hat{X}^i(\omega) \leftarrow\ & \arg\max_{x^i} \pi^i\left(x^i, X^{-i}(\omega), \omega\right) \\
& +\beta E\left\{\Phi^i\left(x^i, X^{-i}(\omega), \omega, \omega'\right) + V^i\left(\omega'\right)|\omega, x^i, X^{-i}(\omega)\right\} \qquad (5) \\
\hat{V}^i(\omega) \leftarrow\ & \max_{x^i} \pi^i\left(x^i, X^{-i}(\omega), \omega\right) \\
& +\beta E\left\{\Phi^i\left(x^i, X^{-i}(\omega), \omega, \omega'\right) + V^i\left(\omega'\right)|\omega, x^i, X^{-i}(\omega)\right\} \qquad (6)
\end{aligned}$$

where $i = 1, \ldots, N$. Note that the old guesses for opponents' strategies, $X^{-i}(\omega)$, and the old guess for player i's value at ω, $V^i(\omega)$, are used when computing the new guesses $\hat{V}^i(\omega)$ and $\hat{X}^i(\omega)$. This procedure is, therefore, a Gauss-Jacobi scheme at each state $\omega \in \Omega$.

There are two ways to update the $V^i(\omega)$ and $X^i(\omega)$ functions. PM1 and Gauss-Jacobi methods would compute $\hat{V}^i(\omega)$ and $\hat{X}^i(\omega)$ for all ω before replacing the old guesses for $V^i(\omega)$ and $X^i(\omega)$ with their new values. In contrast, our Gauss-Seidel scheme first computes $\hat{V}^i(\omega)$ and $\hat{X}^i(\omega)$ for each player i, $i = 1, \ldots, N$, and them immediately replaces the old guesses for state ω with the new guesses for state ω, as in

$$\begin{aligned} X^i(\omega) &\leftarrow \hat{X}^i(\omega) \\ V^i(\omega) &\leftarrow \hat{V}^i(\omega) \end{aligned}$$

This updating procedure is executed for each state $\omega \in \Omega$ in the specified order. This algorithm is an example of what is called block Gauss-Seidel in the literature on nonlinear equations, where the block is the set of $V^i(\omega)$ and $X^i(\omega)$ for all firms at an ω. (See Judd, 1999, for a more extensive discussion of these Gaussian methods for solving nonlinear equations.) The Gauss-Seidel algorithm cycles through the states $\omega \in \Omega$ until the changes in the value and policy functions are deemed small.

Some type of Gaussian scheme is probably necessary since the size of the problem is so large that alternatives, such as Newton's method, are probably infeasible for the whole problem. However, there are many possible adaptations of the block Gauss-Seidel algorithm, and there are surely better ones than this. This is the most commonly used approach and we will use it for both discrete- and continuous-time algorithms. This paper focusses on issues related to computing expectations. Future work will examine alternative specifications of the blocks, methods within blocks, and acceleration methods.

2.3 The Curse of Dimensionality

The key difficulty is computing the conditional expectations in the updates in equations (5) and (6). Setting $\Phi^i(x, \omega, \omega') = 0$ to simplify the notation, the conditional expectation is

$$E\left\{V^i(\omega')|\omega, x\right\} = \sum_{\omega'} V^i(\omega') \Pr(\omega'|\omega, x).$$

Note that this involves summing over all future states ω' such that $\Pr(\omega'|\omega, x) \neq 0$.

Perhaps the simplest case arises when transitions are independent and each transition is restricted to going one step up, one step down, or not moving, i.e. $\omega^{i\prime} \in \{\omega^i - 1, \omega^i, \omega^i + 1\}$. Then the expectation consists of 3^N terms,

$$E\left\{V^i(\omega')|\omega, x\right\} = \sum_{\{\omega' \in \Omega : |\omega^{i\prime} - \omega^i| \leq 1, i=1,\ldots,n\}} V^i(\omega') \prod_{i=1}^{N} \Pr^i\left(\omega^{i\prime}|\omega^i, x\right).$$

More generally, if a player can be at one of K states tomorrow and transitions are independent across players, then the expectation involves summing over K^N terms. Since K^N grows exponentially with N, computing the expectation over future states is subject to a curse of dimensionality.

3 Continuous-time Model

We now describe the continuous-time stochastic game model with discrete states that we will study. The only substantive difference is that time is now continuous. As with the discrete-time model, the horizon is infinite, the state at time t is $\omega_t \in \Omega$ where $\Omega \subset \mathbb{Z}^M$ is the state space, there are N players, and player i's control (action) at time t is denoted by x_t^i.

The key difference is that the state evolves probabilistically according to a controlled continuous-time, finite-state Markov process. In discrete time the time path of the state is a sequence. In the continuous-time model, it is a piecewise-constant, right-continuous function of $t \geq 0$. Jumps occur at random times according to a Poisson process which depends on the controls and the state. At time t, the hazard rate of some jump occurring is $\phi(x_t, \omega_t)$. When a jump occurs, the state changes according to the transition probability $f(\omega'|\omega_t, x_t)$, where ω denotes the old state and $\omega' = \lim_{s \to t^+} \omega_s$ is the new one. That is, $f(\omega'|\omega, x)$ characterizes the transitions of the first-order Markov process. We assume $f(\omega|\omega, x) = 0$, which says that any jump will alter the state. Since a jump from a state to itself does not change the game, we simply ignore it and instead adjust, without loss of generality, the hazard rate of a jump so that $f(\omega|\omega, x) = 0$.

We will often use a differential notation to express the transition dynamics. Over a short interval of time of length $h > 0$, the law of motion is

$$\begin{aligned} \Pr(\omega_{t+h} \neq \omega_t | \omega_t, x_t) &= \phi(x_t, \omega_t) h + o(h^2) \\ \Pr(\omega_{t+h} = \omega' | \omega_t, x_t, \omega' \neq \omega_t) &= f(\omega' | \omega_t, x_t) + o(h^2) \end{aligned}$$

In the special case of independent transitions, the coordinates of the state that are controlled by player i evolve according to

$$\begin{aligned} \Pr^i(\omega_{t+h} \neq \omega_t | \omega_t, x_t) &= \phi^i(x_t, \omega_t) h + o(h^2) \\ \Pr^i(\omega_{t+h}^i = \omega' | \omega_t^i, x_t^i, \omega' \neq \omega_t^i) &= f^i(\omega' | \omega_t^i, x_t^i) + o(h^2) \end{aligned}$$

Here $\phi(x,\omega) = \sum_{i=1}^{N} \phi^i(x^i,\omega^i)$ is the hazard rate of a jump occurring, i.e., the total hazard of a change in the state.

This notation displays the critical fact about continuous-time Markov processes on finite states: during a small time interval dt, there will be (with probability infinitesimally close to one) at most one jump. In the discrete-time model where firms follow independent jump processes, we must keep track of all possible combinations of players' state transitions between time t and time $t+1$. The possibility of more than one firm changing states disappears in the continuous-time model and results in simpler, and computationally more tractable, models.

The remaining aspects of the continuous-time model are essentially the same as in the discrete-time model. The payoff to player i consists of two components. First, player i receives a payoff flow equal to $\pi^i(x_t,\omega_t)$. Second, $\Phi^i(x_{T_m^-},\omega_{T_m^-},\omega_{T_m^+})$ is the instantaneous change in the wealth of player i when the state moves from $\omega_{T_m^-}$ to $\omega_{T_m^+}$ at a random time T_m and controls were $x_{T_m^-}$. Similar to discrete time, $\pi^i(x_t,\omega_t)$ captures profits from product market competition net of investment and $\Phi^i(x_{T_m^-},\omega_{T_m^-},\omega_{T_m^+})$ things like the scrap value a firm receives upon exiting the industry or the setup cost that a firm pays upon successful entry into the industry. In the continuous-time model, there is a stock and flow distinction between π^i and Φ^i since π^i represents a flow of money and Φ^i is a change in the stock of wealth.

