OPERATIONS RESEARCH
Vol. 58, No. 4, Part 2 of 2, July–August 2010, pp. 1116–1132
ISSN 0030-364X | EISSN 1526-5463 | 10 | 5804 | 1116

DOI 10.1287/opre.1100.0843

A User's Guide to Solving Dynamic Stochastic Games Using the Homotopy Method

Ron N. Borkovsky
Rotman School of Management, University of Toronto, Toronto, Ontario M5S 3E6, Canada, ron.borkovsky@rotman.utoronto.ca

Ulrich Doraszelski
Department of Economics, Harvard University, Cambridge, Massachusetts 02138, doraszelski@harvard.edu

Yaroslav Kryukov
Tepper School of Business, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, kryukov@cmu.edu

This paper provides a step-by-step guide to solving dynamic stochastic games using the homotopy method. The homotopy method facilitates exploring the equilibrium correspondence in a systematic fashion; it is especially useful in games that have multiple equilibria. We discuss the theory of the homotopy method and its implementation and present two detailed examples of dynamic stochastic games that are solved using this method.

Subject classifications: homotopy method; dynamic stochastic games; Markov-perfect equilibrium; equilibrium correspondence; game theory; industrial organization; numerical methods.
Area of review: Special Issue on Computational Economics.
History: Received February 2008; revision received November 2009; accepted December 2009. Published online in *Articles in Advance* July 14, 2010.

1. Introduction

There has been much interest in game-theoretic models of industry evolution and, in particular, in the framework introduced by Ericson and Pakes (1995) that is at the heart of a large and growing literature in industrial organization and other fields (see Doraszelski and Pakes 2007 and the references therein). Ericson and Pakes (1995) provide a model of dynamic competition in an oligopolistic industry with investment, entry, and exit. Their framework is designed to facilitate numerical analysis of a wide variety of phenomena that are too complex to be explored in analytically tractable models. Methods for computing Markov-perfect equilibria are therefore a key part of this stream of research. This paper contributes by providing a step-by-step guide to solving dynamic stochastic games using the homotopy method.

Computing a Markov-perfect equilibrium of a dynamic stochastic game amounts to solving a large system of equations. To date, the Pakes and McGuire (1994) algorithm has been used most often to compute equilibria of dynamic oligopoly models. This backward solution method falls into the broader class of Gaussian methods. The idea behind Gaussian methods is that it is harder to solve a large system of equations once than to solve smaller systems many times, and that it may therefore be advantageous to break up a large system into small pieces.

The drawback of Gaussian methods is that they offer no systematic approach to computing multiple equilibria. The potential for multiplicity in the Ericson and Pakes (1995) framework is widely recognized; see p. 570 of Pakes and McGuire (1994) and, more recently, the examples of multiple equilibria in the online appendix to Doraszelski and Satterthwaite (2010) and in Besanko et al. (2010a). To identify more than one equilibrium (for a given parameterization of the model), the Pakes and McGuire (1994) algorithm must be restarted from different initial guesses. However, different initial guesses may or may not lead to different equilibria. A similar remark applies to the stochastic approximation algorithm of Pakes and McGuire (2001), the other widely used method for computing equilibria.

This, however, still understates the severity of the problem. When there are multiple equilibria, the trial-and-error approach of restarting the Pakes and McGuire (1994) algorithm from different initial guesses is sure to miss a substantial fraction of them, regardless of how many initial guesses are tried: As shown by Besanko et al. (2010a), if a dynamic stochastic game has multiple equilibria, then some of them cannot possibly be computed by the Pakes and McGuire (1994) algorithm. It is important, therefore, to consider alternative algorithms that can identify multiple equilibria and thus provide us with a more complete picture of the set of solutions to a dynamic stochastic game.

The homotopy method allows us to explore the equilibrium correspondence in a systematic fashion. The homotopy method is a type of path-following method. Starting from a single equilibrium that has already been computed for a given parameterization of the model, the homotopy algorithm traces out an entire path of equilibria by varying one or more parameters of the model. Whenever we can find such a path and multiple equilibria are the result of

the path bending back on itself, then the homotopy method is guaranteed to identify them. We note at the outset that it is not assured that any given path includes all possible equilibria at a given value of the parameter vector.

In this paper we discuss the theory of the homotopy method as well as HOMPACK90, a suite of Fortran90 routines developed by Watson et al. (1997) that implements this method. To explore the equilibrium correspondence of a dynamic stochastic game, we set up the homotopy algorithm to vary one or more parameters of the model. This type of application is referred to as a natural-parameter homotopy. We discuss potential problems that one may encounter in using HOMPACK90 in this way and offer some guidance as to how to resolve them.

We present two examples of dynamic stochastic games and show, step by step, how to solve them using the homotopy method. In order to use the homotopy method, one must formulate a problem as a system of equations. Our first example, the learning-by-doing model of Besanko et al. (2010a), is particularly well suited for the homotopy method because it is straightforward to express the equilibrium conditions as a system of equations. We discuss in detail how this is done. Moreover, we illustrate the computational demands of the homotopy method using the learning-by-doing model as an example.

Our second example, the quality ladder model of Pakes and McGuire (1994), presents a complication. As investment cannot be negative, the problem that a firm has to solve is formulated using a complementary slackness condition, a combination of equalities and inequalities. We show how to reformulate this complementary slackness condition as a system of equations that is amenable to the homotopy method.

In sum, this paper provides a step-by-step guide to solving dynamic stochastic games using the homotopy method. Our goal here is to enable the reader to start using HOMPACK90 as quickly as possible. To this end, we also make the code for the learning-by-doing and quality ladder models available on our homepages. The code is accompanied by additional detailed instructions on how to set up and use it.

2. The Theory of the Homotopy Method

A Markov-perfect equilibrium of a dynamic stochastic game consists of values, i.e., expected net present values of per-period payoffs, and policies, i.e., strategies, for each player in each state. Values are typically characterized by Bellman equations and policies by optimality conditions (e.g., first-order conditions). These equilibrium conditions depend on the parameterization of the model. Collecting Bellman equations and optimality conditions for each player in each state, the equilibrium conditions amount to a system of equations of the form

$$\mathbf{H}(\mathbf{x}, \lambda) = \mathbf{0}, \tag{1}$$

where $\mathbf{H}$: $\mathbb{R}^{N+1} \to \mathbb{R}^N$, $\mathbf{x} \in \mathbb{R}^N$ is the vector of unknown values and policies, and $\mathbf{0} \in \mathbb{R}^N$ is a vector of zeros. $\lambda \in [0, 1]$ is the so-called homotopy parameter.[1] Depending on the application at hand, the homotopy parameter maps into one or more of the parameters of the model. The object of interest is the equilibrium correspondence

$$\mathbf{H}^{-1} = \{(\mathbf{x}, \lambda) \mid \mathbf{H}(\mathbf{x}, \lambda) = \mathbf{0}\}.$$

The homotopy method aims to trace out entire paths of equilibria in $\mathbf{H}^{-1}$ by varying both $\mathbf{x}$ and λ.

To this end, we define a parametric path as a set of functions $(\mathbf{x}(s), \lambda(s))$ such that $(\mathbf{x}(s), \lambda(s)) \in \mathbf{H}^{-1}$. The points on the path are indexed by the auxiliary variable s; as we move along the path, s either increases or decreases monotonically; however, neither $\mathbf{x}$ nor λ necessarily vary monotonically. Differentiating $\mathbf{H}(\mathbf{x}(s), \lambda(s)) = \mathbf{0}$ with respect to s yields

$$\frac{\partial \mathbf{H}(\mathbf{x}(s), \lambda(s))}{\partial \mathbf{x}} \mathbf{x}'(s) + \frac{\partial \mathbf{H}(\mathbf{x}(s), \lambda(s))}{\partial \lambda} \lambda'(s) = \mathbf{0}, \tag{2}$$

where $\partial \mathbf{H}(\mathbf{x}(s), \lambda(s))/\partial \mathbf{x}$ is the $(N \times N)$ Jacobian of $\mathbf{H}$ with respect to $\mathbf{x}$, $\mathbf{x}'(s)$ and $\partial \mathbf{H}(\mathbf{x}(s), \lambda(s))/\partial \lambda$ are $(N \times 1)$ vectors, and $\lambda'(s)$ is a scalar. This system of equations captures the conditions that must be satisfied in order to remain "on path." As this is a system of N differential equations in $N + 1$ unknowns, $x_i'(s)$, $i = 1, \ldots, N$, and $\lambda'(s)$, it has many solutions. Although these solutions differ by monotone transformations of s, they all describe the same path. One solution obeys the so-called basic differential equations

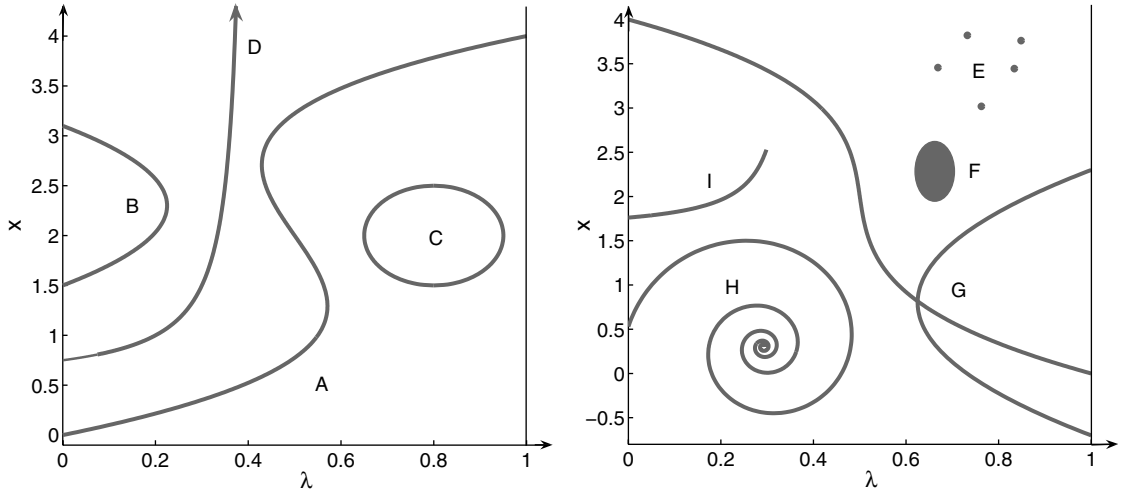
$$y_i'(s) = (-1)^{i+1} \det\left(\left[\frac{\partial \mathbf{H}(\mathbf{y}(s))}{\partial \mathbf{y}}\right]_{-i}\right), \quad i = 1, \ldots, N+1, \tag{3}$$

where $\mathbf{y}(s) = (\mathbf{x}(s), \lambda(s))$, and the notation $[\cdot]_{-i}$ is used to indicate that the ith column is removed from the $(N \times (N+1))$ Jacobian $\partial \mathbf{H}(\mathbf{y}(s))/\partial \mathbf{y}$ of $\mathbf{H}$ with respect to $\mathbf{y}$. A proof that the basic differential equations (3) satisfy the conditions (2) for remaining on path can be found in Garcia and Zangwill (1979) and on pp. 27–28 of Zangwill and Garcia (1981).

An implementation of the homotopy method—a homotopy algorithm—can be used to trace out entire paths of equilibria by numerically solving the basic differential equations (3). An already computed equilibrium provides the initial condition. From there a homotopy algorithm uses the basic differential equations to determine the next step along the path. In this manner, it continues to follow the path step by step.