Players discount future payoffs at the common rate $\rho > 0$. The objective of player i is

$$E\left\{\int_0^\infty e^{-\rho t}\pi^i(x_t,\omega_t)\,dt + \sum_{m=1}^{\infty} e^{-\rho T_m}\Phi^i\left(x_{T_m^-},\omega_{T_m^-},\omega_{T_m^+}\right)\right\},$$

where T_m is the random time of the m'th jump in the state, $x_{T_m^-}$ is the control just before the m'th jump, $\omega_{T_m^-}$ is the state just before the m'th jump, and $\omega_{T_m^+}$ is the state just after the m'th jump.The Bellman equation for player i is similar to the one in discrete time. Let $X(\omega)$ represent the players strategies. Over a short interval of time of length $\Delta t > 0$, player i solves the dynamic programming equation

$$\begin{aligned} V^i(\omega) &= \max_{x^i} \pi^i\left(x^i,x^{-i},\omega\right)dt + (1-\rho\Delta t)\left(1-\phi\left(x^i,X^{-i}(\omega),\omega\right)\Delta t - o\left(\Delta t^2\right)\right)V^i(\omega) \\ &\quad + (1-\rho\Delta t)\left(\phi\left(x^i,X^{-i}(\omega),\omega\right)\Delta t + o\left(\Delta t^2\right)\right) \\ &\qquad \times E\left\{\Phi^i\left(x^i,X^{-i}(\omega),\omega,\omega'\right) + V^i(\omega')\,|\omega,x^i,X^{-i}(\omega)\right\}, \end{aligned}$$

which, as $\Delta t \to 0$, simplifies to the Bellman equation

$$\begin{aligned}\rho V^i(\omega) = \max_{x^i} \quad & \pi^i\left(x^i, X^{-i}(\omega), \omega\right) - \phi\left(x^i, X^{-i}(\omega), \omega\right) V^i(\omega) \\ & + \phi\left(x^i, X^{-i}(\omega), \omega\right) E\left\{\Phi^i(x^i, X^{-i}(\omega), \omega, \omega') + V^i(\omega') \,|\omega, x^i, X^{-i}(\omega)\right\}.\end{aligned}$$

Hence, $V^i(\omega)$ can be interpreted as the asset or option value (to player i) of participating in the game. This option is priced by requiring that the opportunity cost of holding it, $\rho V^i(\omega)$, equals the current cash flow, $\pi^i(x^i, X^{-i}(\omega), \omega)$, plus the expected capital gain or loss flow. The latter is given by the expected capital gain or loss conditional on a jump occurring, $E\left\{\Phi^i(x^i, X^{-i}(\omega), \omega, \omega') + V^i(\omega') \,|\omega, x^i, X^{-i}(\omega)\right\} - V^i(\omega)$, times the likelihood of doing so, $\phi(x^i, X^{-i}(\omega), \omega)$.

In the special case of independent movements, player i solves the problem

$$\begin{aligned}V^i(\omega) = & \max_{x^i} \pi^i\left(x^i, X^{-i}(\omega), \omega\right) dt \\ & - (1-\rho dt)\left(1 - \phi\left(x^i, X^{-i}(\omega), \omega\right) dt\right) V^i(\omega) \\ & + (1-\rho dt) E\left\{\Phi^i\left(x^i, X^{-i}(\omega), \omega, \omega^{i\prime}, \omega^{-i}\right) + V^i\left(\omega^{i\prime}, \omega^{-i}\right) |\omega^i, x^i\right\} \phi^i\left(x^i, \omega^i\right) dt \\ & + (1-\rho dt) \sum_{j \neq i} E\left\{\Phi^i\left(x^i, X^{-i}(\omega), \omega, \omega^{j\prime}, \omega^{-j}\right) + V^i\left(\omega^{j\prime}, \omega^{-j}\right) |\omega^j, x^j\right\} \phi^j\left(x^j, \omega^j\right) dt,\end{aligned}$$

$$\begin{aligned}\Pi^i\left(\omega, x^i\right) = & \pi^i\left(x^i, X^{-i}(\omega), \omega\right) dt - (1-\rho dt)\left(1 - \phi\left(x^i, X^{-i}(\omega), \omega\right) dt\right) V^i(\omega) \\ & + (1-\rho dt) E\left\{\Phi^i\left(x^i, X^{-i}(\omega), \omega, \omega^{i\prime}, \omega^{-i}\right) + V^i\left(\omega^{i\prime}, \omega^{-i}\right) |\omega^i, x^i\right\} \phi^i\left(x^i, \omega^i\right) dt \\ & + (1-\rho dt) \sum_{j \neq i} E\left\{\Phi^i\left(x^i, X^{-i}(\omega), \omega, \omega^{j\prime}, \omega^{-j}\right) + V^i\left(\omega^{j\prime}, \omega^{-j}\right) |\omega^j, x^j\right\} \phi^j\left(x^j, \omega^j\right) dt,\end{aligned}$$

which simplifies to the Bellman equation

$$\begin{aligned}\rho V^i(\omega) = \max_{x^i} \quad & \pi^i\left(x^i, x^{-i}, \omega\right) - \phi\left(x^i, X^{-i}(\omega), \omega\right) V^i(\omega) \\ & + \phi^i\left(x^i, \omega^i\right) E\left\{\Phi^i\left(x^i, X^{-i}(\omega), \omega, \omega^{i\prime}, \omega^{-i}\right) + V^i\left(\omega^{i\prime}, \omega^{-i}\right) |\omega^i, x^i\right\} \\ & + \sum_{j \neq i} \phi^j\left(x^j, \omega^j\right) E\left\{\Phi^i\left(x^i, X^{-i}(\omega), \omega, \omega^{j\prime}, \omega^{-j}\right) + V^i\left(\omega^{j\prime}, \omega^{-j}\right) |\omega^j, x^j\right\}.\end{aligned}$$

3.1 Computational Strategy

In its basic form, our computational strategy adapts the Gauss-Seidel algorithm to a continuous-time setting. That is, to compute the equilibrium value and policy functions, we combine a block Gauss-Seidel scheme across states with a Gauss-Jacobi scheme for computing players'

values at each states. More specifically, given a state $\omega \in \Omega$, we replace equations (5) and (6) by

$$\begin{aligned}
\hat{X}^i(\omega) \leftarrow & \arg\max_{x^i} \pi^i\left(x^i, X^{-i}(\omega), \omega\right) \\
& -\phi\left(x^i, X^{-i}(\omega), \omega\right) V^i\left(x^i, X^{-i}(\omega), \omega\right) \\
& +\phi\left(x^i, X^{-i}(\omega), \omega\right) E\left\{\Phi^i\left(x^i, X^{-i}(\omega), \omega, \omega'\right) + V^i(\omega') \,|\omega, x^i, X^{-i}(\omega)\right\}, \qquad (7) \\
\hat{V}^i(\omega) \leftarrow & \frac{1}{\rho + \phi\left(\hat{X}^i(\omega), X^{-i}(\omega), \omega\right)} \pi^i\left(\hat{X}^i(\omega), X^{-i}(\omega), \omega\right) \\
& + \frac{\phi\left(\hat{X}^i(\omega), X^{-i}(\omega), \omega\right)}{\rho + \phi\left(\hat{X}^i(\omega), X^{-i}(\omega), \omega\right)} \qquad (8) \\
& \quad \times E\left\{\Phi^i\left(\hat{X}^i(\omega), X^{-i}(\omega), \omega, \omega'\right) + V^i(\omega') \,|\omega, \hat{X}^i(\omega), X^{-i}(\omega)\right\},
\end{aligned}$$

where $i = 1, \ldots, N$, to update the value and policy functions for each player i. The remainder of the algorithm proceeds as before. Note that by dividing through by $\rho + \phi(\hat{x}^i(\omega), x^{-i}(\omega), \omega)$, we ensure that equation (8) is contractive for any one firm (holding fixed the strategies of the other firms) since

$$\frac{\phi(\hat{X}^i(\omega), X^{-i}(\omega), \omega)}{\rho + \phi(\hat{X}^i(\omega), X^{-i}(\omega), \omega)} < 1$$

is a contraction factor as long as ϕ is bounded above. The equations in (7) and (8) is the continuous-time analog of the discrete-time Bellman map but with a key difference. In the discrete-time case, the discount rate β is a constant contraction factor whereas here the contraction factor depends on the strategy.