Regularity and Smoothness Requirements. A closer inspection of the basic differential equations (3) reveals a potential difficulty. If the Jacobian $\partial \mathbf{H}(\mathbf{y}(s))/\partial \mathbf{y}$ is not of full rank at some point $\mathbf{y}(s)$ on the solution path, then the

Figure 1. Examples of solution paths if **H** is regular (left panel) and not regular (right panel).



determinant of each of its square submatrices is zero. Thus, according to the basic differential equations (3), $y_i'(s) = 0$, $i = 1, \ldots, N + 1$, and the homotopy method is stuck at point $\mathbf{y}(s)$. A central condition in the mathematical literature on the homotopy method is thus that the Jacobian must have full rank at all points on the solution path. If so, the homotopy is called regular. More formally, **H** is regular if $\operatorname{rank}(\partial \mathbf{H}(\mathbf{y})/\partial \mathbf{y}) = N$ for all $\mathbf{y} \in \mathbf{H}^{-1}$. The regularity requirement—and a certain smoothness requirement to be discussed below—ensures that the set of solutions $\mathbf{H}^{-1}$ consists only of continuous paths. The left panel of Figure 1 shows examples of possible solution paths if **H** is regular: (A) paths that start at $\lambda = 0$ and end at $\lambda = 1$, (B) paths that start and end at $\lambda = 0$ or $\lambda = 1$, (C) loops, and (D) paths that start at $\lambda = 0$ or $\lambda = 1$ but never end because x (or a component of **x** in the case of a vector) tends to $+\infty$ or $-\infty$.[2] The right panel of Figure 1 shows examples of solution paths that are ruled out by the regularity requirement: (E) isolated equilibria, (F) continua of equilibria, (G) branching points,[3] (H) paths of infinite length that start at $\lambda = 0$ or $\lambda = 1$ and converge to single points (spirals), and (I) paths that start at $\lambda = 0$ or $\lambda = 1$ but suddenly terminate.

In practice, it is often hard to establish regularity because the Jacobian of a system of equations that characterizes the equilibria of a dynamic stochastic game formulated in the Ericson and Pakes (1995) framework tends to be intractable. This stems partly from the fact that the Jacobian for such a system is typically quite large because the system includes at least two equations (Bellman equation and optimality condition) for each state of the industry, and even "small" models with few firms and few states per firm tend to have hundreds of industry states.

The other major requirement of the homotopy method is smoothness in the sense of differentiability. This yields solution paths that are smooth and free of sudden turns or kinks. Formally, if **H** is continuously differentiable in addition to regular, then the set of solutions $\mathbf{H}^{-1}$ consists only of continuously differentiable paths. This result is known as the path theorem and essentially follows from the implicit function theorem (see, e.g., p. 20 of Zangwill and Garcia 1981). Moreover, for a path to be described by the basic differential equations (3), it must be the case that **H** is twice continuously differentiable in addition to regular. This result is known as the BDE theorem (see, e.g., pp. 27–28 of Zangwill and Garcia 1981).

The smoothness requirement is nontrivial and easily violated, for example, by nonnegativity constraints on components of **x**, say because investment cannot become negative, or by distributions with nondifferentiable cumulative distribution functions such as the uniform distribution that is often used to model random scrap values and setup costs. Section 5 explains how to deal with such complications.

There is a subtle difference between the homotopy method, a mathematical theory, and the homotopy algorithm, a numerical method. In theory, the homotopy method is used to describe solution paths. In practice, a homotopy algorithm takes discrete steps along such a path. This can be beneficial because the homotopy algorithm may succeed in tracing out a solution path even if the regularity and/or smoothness requirements are violated; as the homotopy algorithm proceeds along the solution path in discrete steps, it may skip over points at which one or both of these requirements are violated. However, this also can lead to a complication. As the homotopy algorithm proceeds in discrete steps, it may jump from one solution path to another, thus failing to trace out either path in its entirety. These issues are discussed further in §5.5.

3. The HOMPACK90 Software Package

HOMPACK90 is as a suite of Fortran90 routines that traces out a path in

$$\mathbf{H}^{-1} = \{\mathbf{y} \mid \mathbf{H}(\mathbf{y}) = 0\}.$$

Table 1. Path-following algorithms and dense vs. sparse Jacobian in HOMPACK90.

	Dense Jacobian	Sparse Jacobian
ODE-based	FIXPDF	FIXPDS
Normal flow	FIXPNF	FIXPNS
Augmented Jacobian	FIXPQF	FIXPQS

The notation $\mathbf{y} = (\mathbf{x}, \lambda) \in \mathbb{R}^{N+1}$ underlines that the homotopy method does not make a distinction between the unknown variables $\mathbf{x} \in \mathbb{R}^N$ and the homotopy parameter $\lambda \in [0, 1]$. Here we just give a brief overview that is meant to enable the reader to start using HOMPACK90 as quickly as possible. A detailed description of HOMPACK90 is given in Watson et al. (1987, 1997).

To use HOMPACK90, the user must provide Fortran90 code for the system of equations and its Jacobian. In addition, the user must supply HOMPACK90 with an initial condition in the form of a solution to the system of equations for a particular parameterization. HOMPACK90 then traces out a solution path. HOMPACK90 offers several different path-following algorithms as well as storage formats for the Jacobian of the system of equations. Table 1 gives an overview. Below we proceed to discuss the differences between the various path-following algorithms and storage formats as well as ways to generate initial conditions. In §4.3 we then compare the implications of the various path-following algorithms and storage formats for the performance of HOMPACK90.

The output of HOMPACK90 includes a sequence of solutions to the system of equations, saved to binary files,[4] and an exit flag that indicates a normal ending or several kinds of failure. We discuss some of the potential problems in §5.5.

3.1. Path-Following Algorithms

HOMPACK90 traces out a parametric path $\mathbf{y}(s) \in \mathbf{H}^{-1}$ as a sequence of points, indexed by k. The kth point in the sequence is $\{s^k, \mathbf{y}^k\}$, where $\mathbf{y}^k$ is understood to represent $\mathbf{y}(s^k)$. The step along the path from one point to the next starts by choosing $\Delta s = s^{k+1} - s^k$. HOMPACK90 adjusts this step length based on the curvature of the path. Then HOMPACK90 computes the next point $\mathbf{y}^{k+1}$ using a two-phase method. The predictor phase generates a guess for $\mathbf{y}^{k+1}$; the corrector phase then improves this guess using a version of Newton's method. The difference between the various available path-following algorithms lies in the implementation of the predictor and corrector phases.

ODE-Based. The predictor phase directly applies the system of differential equations (2). It first solves the system of linear equations

$$\frac{\partial \mathbf{H}(\mathbf{y}^k)}{\partial \mathbf{y}} \Delta \mathbf{y} = 0 \tag{4}$$

to obtain $\Delta \mathbf{y}$ and then computes the guess for the next point as $\mathbf{y}^{k+1} = \mathbf{y}^k + \Delta \mathbf{y} \Delta s$. As this predictor step tends to be very precise, typically several such steps are executed before a corrector step becomes necessary.

Normal Flow. The predictor phase uses a Hermite cubic extrapolation from the previous two points as a guess for the next point. Although the Hermite cubic extrapolation is much easier to compute than solving the system of linear equations (4), it is also much cruder. The corrector phase is thus necessary at every step.

Augmented Jacobian. The predictor phase is the same as in the normal flow algorithm, but the augmented Jacobian algorithm takes a different approach to the corrector phase (quasi-Newton instead of Newton steps).

3.2. Jacobian

HOMPACK90 requires the user to provide a routine that returns the Jacobian $\partial \mathbf{H}(\mathbf{y})/\partial \mathbf{y}$ at a given point $\mathbf{y}$. Due to the key role of the Jacobian in the homotopy algorithm, we next discuss some details on ways to compute and represent it.

Numerical vs. Analytical Jacobian. The easiest way to compute the Jacobian is to do so numerically using a one- or two-sided finite-difference scheme (see, e.g., Chapter 7 of Judd 1998). However, we found that, due to the limited precision of numerical differentiation, the ODE-based algorithm takes small steps, and this significantly increases the time needed to traverse an entire path; normal flow and augmented Jacobian algorithms are more robust to imprecise Jacobians.

The obvious solution is to use analytical instead of numerical differentiation, but this carries a high fixed cost of deriving, coding, and debugging the Jacobian. Instead, we turn to automatic differentiation, a technique for generating computer programs with statements for the computation of derivatives based on the chain rule of differential calculus. Automatic differentiation relies on the fact that every function, no matter how complicated, is executed on a computer as a sequence of elementary operations (addition, multiplication, etc.) and functions (exp, sin, etc.). By applying the chain rule over and over again to the composition of those elementary operations, one can compute, in a completely mechanical fashion, a derivative of the function.

We use ADIFOR to automatically differentiate Fortran code. ADIFOR is described in Bischof et al. (1996); here we just give a brief overview.[5] The input to ADIFOR is the Fortran90 code that returns $\mathbf{H}(\mathbf{y})$ at a given point $\mathbf{y}$. ADIFOR analyzes the code and from it generates new code. This code receives a pair of $((N+1) \times 1)$ vectors $(\mathbf{y}, \Delta \mathbf{y})$ and returns the $(N \times 1)$ vector

$$\Delta \mathbf{H} = \frac{\partial \mathbf{H}(\mathbf{y})}{\partial \mathbf{y}} \Delta \mathbf{y}.$$

Thus, we obtain the jth column of the Jacobian via a single call to the ADIFOR-generated code with $\Delta \mathbf{y}$ set to the jth basis vector. Repeating this for $j = 1, \ldots, N+1$ we assemble the entire Jacobian.

This column-by-column approach is necessary because the ADIFOR-generated code never computes the elements $\partial H_i(\mathbf{y})/\partial y_j$ of the Jacobian $\partial \mathbf{H}(\mathbf{y})/\partial \mathbf{y}$ explicitly, but instead transforms $(\mathbf{y}, \Delta\mathbf{y})$ into $\Delta\mathbf{H}$ in a process that parallels the computation of $\mathbf{H}(\mathbf{y})$. For example, if $H(y_1, y_2) = u(y_1)v(y_2)$ is a function of the scalars y_1 and y_2, where u and v are themselves functions, then the ADIFOR-generated code computes

$$\Delta u := u'(y_1)\Delta y_1,$$

$$\Delta v := v'(y_2)\Delta y_2,$$

$$\Delta H := u(y_1)\Delta v + v(y_2)\Delta u.$$

Dense vs. Sparse Jacobian. In many applications, including dynamic stochastic games, the number of equations and unknowns is large, but any given equation involves only a small number of unknowns (because the transitions from one state to the next are typically restricted to a small set of "nearby" states), leading to a Jacobian comprising mostly zeros. Such a Jacobian is called sparse and can be more efficiently represented using a sparse matrix storage format that consists of a list of the nonzero elements with corresponding row and column indices rather than as a dense matrix that consists of the entire set of $N(N+1)$ elements.

Taking advantage of the sparse nature of the Jacobians in dynamic stochastic games offers a decrease in computation time, and in fact we show in §4.3 that this decrease is substantial. The additional efficiency comes from lower memory requirements and faster linear algebra operations. In addition, the Jacobian of a very large system of equations may exceed the available memory unless it is stored as a sparse matrix.

The use of sparse Jacobians is complicated by two issues. First, there is additional coding because the user must specify the "sparsity structure," i.e., the row and column indices of potentially nonzero elements. In practice, this means going through the system of equations and identifying the elements of $\mathbf{y}$ that appear in a given equation.

Second, the sparse and dense versions of the various path-following algorithms take different approaches to solving systems of linear equations. In all cases, the linear algebra routines in HOMPACK90 were selected for speed, not reliability, which means that they can and do fail for certain problems. Our experience has been that the sparse linear solver is more likely to fail than the dense linear solver; §5.6 provides further detail and offers some solutions.