3.2 Breaking the Curse of Dimensionality

The main advantage of continuous-time models now becomes clear. The simplest case arises when transitions are independent and each transition is limited to going one up or down. When the transitions are independent, the multidimensional integral representing the expectation over future states decomposes into N one-dimensional integrals over future states. Moreover, each of these conditional expectations now consist of only 2 terms representing the two possible transitions for each firm. Here we exploit the fact that, unlike discrete time, there is no need to explicitly consider the possibility of remaining in the same state. In fact,

setting $\Phi^i(x, \omega, \omega') = 0$ to simplify the notation, we have

$$E\left\{V^i\left(\omega^{j\prime}, \omega^{-j}\right)|\omega^j, x^j\right\} = \sum_{\{\omega' \in \Omega: |\omega^{j\prime} - \omega^j| = 1\}} V^i\left(\omega^{j\prime}, \omega^{-j}\right) f^j\left(\omega^{j\prime}|\omega^j, x^j\right). \tag{9}$$

In this continuous-time model, we need to sum over $2N$ terms rather than the 3^N terms necessary in the discrete-time model. More generally, if a player can be at one of K states tomorrow and transitions are independent across players, then the expectation in the continuous-time model involves summing over $(K-1)N$ terms whereas the discrete-time model cause this expectation to consist of K^N terms. Since $(K-1)N$ grows linearly rather than exponentially with N, computing the expectation over future states is no longer subject to the curse of dimensionality as we increase the number of players.

This observation is still relevant in many cases where the firms experience a common shock. Suppose, for example, that there are both firm-specific states, such as firm-specific productivity or capacity, as well as aggregate states, such as industry demand, which affect all players. Then the conditional expectation $E\left\{V^i\left(\omega^{j\prime}, \omega^{-j}\right)|\omega^j, x^j\right\}$ can be decomposed into changes in the aggregate states and changes in the firm-specific states. Again, in a continuous-time model, we will not have changes in both the aggregate and firm-specific states. Therefore, the pieces of $E\left\{V^i\left(\omega^{j\prime}, \omega^{-j}\right)|\omega^j, x^j\right\}$ representing no change in aggregate states will be as in (9). The changes in aggregate states will just add some more summands to (9) where the number of additional terms is just the number of possible successor states for the aggregate variables.

3.3 Exploiting Symmetry with Precomputed Pointers

The first advantage of continuous time is that we avoid the curse of dimensionality when computing conditional expectations. We next make a second improvement possible only in the continuous-time case. Table 1 displays the typical storage scheme used to store equilibrium values and strategies, and noted that considerable effort is necessary to compute addresses via the $\psi(\omega)$ function. Each time we compute a sum or expectation at a state ω we will need to compute the addresses of the neighbors of ω. One way to economize on this repeated computation of neighbors' addresses is by precomputing the addresses of the neighbors of ω and storing them along with values and strategies on the row that contains the other information for state ω. For example, in the case of a two-player game in continuous time, we could augment $\mathbb{M}$, the array containing the game's information, with six extra

columns as in the following matrix

Table 2: Storage with precomputed pointers

Row	Column							
	1	2	3		10	11	...	15
	⋮	⋮	⋮	⋮	⋮	⋮	...	⋮
i	ω_1	ω_2	ω_3	...	$\psi(\omega + e_1)$	$\psi(\omega + e_2)$	...	$\psi(\omega - e_3)$
	⋮	⋮	⋮	⋮	⋮	⋮	...	⋮

where $e_1 = (1, 0, 0)$, $e_2 = (0, 1, 0)$, etc. For N players, each row will contain $4N$ numbers: the N states for each player (listed in ascending order when we have a symmetric game), the N values of the players, the N strategies, and the $2N$ numbers indicating the location of the $2N$ possible successor states[3]. This is a reasonable demand on space. A simple storage scheme would only double the space needs relative to the standard procedure. A more sophisticated approach would use only half a computer word to store each address, and thereby use only half as much space on the precomputed addresses.

This storage scheme reduces running times since the addresses of successor states are computed once and stored. A casual examination of the $\psi(\omega)$ functions used indicates that the savings should be substantial. We will explore the savings in an example below.

This approach could also be done in theory for discrete-time games but is generally not feasible since the number of successor states would be 3^N instead of $2N$. The space demands would exceed available space except for small discrete-time games. Therefore, this is another advantage that is essentially available only to continuous-time games.

4 Example: The Pakes-McGuire Quality Ladder Model

We will use the quality ladder model described in PM1 to test our computational points. This is a very particular model that we use only because it was used in PM1. Some of our points are model independent, but some will depend on PM1 model. However, it is a sensible example since experience with other models is similar. We begin with the original discrete-time model and then reformulate it in continuous time. We want to focus on a simple and clear example that highlights the computational issues. We also want to avoid existence problems that may arise with entry and exit and the complexities that may be

[3]For some ω there are fewer states. In that case, we insert a nonsensical value, such as a negative number, indicating that there is no such state.

introduced to solve these issues (see Doraszelski and Satterthwaite, 2001, for details and a way to resolve these difficulties). Therefore, we eliminate the entry and exit aspects in PM1, and set $\Phi^i(x, \omega, \omega') = 0$. This will allow us to make clean comparisons between the Gauss-Seidel method now typically used for discrete time games and our continuous-time methods. In a later section we will describe how to introduce entry and exit into our continuous-time model.

4.1 Discrete-time Model

The quality ladder model assumes that there are N firms with differentiated products engaged in price competition. Each product also has a quality dimension that may change with firm investment or with depreciation. Therefore, firm i will produce good i which will have quality ω^i. We first describe product demand and then turn to the investment process.

Demand Quality is assumed to be discrete with $\omega^i \in \{1, 2, ..., L\}$. The state space is thus $\Omega = \{1, ..., L\}^N \subset \mathbb{Z}^N$. Each consumer purchases at most one unit of one good. The utility consumer k derives from purchasing good i is $g(\omega^i) - p^i + \epsilon^{ik}$, where ϵ^{ik} represents taste differences among consumers. $g(\omega^i)$ maps product quality into the consumer's valuation according to

$$g(\omega^i) = \begin{cases} \omega^i, & \omega^i \leq \omega^*, \\ \omega^* + \ln\left(2 - e^{-(\omega^i - \omega^*)}\right), & \omega^i > \omega^*. \end{cases}$$

There is an outside good, good 0, which has utility ϵ^{0k}. We assume that the idiosyncratic taste parameters ϵ^{0k}, ϵ^{1k}, ...,ϵ^{Nk} are independently and identically extreme value distributed across individuals; therefore, the demand for firm i's product is

$$q^i(p^1, \ldots, p^N; \omega) = m \frac{\exp\left(g(\omega^i) - p^i\right)}{1 + \sum_{j=1}^N \exp\left(g(\omega^j) - p^j\right)},$$

where $m > 0$ is the size of the market (the measure of consumers).

Price competition In each period, firm i knows the quality of his and his opponents' goods, and chooses the price p^i of good i to maximize profits, thereby solving

$$\max_{p^i \geq 0} q^i(p^1, \ldots, p^N; \omega)\left(p^i - c\right),$$

where $c \geq 0$ is the unit cost of production. The first-order condition is of firm i is

$$0 = q_i^i(p^1, \ldots, p^N; \omega)\left(p^i - c\right) + q^i(p^1, \ldots, p^N; \omega).$$

It can be shown that there exists a unique Nash equilibrium $\left(p^{1*}(\omega), \ldots, p^{N*}(\omega)\right)$ of the product market game; see Caplin and Nalebuff (1991). The Nash equilibrium at any particular state ω can be computed easily by numerically solving the system of first-order conditions, but we will not do so here.

Law of motion The state ω^i represents the quality of the product sold in the current period. However, the quality in the next period is determined by $x^i \geq 0$, firm i's investment in quality improvement, and depreciation. The outcome of the investment and depreciation processes is assumed to be stochastic, and make the game dynamic.

In each period, firm i spends $x^i \geq 0$ to increase future product quality. If the investment is successful, then the quality increases by one level. Expenditures in investment increase the probability of quality improvement; in particular, we assume that the probability of success is $p(x^i) = \frac{\alpha x^i}{1+\alpha x^i}$, where $\alpha > 0$ is a measure of the effectiveness of investment. If a firm is hit with a depreciation shock, quality will decline by one level; we assume that a firm is hit by a depreciation shock with probability $\delta \in [0, 1]$.

We differ from the PM1 quality ladder model in that our depreciation is firm-specific and independent across firms whereas PM1 assume an industry-wide depreciation shock. We do this to focus on the key computational details related to the curse of dimensionality in discrete-time models. Aggregate shocks can be added to continuous-time as easily as discrete-time models, but there is no difference in their impact on computational issues. The key problems arise with firm-specific changes and so we focus on those.

Combining the investment and depreciation processes, the quality of firm i's product changes according to the transition function $\Pr^i(\omega'|\omega, x)$. The transition function $\Pr^i(\omega'|\omega^i, x^i)$ is the probability that firm i will have product quality ω' tomorrow given that it has product quality ω^i today. Assume

$$\Pr^i(\omega'|\omega^i, x^i) = \begin{cases} \frac{(1-\delta)\alpha x^i}{1+\alpha x^i}, & \omega' = \omega^i + 1, \\ \frac{1-\delta+\delta\alpha x^i}{1+\alpha x^i}, & \omega' = \omega^i, \\ \frac{\delta}{1+\alpha x^i}, & \omega' = \omega^i - 1 \end{cases}$$

if $\omega^i \in \{2, \ldots, L-1\}$. Clearly, firm i cannot move further down (up) from the lowest

(highest) product quality. We therefore set

$$\begin{aligned}
\Pr^i(\omega'|\omega^1, x^i) &= \begin{cases} \frac{(1-\delta)\alpha x^i}{1+\alpha x^i}, & \omega' = \omega^2, \\ \frac{1+\delta\alpha x^i}{1+\alpha x^i}, & \omega' = \omega^1 \end{cases} \\
\Pr^i(\omega^{i\prime}|\omega^L, x^i) &= \begin{cases} \frac{1-\delta+\alpha x^i}{1+\alpha x^i}, & \omega' = \omega^L, \\ \frac{\delta}{1+\alpha x^i}, & \omega' = \omega^{L-1} \end{cases} .
\end{aligned}$$

Payoff function The per-period payoff of firm i is derived from the Nash equilibrium of the product market game and given by

$$\pi^i(x, \omega) \equiv q^i(p^1(\omega), \ldots, p^N(\omega); \omega)(p^i(\omega) - c) - x^i,$$

where we have subtracted investment x^i from the profit from product market competition.

Parameterization. We parameterize our example using the values in PM1. The effectiveness of investment is $\alpha = 3$, the depreciation probability is $\delta = 0.7$, market size is m, and the marginal cost of production is $c = 5$. We will examine various discount factors, including the choice $\beta = 0.925$, which corresponds to a yearly interest rate of 7.5 percent, made in PM1. Finally, we will first let $L = 18$ with $\omega^l = 3l - 10$, $l = 1, \ldots, L$, and $\omega^* = 12$, as done in Pakes et al. (1993), but will also examine other cases.

4.2 Continuous-time Model

In the interest of brevity, we start by noting that the details of product market competition remain unchanged. In the continuous-time model we can thus reinterpret $\pi^i(x, \omega)$ as the payoff flow of firm i.

Law of motion. Next we turn to the state-to-state transitions. We want to compare discrete-time and continuous-time games. Therefore, we will use the same transition process as described for our discrete-time model. Therefore, the hazard rate for the investment project of firm i is successful is given by the hazard rate $h(x^i) = \alpha x^i/(1+\alpha x^i)$, the same choice as our probability function in the discrete-time model. This is the appropriate choice since the expected time to the first success is $h\,(x)^{-1}$ in both discrete and continuous time. Similarly, we take $\delta > 0$ to be the depreciation hazard where δ was the depreciation probability in the discrete-time model.

Jumps in the i'th coordinate of the state thus occur according to a Poisson process with hazard rate

$$\phi^i(x^i) = \frac{\alpha x^i}{1+\alpha x^i} + \delta,$$

and when a jump occurs, the ith coordinate of the state changes according to the transition probability

$$f^i(\omega'|\omega^i, x^i) = \begin{cases} \frac{\phi^i(x^i)}{\delta+\phi^i(x^i)} & , \quad \omega' = \omega^{l+1}, \\ \frac{\delta}{\delta+\phi^i(x^i)} & , \quad \omega' = \omega^{l-1} \end{cases}$$

if $\omega' = \omega^l$, where $l \in \{2, \ldots, L-1\}$. Since firm i cannot move further down (up) from the lowest (highest) product quality, we set

$$\begin{aligned} \phi^i(x^i) &= \frac{\alpha x^i}{1+\alpha x^i} \\ f^i(\omega^2|\omega^1, x^i) &= 1 \end{aligned}$$

and

$$\begin{aligned} \phi^i(x^i) &= \delta, \\ f^i(\omega^{L-1}|\omega^L, x^i) &= 1. \end{aligned}$$

Parameterization. We can easily match the parameters of the continuous-time model to those of the discrete-time model. First, if Δ is the unit of time for one period in the discrete-time model, then β and ρ are related by $\beta = e^{-\rho\Delta}$ and $\rho = -(\ln\beta)\,\Delta^{-1}$. Second, if the discrete-time depreciation rate is $e^{-\delta\Delta}$ then the continuous-time hazard rate for depreciation rate is δ.

5 Stopping Rules

Since we are using Gauss-Seidel schemes to compute equilibria, convergence is linear and the choice of stopping rule is generally important. A fair comparison between the discrete- and continuous-time methods requires a careful application of accuracy estimates and stopping rules. In this section we discuss the standard adaptive approach to error estimation as applied to our models

Let $V^i(\omega)$ ($X^i(\omega)$) denote the value to (strategy of) player $i = 1, \ldots, N$ in state $\omega \in \Omega$ at the beginning of an iteration and let $\hat{V}^i(\omega)$ ($\hat{X}^i(\omega)$) denote the value (strategy) at the end

of iteration i. Define the L_∞-norm of a set of value functions $V^i(\omega)$ to be

$$\|V\|_\infty = \max_{i=1,\ldots,N} \max_{\omega\in\Omega} |V^i(\omega)|.$$

and the L_2-norm to be

$$\|W\|_2 = \left(\frac{1}{|\Omega|S} \sum_{i=1}^{N} \sum_{\omega\in\Omega} \left(V^i(\omega)\right)^2 \right)^{1/2}.$$

We need a measure of when two value functions are close. We also want it to be unit-free and to describe the relative difference. Therefore, we define the L_∞-relative difference between $\hat{V}$ and V to be

$$E_\infty\left(\hat{V}, V\right) = \left\| \frac{\hat{V} - V}{1 + |\hat{V}|} \right\|_\infty.$$

and the L_2-relative difference to be

$$E_2\left(\hat{V}, V\right) = \left\| \frac{\hat{V} - V}{1 + |\hat{V}|} \right\|_2.$$

We similarly define $E_\infty\left(\hat{X}, X\right)$ and $E_2\left(\hat{X}, X\right)$.

In practice, we compute $E_\infty\left(\hat{V}, V\right)$ (or $E_2\left(\hat{V}, V\right)$) and stop when $E_\infty\left(\hat{V}, V\right)$ (or $E_2\left(\hat{V}, V\right)$) falls below some target. However, we really want to know $E_\infty\left(\hat{V}, V_\infty\right)$ (and $E_2\left(\hat{V}, V_\infty\right)$ where V_∞ is the limit of our sequence of guesses and, presumably, a solution. To do this we need to apply some ideas from the theory of sequences. If a sequence of points, x_t, satisfies the contraction relation

$$\|x_{t+1} - x_t\| \leq \theta \|x_t - x_{t-1}\|$$

then the distance to the limit x_∞ satisfies

$$\|x_{t+1} - x_\infty\| \leq \|x_t - x_{t-1}\| / (1 - \theta)$$

Define $\Delta_t = \|x_t - x_{t-1}\|$ and suppose that $\Delta_{t+1} = \theta\Delta_t$. Then, for all t and s, $\theta^s = \Delta_t/\Delta_{t-s}$ which implies that

$$\theta = \left(\frac{\Delta_t}{\Delta_{t-s}} \right)^{1/s} \tag{10}$$

In our computations, we are often in the position of observing Δ_t without knowing θ. Equation (10) gives us a way to estimate θ at iterate t using the previous s iterates since it is a

long run average of the recent rate of convergence. A more conservative approach would be to use

$$\max\left\{\left(\frac{\Delta_\tau}{\Delta_{\tau-s}}\right)^{1/s} | t_1 < \tau < t\right\}$$

which is the worst s-iterate performance over the previous $t - t_1 - s$ iterations.

Next, define $\varepsilon = \|x_t - x_\infty\|$. Then, approximately, we have

$$\varepsilon_t = \Delta_t / (1 - \theta) \tag{11}$$

Our task is to find some iteration s such that we have achieved a target accuracy $\bar{\varepsilon}$. If our current value of $\|x_t - x_{t-1}\|$ is Δ_t then we need s more iterations to achieve an error of $\bar{\varepsilon}$ where s satisfies

$$\bar{\varepsilon} = \theta^s \Delta_t / (1 - \theta)$$

This implies that s is given by

$$S(\theta) = \frac{\ln(\bar{\varepsilon}/\Delta_t) + \ln(1 - \theta)}{\ln \theta} \tag{12}$$

A common practice is to stop when Δ_t is below some target $\bar{\Delta}$. However, the true error could be a factor $(1 - \theta)^{-1}$ greater than $\bar{\Delta}$. This factor is 10 if $\theta = .9$ or 100 if $\theta = .99$. A careful strategy would examine the history of Δ_t to estimate θ and then proceed at least s extra steps. So, suppose that the objective is to reduce error to $\bar{\varepsilon}$. Then if $\Delta_t = \bar{\varepsilon}$, $S(\theta)$ more iterates to push error below Table 3 gives some examples of what this requires for alternative values of θ.