3.3. Initial Condition

The final input to HOMPACK90 is an initial condition in the form of a solution to the system of equations for the particular parameterization associated with $\lambda = 0$. In some cases, if the parameterization associated with $\lambda = 0$ is trivial, the solution can be derived analytically. A good example is the case of a zero discount rate that turns a dynamic stochastic game into a set of disjoint static games played out in every state. Another example is a particular parameterization that makes movements through state space unidirectional and thus allows the game to be solved by backwards induction (see, e.g., Judd et al. 2007).

More generally, a solution for a particular parameterization can be computed numerically using a number of approaches such as Gaussian methods including (but not limited to) the Pakes and McGuire (1994) algorithm, other nonlinear solvers (see Ferris et al. 2007), and artificial homotopies (see Zangwill and Garcia 1981, Chapter 1), which can also be implemented using HOMPACK90. Finally, one can use a solution obtained by tracing out a path along a different parameter as an initial condition (see §5.1 for an example of path following along several parameters).

4. Example 1: The Learning-by-Doing Model

We begin with the learning-by-doing model of Besanko et al. (2010a). The description of the model is abridged. Please refer to Besanko et al. (2010a) for economic motivation and greater detail on some derivations.

4.1. Model

Firms and States. We consider a discrete-time, infinite-horizon stochastic game. Firm $n \in \{1, 2\}$ is described by its stock of know-how (or experience) $e_n \in \{1, \ldots, M\}$. At any point in time, the industry is completely characterized by a vector of firms' states $\mathbf{e} = (e_1, e_2) \in \{1, \ldots, M\}^2$. We refer to $\mathbf{e}$ as the state of the industry. We use $\mathbf{e}^{[2]}$ to denote the vector (e_2, e_1) found by interchanging the stocks of know-how of firms 1 and 2.

Each period, firms observe the state of the industry and set prices for their respective goods. By making a sale, a firm can add to its stock of know-how. At the same time, the firm faces the possibility of organizational forgetting, leading to the law of motion $e_n' = e_n + q_n - f_n$, where e_n' and e_n are firm n's stock of know-how in the subsequent and current period, respectively, the random variable $q_n \in \{0, 1\}$ indicates whether firm n makes a sale, and the random variable $f_n \in \{0, 1\}$ represents organizational forgetting. If $q_n = 1$, the firm gains a unit of know-how through learning by doing, whereas it loses a unit of know-how through organizational forgetting if $f_n = 1$.

Learning by Doing. Firm n's marginal cost of production $c(e_n)$ depends on its stock of know-how e_n through a learning curve with a progress ratio of $\rho \in (0, 1]$:

$$c(e_n) = \begin{cases} \kappa e_n^{\eta} & \text{if } 1 \leqslant e_n < m, \\ \kappa m^{\eta} & \text{if } m \leqslant e_n \leqslant M, \end{cases}$$

where $\eta = \log_2 \rho$. Marginal cost decreases by $100(1-\rho)$ percent as the stock of know-how doubles, so that a

lower progress ratio implies a steeper learning curve. The marginal cost of production at the top of the learning curve, $c(1)$, is $\kappa > 0$, and m represents the stock of know-how at which a firm reaches the bottom of its learning curve.

Organizational Forgetting. The probability that firm n loses a unit of know-how through organizational forgetting is

$$\Delta(e_n) = \Pr(f_n = 1) = 1 - (1-\delta)^{e_n},$$

where $\delta \in [0, 1]$ is the forgetting rate.

Demand. Each period, one (nonstrategic) buyer enters the market and purchases a unit of the good from one of the two firms. The net utility of good n to a buyer is $v - p_n + \varepsilon_n$, where p_n is the price, v is the fixed component of utility, and ε_n is a stochastic component that captures the buyer's idiosyncratic preference for good n. The buyer's idiosyncratic preferences $(\varepsilon_1, \varepsilon_2)$ are unobservable to firms and are independently and identically type 1 extreme value distributed. The buyer purchases the good that gives it the highest net utility, so the probability that firm n makes a sale is given by

$$D_n(\mathbf{p}) = \Pr(q_n = 1) = \frac{\exp(v - p_n)}{\sum_{k=1}^{2} \exp(v - p_k)} = \frac{1}{1 + \exp(p_n - p_{-n})},$$

where $\mathbf{p} = (p_1, p_2)$ is the vector of prices and we adopt the convention of using p_{-n} to denote the price charged by the other firm.

State-to-State Transitions. From one period to the next, the transition probabilities are

$$\Pr(e_n' \mid e_n, q_n) = \begin{cases} 1 - \Delta(e_n) & \text{if } e_n' = e_n + q_n, \\ \Delta(e_n) & \text{if } e_n' = e_n + q_n - 1, \end{cases}$$

where, at the upper and lower boundaries of the state space, we modify the transition probabilities to be $\Pr(M \mid M, 1) = 1$ and $\Pr(1 \mid 1, 0) = 1$, respectively.

Bellman Equation and First-Order Condition. Define $V_n(\mathbf{e})$ to be the expected net present value of firm n's cash flows if the industry is currently in state $\mathbf{e}$. The value function $\mathbf{V}_n$: $\{1, \ldots, M\}^2 \to \mathbb{R}$ is implicitly defined by the Bellman equation

$$V_n(\mathbf{e}) = \max_{p_n} D_n(p_n, p_{-n}(\mathbf{e}))(p_n - c(e_n)) + \beta \sum_{k=1}^{2} D_k(p_n, p_{-n}(\mathbf{e}))\overline{V}_{nk}(\mathbf{e}), \quad (5)$$

where $p_{-n}(\mathbf{e})$ is the price charged by the other firm in state $\mathbf{e}$, $\beta \in (0, 1)$ is the discount factor, and $\overline{V}_{nk}(\mathbf{e})$ is the expectation of firm n's value function conditional on the buyer purchasing the good from firm $k \in \{1, 2\}$ in state $\mathbf{e}$ as given by

$$\overline{V}_{n1}(\mathbf{e}) = \sum_{e_1'=e_1}^{e_1+1} \sum_{e_2'=e_2-1}^{e_2} V_n(\mathbf{e}') \Pr(e_1' \mid e_1, 1) \Pr(e_2' \mid e_2, 0), \quad (6)$$

$$\overline{V}_{n2}(\mathbf{e}) = \sum_{e_1'=e_1-1}^{e_1} \sum_{e_2'=e_2}^{e_2+1} V_n(\mathbf{e}') \Pr(e_1' \mid e_1, 0) \Pr(e_2' \mid e_2, 1). \quad (7)$$

The policy function $\mathbf{p}_n$: $\{1, \ldots, M\}^2 \to \mathbb{R}$ specifies the price $p_n(\mathbf{e})$ that firm n sets in state $\mathbf{e}$. To determine it, let $h_n(\cdot)$ be the maximand in the Bellman equation (5). Differentiating $h_n(\cdot)$ with respect to p_n we obtain the first-order condition

$$0 = D_n(p_n, p_{-n}(\mathbf{e}))(1 - (p_n - c(e_n)) - \beta \overline{V}_{nn}(\mathbf{e}) + h_n(\cdot)).$$

It is straightforward to show that the pricing decision $p_n(\mathbf{e})$ is uniquely determined by the solution to the first-order condition.

Equilibrium. We restrict attention to symmetric Markov-perfect equilibria. In a symmetric equilibrium the pricing decision taken by firm 2 in state $\mathbf{e}$ is identical to the pricing decision taken by firm 1 in state $\mathbf{e}^{[2]}$, i.e., $p_2(\mathbf{e}) = p_1(\mathbf{e}^{[2]})$, and similarly for the value function. It therefore suffices to determine the value and policy functions of firm 1, and we define $V(\mathbf{e}) = V_1(\mathbf{e})$ and $p(\mathbf{e}) = p_1(\mathbf{e})$ for each state $\mathbf{e}$. To simplify the notation, we further define $\overline{V}_k(\mathbf{e}) = \overline{V}_{1k}(\mathbf{e})$ to be the conditional expectation of firm 1's value function and $D_k(\mathbf{e}) = D_k(p(\mathbf{e}), p(\mathbf{e}^{[2]}))$ to be the probability that the buyer purchases from firm $k \in \{1, 2\}$ in the state $\mathbf{e}$.

Parameterization. We focus on the ways in which learning by doing and organizational forgetting affect pricing behavior, and the industry dynamics implied by that behavior. Besanko et al. (2010a) prove that the model has a unique equilibrium if $\delta = 0$ or $\delta = 1$. It is therefore natural to use the homotopy method to trace out the equilibrium correspondence by varying δ from 0 to 1. We thus make the forgetting rate δ a function of the homotopy parameter λ and set

$$\delta(\lambda) = \delta^{start} + \lambda(\delta^{end} - \delta^{start}).$$

In particular, if $\delta^{start} = 0$ and $\delta^{end} = 1$, then the homotopy method traces out the equilibrium correspondence from $\delta(0) = 0$ to $\delta(1) = 1$. To explore the role of learning by doing, we repeat this procedure for 100 evenly spaced values of $\rho \in [0.01, 1]$. We hold the remaining parameters fixed at the values shown in Table 2.

System of Equations. We are now ready to describe the equilibrium as a system of equations in the form given by (1). Define the vector of unknown values and policies in equilibrium as

$$\mathbf{x} = [V(1,1), V(2,1), \ldots, V(M,1), V(1,2), \ldots, V(M,M), p(1,1), \ldots, p(M,M)]'.$$

Table 2. Parameter values.

Parameter	M	m	β
Value	30	15	$\frac{1}{1.05} = 0.9524$

Note. Learning-by-doing model.

The Bellman equation and first-order condition in state $\mathbf{e}$ are

$$H^1_{\mathbf{e}}(\mathbf{x},\lambda) = -V(\mathbf{e}) + D_1(\mathbf{e})(p(\mathbf{e}) - c(e_1)) + \beta \sum_{k=1}^{2} D_k(\mathbf{e})\overline{V}_k(\mathbf{e}) = 0, \tag{8}$$

$$H^2_{\mathbf{e}}(\mathbf{x},\lambda) = 1 - (1 - D_1(\mathbf{e}))(p(\mathbf{e}) - c(e_1)) - \beta\overline{V}_1(\mathbf{e}) + \beta \sum_{k=1}^{2} D_k(\mathbf{e})\overline{V}_k(\mathbf{e}) = 0, \tag{9}$$

where we substitute for $\overline{V}_1(\mathbf{e})$ and $\overline{V}_2(\mathbf{e})$ using definitions (6) and (7) (imposing symmetry). The collection of Equations (8) and (9) for all states $\mathbf{e} \in \{1, \ldots, M\}^2$ can be written more compactly as

$$\mathbf{H}(\mathbf{x},\lambda) = \begin{bmatrix} H^1_{(1,1)}(\mathbf{x},\lambda) \\ H^1_{(2,1)}(\mathbf{x},\lambda) \\ \vdots \\ H^2_{(M,M)}(\mathbf{x},\lambda) \end{bmatrix} = \mathbf{0}, \tag{10}$$

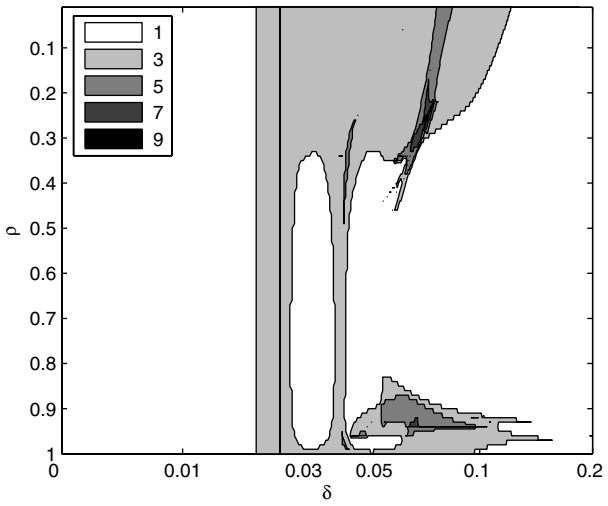
where $\mathbf{0} \in \mathbb{R}^{2M^2}$ is a vector of zeros. Any solution to this system of $2M^2$ equations in $2M^2$ unknowns $\mathbf{x} \in \mathbb{R}^{2M^2}$ is a symmetric equilibrium in pure strategies (for a given value of $\lambda \in [0, 1]$).[6]

4.2. Equilibrium Correspondence

Figure 2 shows the number of equilibria as a function of the progress ratio ρ and the forgetting rate δ. Darker shades indicate more equilibria. The subset of parameterizations that yield three equilibria is fairly large, and we have found up to nine equilibria for some values of ρ and δ. The multiple equilibria describe a rich array of pricing behaviors that are economically meaningful and that are quite different in terms of implied industry structure and dynamics (see Besanko et al. 2010a for a discussion).