Table 3: Necessary Extra Iterations

θ	.9	.95	.97	.98	.985	.99
s	22	58	115	194	278	458

6 Computational Performance Comparisons

This section presents computational results for the discrete- and continuous-time examples. We will look at N-firm cases of the quality ladder model. Even though this is one specific example of a stochastic game, it is useful for many purposes. First, the results related to the curse of dimensionality are clearly robust since they simply involve floating point operations related to computing a multidimensional integral. The computational burden of such computations depends on neither the functional form nor the parameter values.

Also, the N-firm results should be viewed as results for games with N states since the dimensionality in the integrals is the key feature. Therefore, a game with three firms, each with two products and each product having its own capacity and quality corresponds to a 12-firm game in our set of examples. Second, the results related to the speed of convergence may depend on parameter values but there is no reason to believe that our example is atypical. Also, our approach to dealing with convergence issues is also a contribution to the economics literature on solving dynamic games. Third, we will discuss the performance of alternative stopping rules, and test our adaptive strategy on our example.

6.1 Computation Time per Iteration

We first examine the computational performance of computing one iteration in the Gauss-Seidel schemes we use. These results will also be the same for Gauss-Jacobi approaches since they are both a fixed number of state-wise computations.

Table 4 displays the results. We report running times for both code that was optimized by the compiler and code for which the optimization was turned off. We do this since the no-optimization code is more likely to correspond to basic intuition, but optimized code is what one should definitely use in practice. Since optimized code will substantially reduce the cost of computing large sums, it is important to document that the continuous-time approach still enjoys a substantial advantage even when code is optimized. Running times.

Table assumes that there are N firms, each with one state that can have M=9 values. Table 4 reports the number of resulting distinct states and the number of unknowns, which equals one value and one policy per firm per state. Table 4 reports the time used to compute one iteration of the Gauss-Seidel scheme for one firm. We report the percentage of time used to compute each sum for both continuous and discrete time. We also report the percentage of time per iteration used to compute the sum. This was essentially 100% when we solved the discrete-time model without code optimization, but was substantially less for the continuous-time cases with and without code optimization. In all continuous-time games, less than half the computer time was spent on computing the sums that arise in the firms' Bellman equations. We find that code optimization substantially accelerates computation but does not produce an advantage for the discrete-time model.

We also examine the importance of time used to compute addresses. We compare the usual approach where all addresses are repeatedly computed versus the precomputed address approach we describe above. We find that the precomputed pointers substantially reduce

the amount of time used to compute the sums; in fact, this feature alone reduced time by about 20 to 40%.

Table 4: Time per state per form per iteration

Compiler Optimization Off, M=9

			Time per sum (10^{-6} seconds)								
			Discrete		Cont. time: pointers						
N	Distinct	Un-			Compute		Store		Disc./	P'ter	Total
	states	knowns	time	%	time	%	time	%	Cont.	gain	gain
2	45	90	.98	34	.50	23	1.9	15	1.3	1.16	1.5
3	165	495	2.4	58	.54	25	1.8	15	1.9	1.21	2.2
4	495	1980	6.2	79	.64	28	1.8	17	3.5	1.26	4.4
5	1287	6435	17	91	.72	31	1.8	18	7.9	1.29	10
6	3003	18018	46	96	.80	33	1.8	19	20	1.31	26
7	6435	45045	130	100	.96	38	1.9	22	52	1.32	68
8	1.3(4)	1.0(5)	360	100	1.0	40	1.9	24	140	1.34	187
9	2.4(4)	2.2(5)	1000	100	1.1	42	2.0	26	380	1.36	517
10	4.4(4)	4.4(5)	2900	100	1.2	43	2.0	27	1036	1.38	1430
11	7.6(4)	8.3(5)	8100	100	1.3	45	2.1	29	2846	1.39	3961
12	1.2(5)	1.5(6)	2.3(4)	100	1.4	46	2.1	30	7864	1.41	1.1(4)
13	2.0(5)	2.6(6)	6.6(4)	100	1.4	48	2.2	33	2.1(4)	1.40	3.0(4)

Compiler Optimization On, M=9

			Time per sum (10^{-7} seconds)								
			Discrete time		Cont. time: pointers				Disc./		
N	Distinct	Un-			Compute		Store		Cont.	P'ter	total
	states	knowns	time	%	time	%	time	%	time	gain	gain
2	45	90	5.3	55	2.9	42	1.9	35	1.42	1.22	1.7
3	165	495	11	74	2.8	44	1.9	35	2.32	1.22	2.8
4	495	1980	26	87	3.0	48	2.0	43	4.60	1.34	6.2
5	1287	6435	67	97	3.4	53	2.2	46	10.6	1.37	14.6
6	3003	1.8(4)	178	100	3.8	55	2.2	45	26.3	1.39	36.4
7	6435	4.5(4)	526	100	3.9	53	2.4	48	69.9	1.44	101
8	1.3(4)	1.0(5)	1310	100	4.3	55	2.6	50	176.2	1.43	251
9	2.4(4)	2.2(5)	3840	100	4.8	62	2.8	53	492	1.44	709
10	4.4(4)	4.4(5)	1.1(4)	100	5.4	64	2.6	44	1.3(3)	1.52	1950
11	7.6(4)	8.3(5)	3.0(4)	100	5.6	67	3.2	56	3.5(3)	1.48	5179
12	1.2(5)	1.5(6)	8.2(4)	100	5.8	68	3.6	60	9.5(3)	1.44	1.4(4)
13	2.0(5)	2.6(6)	2.4(5)	100	6.3	69	3.8	61	2.6(4)	1.39	3.6(4)
14											

Except for the case of a few states, the continuous-time model was significantly faster. Even when there are only five firms (states) the continuous-time approach is more than 10 times faster. Most of the gain is from avoiding the curse of dimensionality in discrete-time integrals, but the precomputed addresses, a strategy that is feasible only for continuous-time games, also makes a significant contribution.

6.2 Number of Iterations

Table 4 showed that the curse of dimensionality is severe for discrete-time games but not for continuous-time games, and that each iteration of the continuous-time games is far faster than the corresponding discrete-time game. However, this does not prove that continuous-time games are faster to solve since a game is not solved until the iterations in (7,8) converge. We next discuss the difference in the rate of convergence of the Gauss-Seidel method for the two models.

There are good reasons to think that the continuous-time game will need more iterations before convergence. Suppose that the strategic elements in our games were eliminated; in that case, our problem reduces to a disjoint set of dynamic programming problems. In that case, normal value function iteration (a Gauss-Jacobi scheme) would converge at rate β in discrete time. The continuous-time algorithm would not have a uniform contraction factor but would have a different contraction factor at each state. Equation (8) implies that at state ω the contraction factor would be

$$\eta(\omega, X) = \frac{\phi\left(X(\omega), \omega\right)}{\rho + \phi\left(X(\omega), \omega\right)}$$

The quantity $\eta(\omega, X)$ has a simple interpretation: it is the expected present value of a dollar delivered at the next time the state changes if the current state is ω and the players follow the strategies in X. The case of $\rho \ll \phi = 1$ illustrates this case since then

$$\eta = \frac{1}{\rho + 1} \approx 1 - \rho = 1 + \ln \beta \approx \beta$$

If the hazard rate $\phi\left(\omega, X\left(\omega\right)\right)$ is small, then $\eta(\omega, X)$ will also be small and there will be a strong contraction aspect to a value function iteration approach. However, $\eta(\omega, X)$ could be close to one if the hazard rate is large, in which case value function iteration would be very slow.

These facts leads us to worry about the performance of our Gauss-Seidel algorithm. We will use the quality ladder example to display the relative performance of the Gauss-Seidel algorithm for discrete- and continuous-time models, and to illustrate some basic ideas

regarding stopping rules. The only way to learn about these methods is to examine an example. The results we find will depend somewhat on the example. However, our discussion of the approach to stopping rules is a robust strategy for diagnosing convergence for any algorithm, discrete or continuous time.

Tables 5, 6, and 7 display convergence results for our example. We first executed (7,8) until the L_∞ errors $E_\infty(X_t, X_{t+1})$ and $E_\infty(V_{t+1}, V_t)$ failed to decrease any further. In all cases, the iterations continued until $E_\infty(X_t, X_{t+1}) < 10^{-13}$and sometimes continued until $E_\infty(X_t, X_{t+1}) < 10^{-15}$. The final iterates were considered "true" solutions, V_* and X_*, since they satisfied the equilibrium equations to better than 12 significant digits.

We then computed the error measure $E_\infty(V_t, V_*)$ and the other error measures for each iterate t. Each row corresponds to the error value "tol". The Linf-iter column corresponds to the value of $E_\infty(V_t, V_{t+1})$ and the Linf-truth column corresponds to $E_\infty(V_t, V_*)$. The L2-inf and L2-truth columns correspond to the L_2 criterion. The tables report the iterate when these various error criterion were satisfied.

Table 5 displays results for the case of $N = 3$ firms, each having $M = 18$ possible values for its quality state, and for $\beta = 0.925$, 0.98, 0.99, 0.995. Table 6 displays results for the case of $N = 6$ firms, with $M = 9$, and for $\beta = 0.925$, 0.98, 0.99. The discrete-time model took too long to converge to our required target of 12 digit accuracy for $\beta = 0.995$. Table 7 displays the performance of the continuous-time model for those cases.

We see that the continuous-time model needs more iterations to converge, and that this gap increases slightly as we increase β (decrease ρ). This is because the η factors in the continuous-time model substantially exceed the β parameter in the discrete-time games.

Even though the continuous-time model needs more iterations, the increase in the number of iterations is small relative to the other advantages of continuous-time formulations. In particular, When we compare the results in these tables with Table 4, we see that the advantage in total time for the continuous-time model in the six-firm cases is about 10.

6.3 Convergence Criterion and Error Tests

We always want to continue the iterations in (7,8) until the distance from the true solution, $E_\infty(V_t, V_*)$, is small. In practice, we will not know $E_\infty(V_t, V_*)$. Therefore, in practice we must continue until $E_\infty(V_t, V_{t+1})$ satisfies some criterion that uses information from iteration t and earlier. Examination of the results in Tables 5, 6, and 7 show that the convergence formulas we presented above do an excellent job once we get below a modest convergence

Table 5: Convergence: Dependence on Discount Factors - M=18, N=3,

Number of Iterations to Prespecified Tolerance in Discrete Time:

Dtime	**β=0.925**				**β=0.98**				**β=0.99**				**β=0.995**			
tol	Linf-iter	Linf-truth	L2-iter	L2-truth	Linf-iter	Linf-truth	L2-iter	L2-truth	Linf-iter	Linf-truth	L2-iter	L2-truth	Linf-iter	Linf-truth	L2-iter	L2-truth
E-01	27	52	4	33	13	189	6	142	14	324	7	272	18	605	10	546
E-02	54	74	35	57	149	261	92	219	167	469	97	419	244	895	63	837
E-03	76	95	59	78	228	331	185	289	358	608	307	559	580	1175	520	1117
E-04	97	115	80	99	298	401	256	359	500	747	450	698	873	1453	814	1395
E-06	136	154	120	139	437	539	395	498	778	1025	729	976	1431	2010	1373	1952
E-08	176	194	161	179	576	678	534	637	1056	1303	1007	1253	1988	2567	1930	2509

Number of Iterations to Prespecified Tolerance in Continuous Time:

Ctime	**ρ =-ln 0.925**				**ρ =-ln 0.98**				**ρ =-ln 0.99**				**ρ =-ln 0.995**			
tol	Linf-iter	Linf-truth	L2-iter	L2-truth	Linf-iter	Linf-truth	L2-iter	L2-truth	Linf-iter	Linf-truth	L2-iter	L2-truth	Linf-iter	Linf-truth	L2-iter	L2-truth
E-01	16	47	4	14	35	180	6	94	46	324	7	208	60	598	17	450
E-02	31	124	16	85	53	437	29	349	67	832	40	719	85	1618	53	1475
E-03	97	183	49	145	147	679	95	591	144	1322	101	1209	149	2602	123	2459
E-04	160	241	121	203	432	919	345	832	672	1810	558	1697	988	3582	843	3439
E-06	276	357	238	319	915	1401	828	1314	1651	2785	1538	2672	2966	5542	2823	5399
E-08	392	473	354	435	1397	1883	1309	1795	2626	3760	2513	3647	4926	7502	4783	7359

Ratio of Discrete to Continuous Time

D/C	**β=0.925**				**β=0.98**				**β=0.99**				**β=0.995**			
tol	Linf-iter	Linf-truth	L2-iter	L2-truth	Linf-iter	Linf-truth	L2-iter	L2-truth	Linf-iter	Linf-truth	L2-iter	L2-truth	Linf-iter	Linf-truth	L2-iter	L2-truth
E-01	1.69	1.11	1.00	2.36	0.37	1.05	1.00	1.51	0.30	1.00	1.00	1.31	0.30	1.01	0.59	1.21
E-02	1.74	0.60	2.19	0.67	2.81	0.60	3.17	0.63	2.49	0.56	2.43	0.58	2.87	0.55	1.19	0.57
E-03	0.78	0.52	1.20	0.54	1.55	0.49	1.95	0.49	2.49	0.46	3.04	0.46	3.89	0.45	4.23	0.45
E-04	0.61	0.48	0.66	0.49	0.69	0.44	0.74	0.43	0.74	0.41	0.81	0.41	0.88	0.41	0.97	0.41
E-06	0.49	0.43	0.50	0.44	0.48	0.38	0.48	0.38	0.47	0.37	0.47	0.37	0.48	0.36	0.49	0.36
E-08	0.45	0.41	0.45	0.41	0.41	0.36	0.41	0.35	0.40	0.35	0.40	0.34	0.40	0.34	0.40	0.34

Table 6: Convergence: Dependence on Discount Factors - M=18, N=6,

Number of Iterations to Prespecified Tolerance in Discrete Time:

	β=0.925				β=0.98			
tol	**Linf-iter**	**Linf-truth**	**L2-iter**	**L2-truth**	**Linf-iter**	**Linf-truth**	**L2-iter**	**L2-truth**
E-01	28	53	11	31	41	255	6	202
E-02	54	73	34	52	216	327	143	279
E-03	75	94	54	72	294	396	246	349
E-04	95	114	74	92	364	465	317	418
E-06	135	154	113	132	501	602	454	555
E-08	176	194	153	172	639	740	592	693

Number of Iterations to Prespecified Tolerance in Continuous Time:

	ρ =-ln 0.925				ρ =-ln 0.98			
tol	**Linf-iter**	**Linf-truth**	**L2-iter**	**L2-truth**	**Linf-iter**	**Linf-truth**	**L2-iter**	**L2-truth**
E-01	20	82	3	26	34	671	4	391
E-02	42	169	12	105	64	1089	25	834
E-03	123	250	65	194	556	1492	249	1239
E-04	207	329	145	275	991	1894	734	1641
E-06	367	488	312	434	1798	2698	1545	2445
E-08	526	648	471	593	2601	3501	2348	3248

Ratio of Discrete to Continuous Time:

	β=0.925				β=0.98			
tol	**Linf-iter**	**Linf-truth**	**L2-iter**	**L2-truth**	**Linf-iter**	**Linf-truth**	**L2-iter**	**L2-truth**
E-01	1.40	0.65	3.67	1.19	1.21	0.38	1.50	0.52
E-02	1.29	0.43	2.83	0.50	3.38	0.30	5.72	0.33
E-03	0.61	0.38	0.83	0.37	0.53	0.27	0.99	0.28
E-04	0.46	0.35	0.51	0.33	0.37	0.25	0.43	0.25
E-06	0.37	0.32	0.36	0.30	0.28	0.22	0.29	0.23
E-08	0.33	0.30	0.32	0.29	0.25	0.21	0.25	0.21

Table 7: Convergence: Low Discount Factors - M=18, N=6,

Number of Iterations to Prespecified Tolerance in Discrete Time:

	β= 0.99				β= 0.995			
tol	**Linf-iter**	**Linf-truth**	**L2-iter**	**L2-truth**	**Linf-iter**	**Linf-truth**	**L2-iter**	**L2-truth**
E-01	88	535	10	437				
E-02	400	670	188	586				
E-03	571	800	480	717				
E-04	703	930	620	847				
E-06	962	1188	880	1106				
E-08	1221	1447	1139	1365				