Recall that we made the forgetting rate δ a function of the homotopy parameter λ. To visualize the set of equilibria as a correspondence of δ for a specific value of ρ, we need a way to summarize each equilibrium as a single number. The value function in the initial state $(1, 1)$ is the value of a firm at the onset of the industry; $V(1, 1)$ is thus an economically meaningful summary of an equilibrium. Because we are also interested in long-run industry concentration, we further compute the expected Herfindahl

Figure 2. Number of equilibria.



Note. Learning-by-doing model.

index. We proceed in two steps. First, we use the policy function to construct the probability distribution over the next period's state $\mathbf{e}'$ given this period's state $\mathbf{e}$, and from it we compute the limiting (or ergodic) distribution over states, μ^∞. Second, we use this distribution to compute the expected Herfindahl index

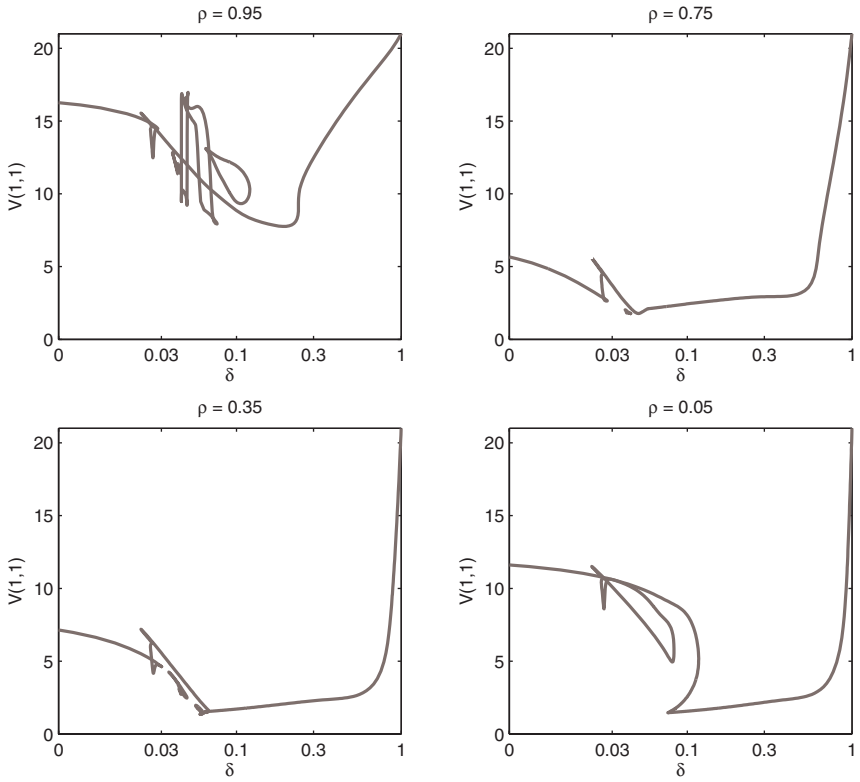
$$HHI^\infty = \sum_{\mathbf{e} \in \{1, \ldots, M\}^2} [(D_1(\mathbf{e}))^2 + (D_2(\mathbf{e}))^2]\mu^\infty(\mathbf{e}).$$

Asymmetric industry structures arise and persist to the extent that $HHI^\infty > 0.5$.

Figures 3 and 4 visualize the equilibrium correspondence in terms of $V(1, 1)$ and HHI^∞, respectively, for a variety of different progress ratios ρ. If the system of equations that characterizes the equilibria is regular, then there must be a path connecting the unique equilibria at $\delta = 0$ and $\delta = 1$. Multiple equilibria arise whenever this main path bends back on itself. Similar to (A) in the left panel of Figure 1, in the bottom-right panel of Figure 4, the path from $\delta = 0$ to $\delta = 1$ is S-shaped around $\delta = 0.1$; when the sign of $\lambda'(s)$ changes from positive to negative at $\delta = 0.1181$, the path is bending backward, and when the sign of $\lambda'(s)$ changes from negative to positive at $\delta = 0.0745$, the path is bending forward.[7] Consequently, Figure 2 indicates that there are three equilibria in the vicinity of $\delta = 0.1$ and $\rho = 0.05$.

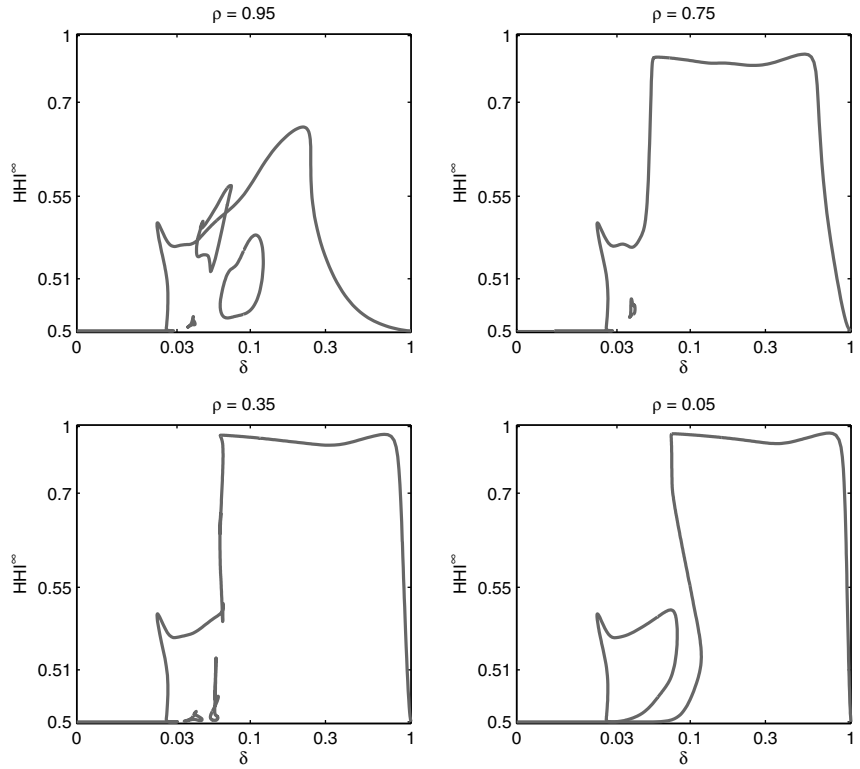
We have also been able to identify one or more loops, similar to (C) in the left panel of Figure 1, that are disjoint from the main path. These loops further add to the multiplicity of equilibria. In the upper-left panel of Figure 4, for example, there is a loop around $\delta = 0.1$; accordingly, Figure 2 indicates that there are three equilibria in the vicinity of $\delta = 0.1$ and $\rho = 0.95$. However, still other loops may exist because, in order to trace out a loop, we must first somehow compute at least one equilibrium on the loop. Unfortunately, there is no way to determine whether one's search for loops—or more generally equilibria—has

Figure 3. Initial firm value $V(1,1)$.



Note. Learning-by-doing model.

Figure 4. Limiting expected Herfindahl index HHI^{∞}.



Note. Learning-by-doing model.

been exhaustive. Figures 2–4 are therefore not necessarily a complete mapping of the equilibria.

In some places, the various paths appear to intersect one another. A case in point is the loop in the upper-left panel of Figure 4 that appears to twice intersect the path from $\delta = 0$ to $\delta = 1$. Although such an intersection seems to resemble the branching point (G) in the right panel of Figure 1, the fact that two equilibria give rise to the same expected Herfindahl index does not mean that the equilibria themselves are the same. We have indeed verified that the various solution paths do not intersect. Thus, the intersections in Figures 3 and 4 do not stem from violations of the regularity requirement.

4.3. Performance

HOMPACK90 offers several different path-following algorithms and storage formats for the Jacobian of the system of equations. Moreover, the user can compute the Jacobian either numerically or analytically. To highlight the implications of these choices for the performance of HOMPACK90, we have conducted a series of experiments. Throughout, we focus on the main path of the equilibrium correspondence from $\delta = 0$ to $\delta = 1$ for a progress ratio of $\rho = 0.75$ (as shown in the upper-right panel of Figure 4). We set the precision in HOMPACK90 to 10^{-10} (see §5.5 for a discussion). We use ADIFOR to analytically compute the Jacobian. All experiments are conducted on a Linux machine with a 64-bit 1 GHz AMD Athlon CPU and 4 GB of RAM.

Path-Following Algorithms. A major issue is the trade-off between robustness and computation time. Computation time is the product of the number of steps it takes to trace out the entire path and the average time per step. This involves yet another trade-off because these two determinants of computation time are affected in opposite ways by the size of the step that the homotopy algorithm takes from one point to the next. Optimally adjusting the step size is a highly nontrivial problem, and the algorithm that does this is a major part of HOMPACK90.

Turning to the choice of a specific path-following algorithm, Watson et al. (1997) describe the normal flow algorithm as the baseline offering a reasonable compromise between robustness and computation time. The ODE-based algorithm is described as the most robust, but the slowest, and the augmented Jacobian algorithm as the least robust, but fastest.

Table 3. Performance: Path-following algorithms and dense vs. sparse Jacobian.

Algorithm/Jacobian	Time (h:m)	No. of steps	Time/step (s)
ODE-based/dense, ana.	22:50	1,596	51.5
Normal flow/dense, ana.	28:59	2,197	47.5
Aug. Jacobian/dense, ana.	25:25	2,250	40.7
ODE-based/sparse, ana.	1:28	1,579	03.4
Normal flow/sparse, ana.	1:44	2,197	02.9
Aug. Jacobian/sparse, ana.	2:43	2,195	04.5

Note. Learning-by-doing model.

Table 3 shows that the ODE-based algorithm is not always slower than the other path-following algorithms. On the contrary, in our experiments the ODE-based algorithm turns out to be fastest: While it takes more time to complete each step, it takes fewer steps to complete the path.

To further investigate this somewhat unexpected finding, in Table 4 we contrast the performance of the different path-following algorithms on separate portions of the solution path. The ODE-based algorithm is faster on the "simple" segment of the path ($\delta \in (0.03, 1]$) without multiplicity and much curvature (so that the unknowns change gradually with the homotopy parameter), but slower on the "complicated" segment of the path ($\delta \in [0, 0.03]$). The reason may lie in the different step-size adjustment procedures of the different path-following algorithms. Indeed, as a closer analysis of the output of HOMPACK90 reveals, the ODE-based algorithm takes much larger (and thus fewer) steps than the other path-following algorithms on the "simple" segment of the path. In contrast, on the "complicated" segment of the path, the ODE-based algorithm takes much smaller (and thus more) steps than the other path-following algorithms.[8]

Returning to Table 3, the comparison between the normal flow and augmented Jacobian algorithms is not clear-cut either. The dense augmented Jacobian algorithm takes less time for each step but requires more steps, and this leads to an overall decrease in computation time. In contrast, the sparse augmented Jacobian algorithm takes more time for each step but requires fewer steps, and this leads to an overall increase in computation time. Although the sparse augmented Jacobian algorithm takes only two fewer steps than the sparse normal flow algorithm in Table 3, Borkovsky et al. (2007) have found that in some applications both the dense and the sparse versions of the augmented Jacobian

Table 4. Performance: "Complicated" vs. "simple" segment of path.

	"Complicated" ($\delta \in [0, 0.03]$)			"Simple" ($\delta \in (0.03, 1]$)		
Algorithm/Jacobian	Time (h:m)	No. of steps	Time/step (s)	Time (h:m)	No. of steps	Time/step (s)
ODE-based/sparse, ana.	0:31	507	3.7	0:57	1,072	3.2
Normal flow/sparse, ana.	0:18	292	3.9	1:26	1,905	2.8
Aug. Jacobian/sparse, ana.	0:26	290	5.4	2:17	1,905	4.3

Note. Learning-by-doing model.

algorithm sometimes take up to 20% fewer steps than their normal flow counterparts.

Jacobian. As is obvious from Table 3, the dense Jacobian algorithms require considerably more computation time. A closer analysis reveals that the additional computation time required by the dense Jacobian algorithms is spent performing linear algebra operations on the Jacobians. Overall, this makes an overwhelmingly strong case for using sparse Jacobians.

With regard to the dense Jacobian algorithms, we have found that the choice between a numerical, hand-coded analytical, or ADIFOR-generated analytical Jacobian has a negligible effect on the time per step. This is because the time required to compute the Jacobian is dwarfed by the time required to solve the system of linear equations that the algorithm must solve to compute the next step along the path.

Turning to the sparse Jacobian algorithms, precision is a key advantage of analytically computed Jacobians. Table 5 shows that although the ODE-based algorithm succeeds in completing the solution path with an analytical Jacobian, it fails to do so with a numerical Jacobian; in particular, it spends much time tracing out a short segment of the path and stops at $\delta = 0.096$ where it reaches the maximum number of steps.

On the other hand, the normal flow algorithm requires the same number of steps to complete the solution path regardless of whether an analytical or numerical Jacobian is used. Interestingly, the path is computed more quickly when a numerical Jacobian is used (presumably because computing the numerical Jacobian requires less time than computing the analytical Jacobian due to the column-by-column approach that ADIFOR requires to assemble to Jacobian). Thus, it appears that the lower precision of the numerical Jacobian is problematic for the ODE-based algorithm but not for the other path-following algorithms. The likely reason is that, in contrast to the other two path-following algorithms, the ODE-based algorithm uses the Jacobian not just in the corrector, but also in the predictor phase.

Overall, our results make an overwhelmingly strong case for using sparse Jacobians. There are also good reasons to prefer analytical over numerical Jacobians, especially because ADIFOR makes the process of computing analytical Jacobians very easy. Finally, we conclude that performance is at least partly problem specific. We therefore recommend conducting experiments on the particular application at hand. The gains from experimentation can be substantial, and experimentation is virtually costless once the system of equations and the Jacobian have been coded.

Table 5. Performance: Path-following algorithms and numerical vs. analytical Jacobian.

Algorithm/Jacobian	Time (h:m)	No. of steps	Time/step (s)
ODE-based/sparse, ana.	1:28	1,579	3.4
ODE-based/sparse, num.	>6:27	>10,000	2.3
Normal flow/sparse, ana.	1:44	2,197	2.9
Normal flow/sparse, num.	1:22	2,197	2.3

Note. Learning-by-doing model.

5. Example 2: The Quality Ladder Model

We next consider the quality ladder model of Pakes and McGuire (1994). To simplify the exposition, we restrict attention to a duopoly without entry and exit. The description of the model is abridged; please see Pakes and McGuire (1994) for details.

5.1. Model

Firms and States. The state of firm $n \in \{1, 2\}$ is $\omega_n \in \{1, \ldots, M\}$ and reflects its product quality. The vector of firms' states is $\boldsymbol{\omega} = (\omega_1, \omega_2) \in \{1, \ldots, M\}^2$, and we use $\boldsymbol{\omega}^{[2]}$ to denote the vector (ω_2, ω_1). Each period, firms first compete in the product market and then make investment decisions. The state in the next period is determined by the stochastic outcomes of these investment decisions and an industrywide depreciation shock that stems from an increase in the quality of an outside alternative. In particular, firm n's state evolves according to the law of motion $\omega_n' = \omega_n + \tau_n - \eta$, where $\tau_n \in \{0, 1\}$ is a random variable governed by firm n's investment $x_n \geqslant 0$, and $\eta \in \{0, 1\}$ is an industrywide depreciation shock. If $\tau_n = 1$, the investment is successful and the quality of firm n increases by one level. The probability of success is $\alpha x_n/(1 + \alpha x_n)$, where $\alpha > 0$ is a measure of the effectiveness of investment. If $\eta = 1$, the industry is hit by a depreciation shock and the qualities of all firms decrease by one level; this happens with probability $\delta \in [0, 1]$.

Below we first describe the static model of product market competition and then turn to investment dynamics.

Product Market Competition. The product market is characterized by price competition with vertically differentiated products. There is a continuum of consumers. Each consumer purchases at most one unit of one product. The utility a consumer derives from purchasing product n is $g(\omega_n) - p_n + \epsilon_n$, where

$$g(\omega_n) = \begin{cases} \omega_n & \text{if } 1 \leqslant \omega_n \leqslant \omega^*, \\ \omega^* + \ln(2 - \exp(\omega^* - \omega_n)) & \text{if } \omega^* < \omega_n \leqslant M \end{cases}$$

maps the quality of the product into the consumer's valuation for it, p_n is the price, and ϵ_n represents the consumer's idiosyncratic preference for product n. There is an outside alternative, product 0, that has utility ϵ_0. Assuming that the idiosyncratic preferences $(\epsilon_0, \epsilon_1, \epsilon_2)$ are independently and identically type 1 extreme value distributed, the demand for firm n's product is

$$D_n(\mathbf{p}; \boldsymbol{\omega}) = m \frac{\exp(g(\omega_n) - p_n)}{1 + \sum_{j=1}^{2} \exp(g(\omega_j) - p_j)},$$

where $\mathbf{p} = (p_1, p_2)$ is the vector of prices and $m > 0$ is the size of the market (the measure of consumers).

Firm n chooses the price p_n of product n to maximize profits. Hence, firm n's profits in state $\boldsymbol{\omega}$ are

$$\pi_n(\boldsymbol{\omega}) = \max_{p_n} D_n(p_n, p_{-n}(\boldsymbol{\omega}); \boldsymbol{\omega})(p_n - c),$$

where $p_{-n}(\boldsymbol{\omega})$ is the price charged by the other firm and $c \geqslant 0$ is the marginal cost of production. Given a state $\boldsymbol{\omega}$, there exists a unique Nash equilibrium of the product market game (Caplin and Nalebuff 1991). It is found easily by numerically solving the system of first-order conditions corresponding to firms' profit-maximization problem. Note that the quality ladder model differs from the learning-by-doing model in that product market competition does not directly affect state-to-state transitions and, hence, $\pi_n(\boldsymbol{\omega})$ can be computed before the Markov-perfect equilibria of the dynamic stochastic game are computed via the homotopy method. This allows us to treat $\pi_n(\boldsymbol{\omega})$ as a primitive in what follows.

Bellman Equation and Complementary Slackness Condition. Define $V_n(\boldsymbol{\omega})$ to be the expected net present value of firm n's cash flows if the industry is currently in state $\boldsymbol{\omega}$. The value function $\mathbf{V}_n\colon \{1, \ldots, M\}^2 \to \mathbb{R}$ is implicitly defined by the Bellman equation

$$V_n(\boldsymbol{\omega}) = \max_{x_n \geqslant 0} \pi_n(\boldsymbol{\omega}) - x_n + \beta\left(\frac{\alpha x_n}{1+\alpha x_n} W_n^1(\boldsymbol{\omega}) + \frac{1}{1+\alpha x_n} W_n^0(\boldsymbol{\omega})\right), \quad (11)$$

where $\beta \in (0, 1)$ is the discount factor and $W_n^{\tau_n}(\boldsymbol{\omega})$ is the expectation of firm n's value function conditional on an investment success ($\tau_n = 1$) and failure ($\tau_n = 0$), respectively, as given by

$$W_n^{\tau_n}(\boldsymbol{\omega}) = \sum_{\eta \in \{0,1\},\, \tau_{-n} \in \{0,1\}} \delta^{\eta}(1-\delta)^{1-\eta}\left(\frac{\alpha x_{-n}(\boldsymbol{\omega})}{1+\alpha x_{-n}(\boldsymbol{\omega})}\right)^{\tau_{-n}} \cdot \left(\frac{1}{1+\alpha x_{-n}(\boldsymbol{\omega})}\right)^{1-\tau_{-n}} \times V_n(\max\{\min\{\omega_n + \tau_n - \eta, M\}, 1\}, \max\{\min\{\omega_{-n} + \tau_{-n} - \eta, M\}, 1\}), \quad (12)$$

where $x_{-n}(\boldsymbol{\omega})$ is the investment of the other firm in state $\boldsymbol{\omega}$. Note that the min and max operators merely enforce the bounds of the state space.

The policy function $\mathbf{x}_n\colon \{1, \ldots, M\}^2 \to [0, \infty)$ specifies the investment of firm n in state $\boldsymbol{\omega}$. Solving the maximization problem on the right-hand side of the Bellman equation (11), we obtain the complementary slackness condition

$$-1 + \beta\frac{\alpha}{(1+\alpha x_n)^2}(W_n^1(\boldsymbol{\omega}) - W_n^0(\boldsymbol{\omega})) \leqslant 0,$$
$$x_n\left(-1 + \beta\frac{\alpha}{(1+\alpha x_n)^2}(W_n^1(\boldsymbol{\omega}) - W_n^0(\boldsymbol{\omega}))\right) = 0, \quad (13)$$
$$x_n \geqslant 0.$$

The investment decision $x_n(\boldsymbol{\omega})$ is uniquely determined by the solution to complementary slackness condition (13) as

$$x_n(\boldsymbol{\omega}) = \frac{-1 + \sqrt{\max\{1, \beta\alpha(W_n^1(\boldsymbol{\omega}) - W_n^0(\boldsymbol{\omega}))\}}}{\alpha}. \quad (14)$$

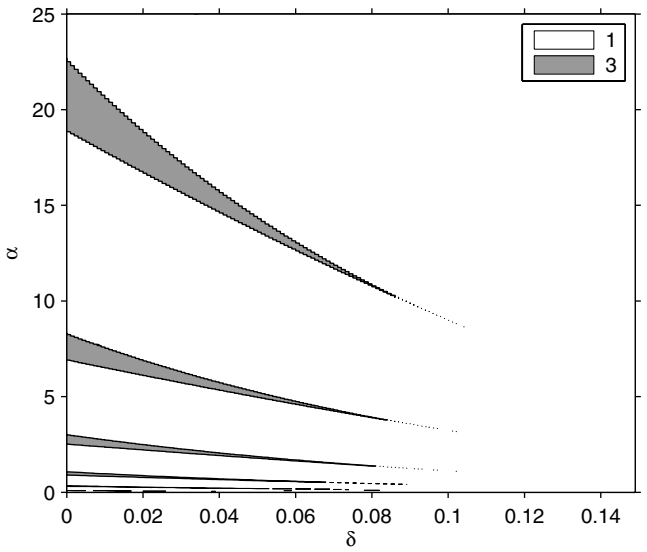
Equilibrium. We restrict attention to symmetric Markov-perfect equilibria. In a symmetric equilibrium, the investment decision taken by firm 2 in state $\boldsymbol{\omega}$ is identical to the investment decision taken by firm 1 in state $\boldsymbol{\omega}^{[2]}$, i.e., $x_2(\boldsymbol{\omega}) = x_1(\boldsymbol{\omega}^{[2]})$, and similarly for the value functions. It therefore suffices to determine the value and policy functions of firm 1, and we define $V(\boldsymbol{\omega}) = V_1(\boldsymbol{\omega})$ and $x(\boldsymbol{\omega}) = x_1(\boldsymbol{\omega})$ for each state $\boldsymbol{\omega}$. Similarly, we define $W^{\tau_1}(\boldsymbol{\omega}) = W_1^{\tau_1}(\boldsymbol{\omega})$ for each state $\boldsymbol{\omega}$.