Number of Iterations to Prespecified Tolerance in Continuous Time:

	ρ =-ln 0.99				ρ =-ln 0.995			
tol	**Linf-iter**	**Linf-truth**	**L2-iter**	**L2-truth**	**Linf-iter**	**Linf-truth**	**L2-iter**	**L2-truth**
E-01	50	1335	5	879	68	2507	7	1905
E-02	81	2234	27	1802	99	4365	28	3788
E-03	699	3101	219	2672	246	6157	151	5582
E-04	1729	3965	1289	3536	2724	7942	2128	7367
E-06	3470	5692	3040	5263	6355	11511	5780	10936
E-08	5197	7419	4767	6990	9924	15080	9350	14505

Ratio of Discrete to Continuous Time:

	β= 0.99				β= 0.995			
tol	**Linf-iter**	**Linf-truth**	**L2-iter**	**L2-truth**	**Linf-iter**	**Linf-truth**	**L2-iter**	**L2-truth**
E-01	1.76	0.40	2.00	0.50				
E-02	4.94	0.30	6.96	0.33				
E-03	0.82	0.26	2.19	0.27				
E-04	0.41	0.23	0.48	0.24				
E-06	0.28	0.21	0.29	0.21				
E-08	0.23	0.20	0.24	0.20				

criterion. The estimate of θ in (10) is nearly constant after the first several iterations. Combining the θ estimate in (10) with the error estimate in (11) did an excellent job in estimating the true error. Furthermore, equation (12) does an excellent job in predicting how many more iterations are necessary to achieve the error target $\bar{\varepsilon}$ once the changes in successive guesses fall below $\bar{\varepsilon}$. The results also indicate that the strategy of stopping when the iterates change by less than $\bar{\varepsilon}$ can produce very large errors. Users of these techniques should definitely use estimates of θ to justify their stopping criterion.

The L_2 criterion is satisfied sooner than the tougher L_∞ test. In some cases, the L_2 criterion is perhaps adequate. In particular, if one just cares about having an approximate solution which is good on average then L_2 is perhaps fine. Table 5 shows that the same adaptive approach will produce good results for the L_2 criterion.

Unfortunately, few authors inform the reader of their stopping rule. This is practice is comparable to empirical papers neglecting to inform the reader of confidence intervals, standard errors, posterior variance, or p values. Every numerical calculation has some error and the reader should be told what diagnostic is used in judging whether the error is acceptably small.

7 Conceptual Advantages of Continuous Time

We have emphasized the computational advantages of continuous-time models. There are also conceptual advantages of continuous-time specifications. We discuss some of these advantages in this section.

7.1 No Artificial Discreteness in Time

Discrete-time specifications often have difficulty modeling dynamic elements of a model and sometimes introduce artificial elements. Consider, for example, depreciation of equipment. In a deterministic model, it is common to specify either a constant rate of depreciation or specify a fixed life for a piece of capital. The later approach creates a vintage problem that greatly increases the size of the state space, so we typically want to assume a constant rate of depreciation, also called exponential depreciation. However, this is difficult to do in a finite-state model. Suppose we assume an integer number of machines and want to model an annual 20% depreciation rate of machinery. If we have five machines today, then that means we will have four next year. The case of, say, seven machines is not easy to model.

Since the state space is integers, the state cannot move to 5.6 units. One could assume that tomorrow's stock is either 5 or 6 and set the probabilities so that the expected depreciation equals the 20% rate. However, that implies that the variance of depreciation depends on the state, an odd assumption.

Another alternative is to assume that each machine has a 20% probability of dying and that machine death is independent across machines. This is a natural way to interpret 20% depreciation but it aggravates the curse of dimensionality in discrete-time models. Again, suppose that we have seven machines in the current period. Then the number of machines in the next period could be anything from zero to seven. If this is true of each of N firms, the integral in the discrete-time model is a sum over 8^N values, an even worse curse of dimensionality. In general, a discrete-time specification forces one to choose between odd assumptions about the nature of transitions like depreciation or tackle a very bad curse of dimensionality.

In continuous time, it is easy to model a constant rate of depreciation. We just say that the hazard rate of depreciation during a period of dt years is $0.20\ n\ dt$ if we have n machines. This exactly models a stochastic exponential depreciation rate. In our specific of hazard rates above, we see that the depreciation hazard rate is easily incorporated in the stochastic transition process. Our continuous-time model cannot not allow for a constant deterministic rate of depreciation, but any such model would require a continuum of allowable states and take us outside the more common framework of finite-state games.

7.2 More Realistic Strategic Reactions

The continuous-time approach also is more sensible from a dynamic strategic perspective. For example, suppose investment proceeds at a stochastic rate and two firms are both trying to expand capacity but both would want to cease this investment effort once the other succeeds. In discrete-time models, particularly if the period of time is large, there is some chance that both will succeed at the same time. This will lead to a condition of excessive investment and perhaps effort to disinvest. This will not happen in a continuous-time game since only one firm will succeed at any particular moment and the other could immediately cease his effort. There will be no "mistakes" in a continuous-time model. In many contexts, this specification is a more realistic description of players' abilities to react to a change in the situation.

7.3 Richer Range of Stochastic Outcomes

A continuous-time model allows for a richer range of stochastic outcomes over finite intervals of time. Many discrete-time models make the assumption that

$$\Pr^i\left(\omega^i - 1|\omega^i, x^i\right) + \Pr^i\left(\omega^i|\omega^i, x^i\right) + \Pr^i\left(\omega^i + 1|\omega^i, x^i\right) = 1$$

This translates into our continuous-time model as

$$f^i\left(\omega^i - 1|\omega^i, x^i\right) + f^i\left(\omega^i + 1|\omega^i, x^i\right) = 1$$

implying that all jumps are either one state up or one state down. This imposes a sense of continuity (you can't go from state 3 to state 5 without passing through state 4) to the problem even though it has a discrete state. This is a natural assumption to make in many models but it has different consequences for continuous and discrete-time models. In discrete-time model this implies that there will be at most a unit change in the state over one period. This means that the minimum amount of time to change the state by n steps is n periods. Discrete-time models have limited flexibility in modelling speed of change without stumbling on either the curse of dimensionality or problems associated with a β close to one.

In continuous-time models, this "continuity" assumption just says that the state changes one unit at a time but does not limit the distribution of jumps over any finite period of time. The speed of change is set by parameters other than the unit of time. Figure 1 illustrates the difference in value function for the discrete- and continuous-time versions of the same model. Note that they differ substantially because the problem is stuck at low values of the state variable for long periods of time.

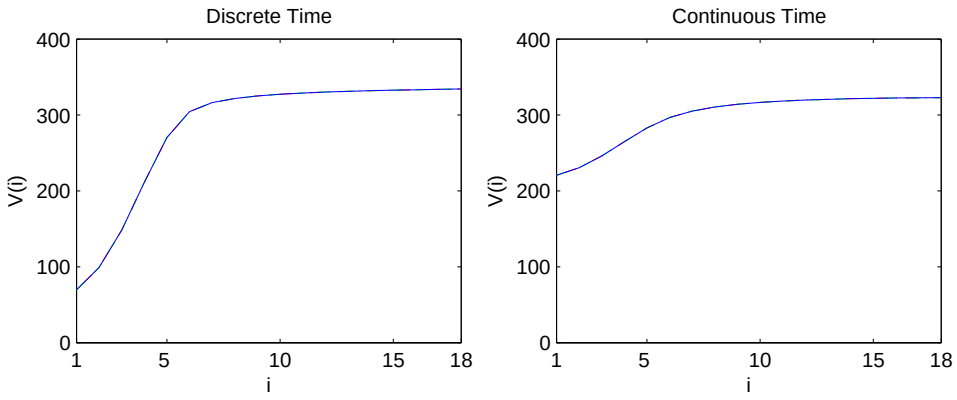

Figure 1: Richer range of stochastic outcomes.

7.4 Distinguish Between Discount Rate and Period Length

A continuous-time specification allows us to separate the discount rate from the period length. Discrete-time models choose a value of β, and that choice models both features of the game. In many cases, we would like to take β close to 1 to represent a short unit of time. However, algorithms such as Gauss-Jacobi and Gauss-Seidel often slow considerably as we increase β towards 1. In fact, value function iteration, which is a Gauss-Jacobi method, converges at exactly the rate β. The following table presents our experience with the number of iterations necessary as β increases. We see that taking β close to 1 is not a practical way to model a short unit of time. In contrast, our continuous time games converged much more rapidly for all but the case of a few firms even though the unit of time is essentially zero.