Parameterization. We allow the homotopy algorithm to vary α and δ:

$$\begin{bmatrix}\alpha(\lambda)\\ \delta(\lambda)\end{bmatrix} = \begin{bmatrix}\alpha^{start}\\ \delta^{start}\end{bmatrix} + \lambda\begin{bmatrix}\alpha^{end} - \alpha^{start}\\ \delta^{end} - \delta^{start}\end{bmatrix}.$$

For example, if $\delta^{start} = 0$ and $\delta^{end} = 1$ while $\alpha^{start} = \alpha^{end}$, then the homotopy algorithm traces out the equilibrium correspondence from $\delta(0) = 0$ to $\delta(1) = 1$. In general, given any starting and ending values for the parameter vector, the homotopy algorithm can trace out an entire path of equilibria by moving along the line in parameter space that connects the starting and ending values. The effectiveness of investment α is a natural parameter to vary because the equilibrium trivially involves no investment if $\alpha = 0$. Moreover, this equilibrium is unique. In addition to α, we allow the rate of depreciation δ to vary because experience suggests that the rate of depreciation is often a key determinant of industry structure and dynamics (see, e.g., Besanko and Doraszelski 2004, Besanko et al. 2010b). Note that in the quality ladder model the equilibrium is not unique at $\delta = 0$ (see Figure 5) and may not be unique at $\delta = 1$. We hold the remaining parameters fixed at the values shown in Table 6.

Figure 5. Number of equilibria.



Note. Quality ladder model.

Table 6. Parameter values.

Parameter	M	m	c	ω^*	β
Value	18	5	5	12	0.925

Note. Quality ladder model.

5.2. The Zangwill and Garcia (1981) Reformulation of the Complementary Slackness Condition

Due to the nonnegativity constraint on investment in the quality ladder model, we obtained a complementary slackness condition instead of a first-order condition as in the learning-by-doing model in §4. To apply the homotopy method, we must reformulate the system of equations and inequalities in (13) as a system of equations.

Consider a general complementary slackness condition on a scalar variable x:

$$\begin{aligned} &A(x) \leqslant 0,\\ &B(x) \leqslant 0,\\ &A(x)B(x) = 0, \end{aligned} \tag{15}$$

where $A(x)$ and $B(x)$ are functions of x. Zangwill and Garcia (1981, pp. 65–68) offer a reformulation of the complementary slackness condition that consists entirely of equations that are continuously differentiable to an arbitrary degree. The idea is to introduce another scalar variable ζ and consider the system of equations

$$A(x) + [\max\{0, \zeta\}]^k = 0, \tag{16}$$

$$B(x) + [\max\{0, -\zeta\}]^k = 0, \tag{17}$$

where $k \in \mathbb{N}$. From Equations (16) and (17), it follows that

$$\zeta = \begin{cases} [-A(x)]^{1/k} & \text{if } A(x) < 0,\\ -[-B(x)]^{1/k} & \text{if } B(x) < 0,\\ 0 & \text{if } A(x) = B(x) = 0. \end{cases} \tag{18}$$

Using the fact that $\max\{0, -\zeta\} \times \max\{0, \zeta\} = 0$ and the solution for ζ in Equation (18), it is easy to see that the system of Equations (16) and (17) is equivalent to the complementary slackness condition in (15). Moreover, this system is $(k-1)$ times continuously differentiable with respect to ζ. Although we take $k = 2$ in what follows, we could easily satisfy the smoothness requirement of the homotopy method by choosing a larger value for k.

The Quality Ladder Model. The complementary slackness condition (13) can be restated as

$$\begin{aligned} &-(1+\alpha x(\boldsymbol{\omega}))^2 + \beta\alpha(W^1(\boldsymbol{\omega}) - W^0(\boldsymbol{\omega})) \leqslant 0,\\ &x(\boldsymbol{\omega})(-(1+\alpha x(\boldsymbol{\omega})^2 + \beta\alpha(W^1(\boldsymbol{\omega}) - W^0(\boldsymbol{\omega}))) = 0,\\ &x(\boldsymbol{\omega}) \geqslant 0, \end{aligned} \tag{19}$$

where we use the fact that we focus on symmetric equilibria in order to eliminate firm indices and multiply through by $(1+\alpha x(\boldsymbol{\omega}))^2$ to simplify the expression. Applying the Zangwill and Garcia (1981) reformulation to the complementary slackness condition (19) yields the equations

$$\begin{aligned} &-(1+\alpha x(\boldsymbol{\omega}))^2 + \beta\alpha(W^1(\boldsymbol{\omega}) - W^0(\boldsymbol{\omega}))\\ &\quad + [\max\{0, \zeta(\boldsymbol{\omega})\}]^k = 0, \end{aligned} \tag{20}$$

$$-x(\boldsymbol{\omega}) + [\max\{0, -\zeta(\boldsymbol{\omega})\}]^k = 0. \tag{21}$$

The terms $[\max\{0, \zeta(\boldsymbol{\omega})\}]^k$ and $[\max\{0, -\zeta(\boldsymbol{\omega})\}]^k$ serve as slack variables that ensure that the inequalities in (13) are satisfied and the fact that $\max\{0, \zeta(\boldsymbol{\omega})\} \times \max\{0, -\zeta(\boldsymbol{\omega})\} = 0$ ensures that the equality in (13) holds.

We can now proceed to define the system of homotopy equations using Equations (20) and (21) along with the Bellman equation

$$\begin{aligned} &-V(\boldsymbol{\omega}) + \pi_1(\boldsymbol{\omega}) - x(\boldsymbol{\omega})\\ &\quad + \beta\left(\frac{\alpha x(\boldsymbol{\omega})}{1+\alpha x(\boldsymbol{\omega})} W^1(\boldsymbol{\omega}) + \frac{1}{1+\alpha x(\boldsymbol{\omega})} W^0(\boldsymbol{\omega})\right) = 0, \end{aligned} \tag{22}$$

where we substitute for $W^1(\boldsymbol{\omega})$ and $W^0(\boldsymbol{\omega})$ using definition (12) (imposing symmetry). This yields a system of $3M^2$ equations in the $3M^2$ unknowns $V(1,1), \ldots, V(M,M)$, $x(1,1), \ldots, x(M,M)$, and $\zeta(1,1), \ldots, \zeta(M,M)$.

Two problems arise. First, because we have added the slack variables, this system of equations is relatively large with $3M^2$ equations and unknowns. This leads to increased memory requirements and computation time. Second, this system of equations yields an extremely sparse Jacobian. Note that the rows of the Jacobian corresponding to Equation (21) each have only one or two nonzero elements. Also note that each column of the Jacobian corresponding to a slack variable has only one nonzero element. We have found that such a Jacobian tends to cause HOMPACK90's sparse linear equation solver to fail; this is discussed further in §5.6.

We address these problems by solving Equation (21) for

$$x(\boldsymbol{\omega}) = [\max\{0, -\zeta(\boldsymbol{\omega})\}]^k, \tag{23}$$

and then substituting for $x(\boldsymbol{\omega})$ in Equations (20) and (22).[9] This reduces the system of $3M^2$ equations in $3M^2$ unknowns to a system of $2M^2$ equations in $2M^2$ unknowns. Moreover, it eliminates the rows and columns of the Jacobian that included only one or two nonzero elements; thus, we have eliminated the excessive sparsity that tends to cause HOMPACK90's sparse linear equation solver to fail.

To this end, define the vector of unknowns in equilibrium as

$$\begin{aligned} \mathbf{x} = [&V(1,1), V(2,1), \ldots, V(M,1), V(1,2), \ldots, V(M,M),\\ &\zeta(1,1), \ldots, \zeta(M,M)]'. \end{aligned}$$

The equations in state $\boldsymbol{\omega}$ are

$$H^1_{\boldsymbol{\omega}}(\mathbf{x},\lambda) = -V(\boldsymbol{\omega}) + \pi_1(\boldsymbol{\omega}) - x(\boldsymbol{\omega}) + \beta\left(\frac{\alpha x(\boldsymbol{\omega})}{1+\alpha x(\boldsymbol{\omega})} W^1(\boldsymbol{\omega}) + \frac{1}{1+\alpha x(\boldsymbol{\omega})} W^0(\boldsymbol{\omega})\right) = 0, \tag{24}$$

$$H^2_{\boldsymbol{\omega}}(\mathbf{x},\lambda) = -(1+\alpha x(\boldsymbol{\omega}))^2 + \beta\alpha(W^1(\boldsymbol{\omega}) - W^0(\boldsymbol{\omega})) + [\max\{0, \zeta(\boldsymbol{\omega})\}]^k = 0, \tag{25}$$

where we substitute for $W^1(\boldsymbol{\omega})$ and $W^0(\boldsymbol{\omega})$ using definition (12) (imposing symmetry) and for $x(\boldsymbol{\omega})$ using definition (23). Note that (24) and (25) are equations that are used to construct the system of homotopy equations, whereas (12) and (23) are simply definitional shorthands for terms that appear in Equations (24) and (25). The collection of Equations (24) and (25) for all states $\boldsymbol{\omega} \in \{1, \ldots, M\}^2$ can be written more compactly as

$$\mathbf{H}(\mathbf{x},\lambda) = \begin{bmatrix} H^1_{(1,1)}(\mathbf{x},\lambda) \\ H^1_{(2,1)}(\mathbf{x},\lambda) \\ \vdots \\ H^2_{(M,M)}(\mathbf{x},\lambda) \end{bmatrix} = \mathbf{0}, \tag{26}$$

where $\mathbf{0} \in \mathbb{R}^{2M^2}$ is a vector of zeros. Any solution to this system of $2M^2$ equations in $2M^2$ unknowns, $\mathbf{x} \in \mathbb{R}^{2M^2}$, is a symmetric equilibrium in pure strategies (for a given value of $\lambda \in [0, 1]$).[10] The equilibrium investment decision $x(\boldsymbol{\omega})$ in state $\boldsymbol{\omega}$ is recovered by substituting the equilibrium slack variable $\zeta(\boldsymbol{\omega})$ into definition (23).