7.5 Functional Form Flexibility

Continuous-time specifications have more flexibility in specifying a model. For example, in discrete time the transition probabilities cannot exceed one. This forces one to look for tractable and interpretable functional forms. On popular choice is $p(x) = \frac{\alpha x}{1+\alpha x}$ which is highly stylized. In the continuous-time specification, one need only to impose nonnegative hazard rates; for example $h(x) = \alpha x^{\gamma}$ is a familiar constant elasticity functional form which can be used in continuous-time models but not in discrete-time models.

8 Generalizations

In this section we discuss some generalizations of our model and algorithm.

8.1 Multiple States per Firm

Most analyses assume just one state per firm. Suppose that there are N firms and the state of each firm is described by D state variables, each having M possible values. This is similar to assuming ND firms each described by one state variable with M possible values except that there is less symmetry - a firm is not indifferent among permutations among its own states nor among another firm's states.

8.2 Common Shocks

Common shocks can be easily accommodated by appropriate definition of the state. Since these shocks are common, they affect discrete- and continuous-time models in the same manner. [MORE TO COME]

8.3 Entry and Exit

Entry and exit can be easily accommodated by appropriate definition of the state and controls. These changes affect discrete- and continuous-time models in the same manner. [MORE OBVIOUS STUFF TO COME]

8.4 Stochastic Algorithm (PM2)

The curse of dimensionality in integration is recognized as an important problem in numerical analysis in general. PM2 uses a stochastic approximation algorithm to reduce the impact of dimensionality on computing equilibrium to stochastic games. PM2 uses three ideas. The first idea is to create approximations of the key integrals which are then updated using a small amount of computation each time a state is visited. The second idea is to visit states via simulations instead of some predetermined order, and focus on states that recur frequently under the equilibrium dynamics. The third idea is to keep track of only states that appear to be in the recurrent class under the equilibrium laws of motion, and not even store information about other states. The first and second ideas could be used jointly or separately in our model and may result in some speedup. The idea of focussing on the recurrent class may be useful but is very sensitive to the size of the recurrent set of states in the equilibrium. In many cases, all states are recurrent states; see, for example, Besanko and Doraszelski (2004) and Doraszelski and Markovitch (2003). Since we want to focus on generic cases, we visit states in a deterministic and nondiscriminatory fashion. However, to give the reader some basis for comparison, we note that PM2 reports that their procedure cuts running time down by about half in the case when there are six firms in their test model, and a reduction by a factor of at most 25 for the case of ten firms. Since this idea is orthogonal to our considerations, it is highly likely that a combination of this aspect of PM2 and our continuous-time model would result in a comparable speedup.

The third piece of PM2 would be useful if the state space is so large that storage considerations forces one to economize on states. However, our precomputed pointer scheme

makes it less likely that this consideration is important if one can store the whole game since the memory allocation and garbage collection steps involved in PM2 could become a serious drag on performance.

9 Conclusions

We have described a continuous-time model of dynamic games on finite states. This specification has considerable qualitative advantages as well as large computational advantages over discrete time. The advantages will be multiple orders of magnitude for games with more than a few states. This advantage will make it possible to examine far more complex and realistic models than currently feasible.

References

[1] Aguirregabiria, V. & Mira, P. (2002). Sequential simulation-based estimation of dynamic discrete games, Working paper, Boston University, Boston.

[2] Auerswald, P. (2001). The complexity of production, technological volatility and inter-industry differences in the persistence of profits above the norm, Working paper, Harvard University.

[3] Bajari, P., Benkard, L. & Levin, J. (2003). Estimating dynamic models of imperfect competition, Working paper, Stanford University, Stanford.

[4] Basar, T. & Olsder, J. (1999). *Dynamic Noncooperative Game Theory*, 2nd edn., Society for Industrial and Applied Mathematics, Philadelphia.

[5] Benkard, L. (2000). A dynamic analysis of the market for wide-bodied commercial aircraft, Working paper no. 7710, NBER, Cambridge.

[6] Besanko, D. and Doraszelski, U. (2004). Capacity dynamics and endogenous asymmetries in firm size, *Rand Journal of Economics* 35(1): 23–49.

[7] Besanko, D., Doraszelski, U., Kryukov, Y. & Satterthwaite, M. (2003). Learning-by-doing, organizational forgetting, and industry dynamics, Working paper, Northwestern University, Evanston.

[8] Caplin, A. & Nalebuff, B. (1991). Aggregation and imperfect competition: On the existence of equilibrium, *Econometrica* **59**(1): 26–59.

[9] Ching, A. (2002). A dynamic oligopoly structural model for the prescription drug markt after patent expiration, Working paper, Ohio State University, Columbus.

[10] Davis, S. & Haltiwanger, J. (1999). Gross job flows, in O. Ashenfelter & D. Card (eds), *Handbook of Labor Economics*, Vol. 3, Elsevier, Amsterdam, pp. 2711–2805.

[11] de Roos, N. (2004). A model of collusion timing, *International Journal of Industrial Organization* 22: 351–387.

[12] Doraszelski, U. (2003). An R&D race with knowledge accumulation, *Rand Journal of Economics* 34(1): 19–41.

[13] Doraszelski, U. & Satterthwaite, M. (2001). Foundations of Markov-perfect industry dynamics: Existence, multiplicity, and incomplete information, Working paper, Northwestern University, Evanston.

[14] Dunne, T., Roberts, M. & Samuelson, L. (1988). Patterns of firm entry and exit in U.S. manufacturing, *Rand Journal of Economics* 19(4): 495–515.

[15] Erdem, E. & Tybout, J. (2003). Trade policy and industrial sector reponses: Using evolutionary models to interpret the evidence, Working paper no. 9947, NBER, Cambridge.

[16] Ericson, R. & Pakes, A. (1995). Markov-perfect industry dynamics: A framework for empirical work, *Review of Economic Studies* 62: 53–82.

[17] Fershtman, C. & Pakes, A. (2000). A dynamic oligopoly with collusion and price wars, *Rand Journal of Economics* 31: 294–326.

[18] Filar, J. & Vrieze, K. (1997). *Competitive Markov decision processes*, Springer, New York.

[19] Goettler, R., Parlour, C. & Rajan, U. (2004). Equilibrium in a dynamic limit order market, Working paper, Carnegie Mellon University, Pittsburgh.

[20] Gowrisankaran, G. (1999). A dynamic model of endogenous horizontal mergers, *Rand Journal of Economics* 30(1): 56–83.

[21] Gowrisankaran, G. (1999). Efficient representation of state spaces for some dynamic models, *Journal of Economic Dynamics and Control* 23: 1077–1098.

[22] Judd, K. (1998). *Numerical Methods in Economics*, MIT Press, Cambridge.

[23] Kulkarni, V. (1995). *Modeling and Analysis of Stochastic Systems*, Chapman and Hall, London.

[24] Langohr, P. (2003). Competitive convergence and divergence: Capability and position dynamics, Working paper, Northwestern University, Evanston.

[25] Markovich, S. (1999). Snowball: The evolution of dynamic oligopolies with network externalities, Working paper, University of Chicago, Chicago.

[26] Pakes, A. (2000). A framework for applied dynamic analysis in I.O., Working paper no. *8024*, NBER, Cambridge.

[27] Pakes, A., Gowrisankaran, G. & McGuire, P. (1993). Implementing the Pakes-McGuire algorithm for computing Markov perfect equilibria in Gauss, Working paper, Yale University, New Haven.

[28] Pakes, A. & McGuire, P. (1994). Computing Markov-perfect Nash equilibria: Numerical implications of a dynamic differentiated product model, *Rand Journal of Economics* 25(4): 555–589.

[29] Pakes, A. & McGuire, P. (2001). Stochastic algorithms, symmetric Markov perfect equilibrium, and the "curse" of dimensionality, *Econometrica* 59(5): 1261–1281.

[30] Pakes, A., Ostrovsky, M. & Berry, S. (2003). Simple estimators for the parameters of discrete dynamic games (with entry/exit examples), Working paper, Harvard University, Cambridge.

[31] Pesendorfer, M. & Schmidt-Dengler, P. (2003). Identification and estimation of dynamic games, Working paper no. 9726, NBER, Cambridge.

[32] Song, M. (2002). Competition vs. cooperation: A dynamic model of investment in the semiconductor industry, Working paper, Harvard University, Cambridge.