Our approach of replacing a model variable with a slack variable can be taken only if one of the equations in the Zangwill and Garcia (1981) formulation admits a closed-form solution for a model variable. (In the case of the quality ladder model, we solved Equation (21) for the investment decision $x(\boldsymbol{\omega})$.) This is always the case if a model variable is constrained to be above/below a constant, as with the nonnegativity constraint in the quality ladder model. However, it is possible that none of the equations in the Zangwill and Garcia (1981) formulation admits a closed-form solution for a model variable. Suppose, for example, we impose an upper bound on the sum of firms' investments in each state, i.e., $x_n(\omega) + x_{-n}(\omega) \leqslant L(\omega)$, in the quality ladder model (say because firms are competing for a scare resource). Then, in solving an equation corresponding to (16) or (17) for $x_n(\boldsymbol{\omega})$, one finds that $x_n(\boldsymbol{\omega}) = f(\zeta_n(\boldsymbol{\omega}), x_{-n}(\boldsymbol{\omega})) = f(\zeta_n(\boldsymbol{\omega}), x_n(\boldsymbol{\omega}^{[2]}))$. That is, the closed-form solution for firm n's policy in state ω, $x_n(\boldsymbol{\omega})$, is a function of its rival's policy in state $\boldsymbol{\omega}$, $x_{-n}(\boldsymbol{\omega})$, and thus its own policy in state $\boldsymbol{\omega}^{[2]}$, $x_n(\boldsymbol{\omega}^{[2]})$. In this case, it is impossible to find a closed-form solution for $x_n(\boldsymbol{\omega})$ as a function of only $\zeta_n(\boldsymbol{\omega})$, and thus it is impossible to eliminate the model variable $x_n(\boldsymbol{\omega})$. On the other hand, in this case, the Jacobian of the system formulated using the "pure" version of the Zangwill and Garcia (1981) formulation is no longer as sparse, thereby reducing our motivation for replacing a model variable with a slack variable in the first place.

5.3. Equilibrium Correspondence

Figure 5 shows the number of equilibria as a function of the effectiveness of investment α and the rate of depreciation δ. We have found up to three equilibria for some values of α and δ. The extent of multiplicity is not nearly as dramatic as in the learning-by-doing model.

To visualize the equilibrium correspondence, we graph the expected Herfindahl index

$$HHI^T = \frac{1}{m^2} \sum_{\boldsymbol{\omega} \in \{1,\ldots,M\}^2} [(D_1(\mathbf{p}(\boldsymbol{\omega}); \boldsymbol{\omega}))^2 + (D_2(\mathbf{p}(\boldsymbol{\omega}); \boldsymbol{\omega}))^2] \mu^T(\boldsymbol{\omega}),$$

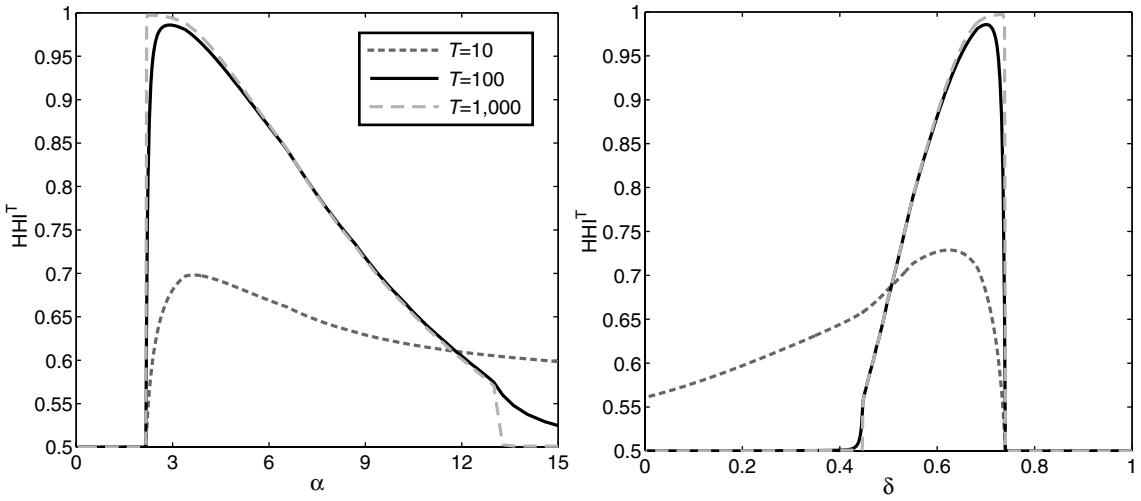
where μ^T is the transient distribution over states in period $T \in \{10, 100, 1{,}000\}$, starting from state $(1, 1)$ in period 0. We use transient distributions rather than the limiting distribution as in the learning-by-doing model because there may be several closed communicating classes.[11]

As can be seen in Figure 6, industry concentration, both in the short and in the long run, is affected by α and δ in nontrivial ways. Whereas the homotopy algorithm computes continuous solution paths, the expected Herfindahl indices in Figure 6 appear to change almost discontinuously in some places. This happens because the shape of the transient distribution, and with it the value of the expected Herfindahl index, changes abruptly as investment in certain states goes to zero. In particular, if investment in state $(1, 1)$ is zero, then both firms are stuck at the lowest-possible quality level. As soon as $x(1, 1) > 0$, however, the industry takes off, thereby giving rise to a nontrivial transient distribution that assigns positive probability to asymmetric industry structures. For example, with $\delta = 0.7$ investment rises from zero to positive around $\alpha = 2.17$ to cause the abrupt change in the expected Herfindahl index in the left panel of Figure 6; with $\alpha = 3$ investment drops from positive to zero around $\delta = 0.74$ in the right panel.

Figure 7 illustrates the ability of the homotopy algorithm to crisscross the parameter space.[12] It combines several slices through the equilibrium correspondence to show how the expected Herfindahl index $HHI^{1,000}$ depends jointly on the effectiveness of investment α and the rate of depreciation δ.

To illustrate why the multiplicity that is displayed in Figure 5 is not discernible in Figure 7, we discuss the three equilibria that we have found at $\alpha = 1.8$ and $\delta = 0.05$. Across these equilibria, investment differs most in state $(15, 15)$, where it is 0, 0.0092, and 0.0365. Table 7 presents the corresponding transient distributions $\mu^{1,000}$ in a subset of the state space around state $(15, 15)$. The differences

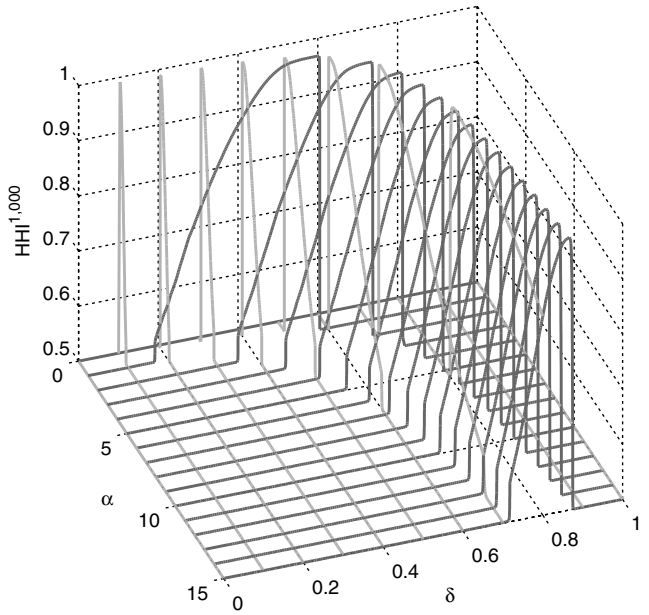
Figure 6. Transient expected Herfindahl index HHI^T at $T \in \{10, 100, 1{,}000\}$ along α with $\delta = 0.7$ (left panel) and along δ with $\alpha = 3$ (right panel).



Note. Quality ladder model.

in investment lead to qualitative differences in industry structure; whereas the equilibria in the left and middle panels give rise to transient distributions $\mu^{1,000}$ with symmetric modal state $(15, 15)$, the equilibrium in the right panel gives rise to a transient distribution $\mu^{1,000}$ with asymmetric modal states $(15, 16)$ and $(16, 15)$. Despite this, the expected Herfindahl indices are virtually identical (and equal to 0.5000) because significant differences between the transient distributions $\mu^{1,000}$ are limited to states in which each firm has a quality level greater than $\omega^* = 12$.

Figure 7. Transient expected Herfindahl index $HHI^{1,000}$ along α and δ.



Note. Quality ladder model.

For this subset of the state space, a change in the state has a negligible impact on product market competition and, accordingly, on the Herfindahl index.

5.4. Scalability

To assess the scalability of HOMPACK90, we change the number of quality levels M, and thus the number of equations $2M^2$, and adjust the quality cutoff ω^* accordingly. We trace out the equilibrium correspondence along $\alpha \in [0, 15]$ with $\delta = 0.7$ held fixed. From Table 8 it appears that the total computation time increases more than linearly in the number of equations. It is to be expected that the time per step increases in the number of equations because solving the systems of linear equations described in §3.1 becomes more burdensome. More surprisingly, the number of steps increases in the number of equations.

Table 7. Transient distributions $\mu^{1,000}$ for the three equilibria at $\alpha = 1.8$ and $\delta = 0.05$.

	15	16
15	0.6888	10^{-5}
16	10^{-5}	0

	15	16
15	0.4274	0.1418
16	0.1418	0.0021

	15	16
15	0.1921	0.2683
16	0.2683	0.0139

Note. Quality ladder model.

Table 8. Scalability: Normal flow algorithm with sparse analytic Jacobian.

M	ω^*	No. of equations	Time (h:m:s)	No. of steps	Time/step (s)
9	6	162	0:00:15	931	0.02
18	12	648	0:24:27	7,608	0.19
27	18	1,458	5:08:27	21,457	0.86

Note. Quality ladder model.

The reason is the following. Recall from §5.2 that the quality ladder model exhibits a kink as the investment in a state switches from zero to positive or vice versa in response to a change in the parameter values (see Equation (14)). While the Zangwill and Garcia (1981) reformulation of the complementary slackness condition smoothes out this kink, it inevitably does so by introducing additional curvature into the solution path. This forces the homotopy algorithm to take small steps. Moreover, the larger the state space, the more kinks there potentially are in the quality ladder model and the more additional curvature is introduced by the Zangwill and Garcia (1981) reformulation. This argument implies that the homotopy algorithm should take large steps and proceed quickly as long as the solution path does not exhibit kinks. Indeed, irrespective of the size of the state space, the homotopy algorithm takes fewer than a hundred steps to traverse the segment along which investment is positive for all states; the rest of the steps are needed to trace out the segment along which investment in some state switches from zero to positive or vice versa.

5.5. Troubleshooting

If HOMPACK90 successfully follows a path to its end, it indicates a normal ending (exit flag 1). The end of the path may be associated with either $\lambda = 1$ or $\lambda = 0$. The latter case, in turn, may indicate genuine multiplicity of equilibria (see case (B) in Figure 1) or that the homotopy algorithm "turned around" and backtracked along the path until it returned to the starting point. HOMPACK90 may also fail to follow a path to its end for other reasons. In the remainder of this section, we detail several types of failures that may occur and give tips for troubleshooting these problems.

With any type of failure, it is good practice to first verify that the regularity and smoothness requirements are satisfied. To check for regularity, we compute the condition numbers of Jacobians along the path. If the condition numbers increase as the homotopy algorithm approaches the point of failure, it is very likely due to a violation of the regularity requirement.[13] It may be possible to avoid this type of failure by making a small change in the parameter values of the model or by relaxing the precision setting so that HOMPACK90 takes larger steps and is thus more likely to "skip over" the singularity.

The homotopy algorithm does not check for smoothness, and it is entirely possible that it successfully follows a path to its end even if the smoothness requirement is violated. In general, however, it is advisable to formulate the problem such that the smoothness requirement is satisfied (see §5.2).

HOMPACK90 may abort if the precision setting is too stringent (exit flags 2 and 6) or too lax (exit flag 5). In the latter case, the homotopy algorithm takes a step and ends up too far from the path to be able to return to it; this often happens on segments with high curvature. The solution is to adjust the precision setting.

HOMPACK90 may reach the maximum number of steps (exit flag 3). Although the obvious solution is to increase the maximum number of steps, it is worth investigating if the homotopy algorithm proceeds slowly because the precision setting is too stringent. The tighter the precision setting, the narrower the "band" around the solution path in which the homotopy algorithm aims to stay and, thus, the smaller the steps that it takes. Also recall from §4.3 that the numerical Jacobian often lacks the precision that allows the ODE-based algorithm to take long steps and proceed quickly. Finally, in the normal flow and augmented Jacobian algorithms, the maximum step size (as set in the input variable SSPAR(5)) can be increased.

If the homotopy algorithm progresses very slowly in the vicinity of the initial condition, then a useful trick is to allow the homotopy algorithm to instead begin at the parameterization originally designated as the end point and proceed "backwards" toward the parameterization originally designated as the starting point. This may alleviate the problem in cases where it allows the homotopy algorithm to approach the segment of high curvature from a segment of low curvature. We suspect that this occurs because some of the path-following algorithms—namely, the normal flow and augmented Jacobian algorithms—predict the next step on the solution path using several previous steps. A segment of low curvature on the solution path may therefore provide the homotopy algorithm with "data" on the path that serves as a good indication of the direction in which to proceed.

If a solution path gets sufficiently close to another, then the homotopy algorithm may jump from one path to another and, in doing so, may fail to traverse the path in its entirety. Similarly, the homotopy algorithm may also jump between one or more segments of the same path. If path jumping is suspected to occur, then it is advisable to tighten the precision setting and/or decrease the maximum step size in order to force the homotopy algorithm to remain close to the desired solution path.

5.6. The Linear Solver

All the path-following algorithms must solve a system of linear equations of the form $\mathbf{Az} = \mathbf{b}$ in each step, where the matrix $\mathbf{A}$ is constructed from the Jacobian $\partial\mathbf{H}(\mathbf{y})/\partial\mathbf{y}$ (Equation (4) in the case of the ODE-based algorithm and the Newton iteration in the corrector step of all three algorithms).[14] The final and perhaps most troubling reason that HOMPACK90 may fail to follow a path to its end is a failure of the linear solver (exit flag 4). This occurs if the Jacobian is (nearly) singular; again, it is good practice to verify that the regularity requirement is satisfied. If this is the case, it is likely that the linear solver cannot handle the problem at hand.

The dense and sparse algorithms in HOMPACK90 differ not only in the storage format of the Jacobian, but also in the low-level numerical linear algebra routines. In our experience, the dense linear solver has been relatively

robust, whereas the sparse linear solver has sometimes failed. The dense algorithms in HOMPACK90 use QR decomposition—a direct method—to solve linear systems. The sparse algorithms use the iterative generalized minimal residual (GMRES) method (Saad and Schultz 1986) coupled with incomplete LU (ILU) preconditioning. Thus, HOMPACK90 solves $(\mathbf{Q}^{-1}\mathbf{A})\mathbf{z} = (\mathbf{Q}^{-1}\mathbf{b})$, where $\mathbf{Q}$ is the ILU preconditioner of $\mathbf{A}$. $\mathbf{Q}$ is chosen to make $(\mathbf{Q}^{-1}\mathbf{A})$ close to diagonal and easy to evaluate.

Both Layne Watson (the principal author of HOMPACK90) and Ken Judd (an authority on numerical methods in economics) acknowledge that the GMRES method can and does fail for some problems. There is no guidance as to which problems are susceptible, but we strongly suspect problems with extremely sparse Jacobians. As explained in §5.2, constructing a system of equations for the quality ladder model using the "pure" version of the Zangwill and Garcia (1981) formulation yields such a sparse Jacobian. In §5.2 we have therefore shown how to reduce the size and sparsity of the Jacobian by eliminating variables.

In addition, we also offer the following suggestions: (i) Reorder the unknowns and/or equations to change the order of the columns and rows of the Jacobian. (ii) Use the dense algorithms if the dimension of the problem is less than several hundred equations. (iii) Increase the limit on the number of GMRES iterations and/or increase the "k" in GMRES(k). (GMRES(k) is restarted every k iterations until the residual norm is small enough; see Watson et al. 1997.) (iv) Remove the ILU preconditioning and use GMRES by itself to solve the linear system. (v) Replace the sparse linear solver in HOMPACK90.[15]

6. Concluding Remarks

This paper provides a step-by-step guide to solving dynamic stochastic games using the homotopy method. We discuss the theory of the homotopy method and its implementation and present two detailed examples of dynamic stochastic games that are solved using this method. We refer the reader to Zangwill and Garcia (1981) for a comprehensive treatment of the theory of the homotopy method and to Allgower and Georg (1992) for a discussion of numerical methods for implementing it.

In this paper we use the homotopy method to trace out an entire path of solutions to a system of equations by varying one or more parameters of the model. This type of application is referred to as a *natural-parameter homotopy*. The homotopy method has other applications. A so-called *artificial homotopy* can be used to obtain a solution for a particular parameterization of a system of equations; it aims to compute just one equilibrium. Artificial homotopies have been widely used to compute Nash equilibria in normal and extensive form games (see Herings and Peeters 2010 and the references therein) and to solve general equilibrium models (see Eaves and Schmedders 1999 and the references therein), in particular, models with incomplete asset markets (Schmedders 1998, 1999). See Berry and Pakes (2007) for an application to estimating demand systems.

An *all-solutions homotopy* can sometimes be used to obtain all solutions to a system of equations with certain properties such as the polynomial system that characterizes the Nash equilibria of a finite game (see, e.g., McKelvey and McLennan 1996). All-solutions homotopies to solve for all Nash equilibria of a finite game have been implemented in the freely available software package Gambit (McKelvey et al. 2006). They are used by Bajari et al. (2010) to compute equilibria of static games of incomplete information and by Bajari et al. (2009a, b) within the context of estimation algorithms. Judd and Schmedders (2006) construct a computational uniqueness proof for a class of dynamic stochastic games in which movements through the state space are unidirectional and the primitives are given by polynomials. We refer the reader to Sommese and Wampler (2005) for a recent treatment of all-solutions homotopies.

Endnotes

1. Boldface is used throughout to distinguish between vector and scalars.
2. The mathematical literature on the homotopy method rules out paths like (D) by imposing a boundary freeness requirement (see, e.g., Zangwill and Garcia 1981, Chapter 3).
3. More formally, (G) is a so-called pitchfork bifurcation. The regularity requirement also rules out transcritical (X-shaped) bifurcations, but is consistent with other types of bifurcations (saddle node and double saddle node). See Golubitsky and Schaeffer (1985) for an introduction to bifurcation theory.
4. This functionality was added by us and is not a part of the original HOMPACK90.
5. ADIFOR can be obtained at http://www-unix.mcs.anl.gov/autodiff/ADIFOR/. Links to other automatic differentiation packages can be found at http://www.autodiff.org.
6. A slightly modified version of Proposition 2 in Doraszelski and Satterthwaite (2010) establishes that such an equilibrium always exists.
7. The orientation of the path taken by the homotopy method is arbitrary and can be reversed by reversing the signs of the basic differential equations (3).
8. Further investigation revealed that the normal flow and augmented Jacobian algorithms indeed limit the maximum step size (as set in the input variable SSPAR(5)). We kept it at the default value to make for a more fair comparison between the different path-following algorithms. Increasing the maximum step size also appears to increase the likelihood that the homotopy algorithm strays from the solution path.
9. We thank Karl Schmedders for suggesting this approach.
10. Theorem 1 in Doraszelski and Satterthwaite (2010) establishes that such an equilibrium always exists.

11. A closed communicating class is a subset of states that the industry never leaves once it has entered it. The transient distributions that we compute account for the probability of reaching any one of the closed communicating classes.
12. We thank Paul Grieco for producing Figure 7. The algorithm presented in Grieco (2008) uses HOMPACK90 to trace out paths in the equilibrium correspondence along multiple dimensions of the parameter space and, in doing so, it computes the largest connected subset of the equilibrium correspondence that includes the initial condition.
13. A matrix is singular if its condition number is infinite. A large condition number signifies that a matrix is nearly singular (Judd 1998, pp. 67–70).
14. The Jacobian $\partial\mathbf{H}(\mathbf{y})/\partial\mathbf{y}$ is an $(N \times (N+1))$ matrix, whereas the linear solver requires a square matrix. All three path-following algorithms therefore add a row to the Jacobian. This extra row is simply a basis vector in the case of the ODE-based and normal flow algorithms and a vector that is tangent to the solution path in the case of the augmented Jacobian algorithm.
15. The authors thank Layne Watson for some of these suggestions and warn the reader that implementing some of them requires in-depth knowledge of HOMPACK90.

Acknowledgments

The authors are greatly indebted to Guy Arie, David Besanko, Paul Grieco, Ken Judd, Lauren Lu, Mark Satterthwaite, Karl Schmedders, Che-Lin Su, Layne Watson, two anonymous referees, and audiences at the World Congress of the Game Theory Society 2008, the Econometric Society 2009 North American Summer Meeting, the University of Toronto, and Tilburg University for comments and suggestions. Borkovsky and Kryukov thank the General Motors Center for Strategy in Management at Northwestern University's Kellogg School of Management for support during this project. Doraszelski gratefully acknowledges financial support from the National Science Foundation under grant 0615615.

References

Allgower, E., K. Georg. 1992. Continuation and path following. *Acta Numerica* **2** 1–64.

Bajari, P., H. Hong, S. Ryan. 2009a. Identification and estimation of a discrete game of complete information. *Econometrica*. Forthcoming.

Bajari, P., H. Hong, J. Krainer, D. Nekipelov. 2009b. Estimating static models of strategic interactions. *J. Bus. Econom. Statist.* Forthcoming.

Bajari, P., H. Hong, J. Krainer, D. Nekipelov. 2010. Computing equilibria in static games of incomplete information using the all-solution homotopy. Working paper, University of Minnesota, Minneapolis.

Berry, S., A. Pakes. 2007. The pure characteristics demand model. *Internat. Econom. Rev.* **48**(4) 1193–1225.

Besanko, D., U. Doraszelski. 2004. Capacity dynamics and endogenous asymmetries in firm size. *RAND J. Econom.* **35**(1) 23–49.

Besanko, D., U. Doraszelski, Y. Kryukov, M. Satterthwaite. 2010a. Learning-by-doing, organizational forgetting, and industry dynamics. *Econometrica* **78**(2) 453–508.

Besanko, D., U. Doraszelski, L. Lu, M. Satterthwaite. 2010b. Lumpy capacity investment and disinvestment dynamics. *Oper. Res.* **58**(4, part 2 of 2) 1178–1193.

Bischof, C., P. Khademi, A. Mauer, A. Carle. 1996. ADIFOR 2.0: Automatic differentiation of Fortran 77 programs. *IEEE Comput. Sci. Engrg.* **3**(3) 18–32.

Borkovsky, R., U. Doraszelski, M. Satterthwaite. 2007. A dynamic quality ladder oligopoly with spillovers. Working paper, Northwestern University, Evanston, IL.

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

Doraszelski, U., A. Pakes. 2007. A framework for applied dynamic analysis in IO. M. Armstrong, R. Porter, eds. *Handbook of Industrial Organization*, Vol. 3. North-Holland, Amsterdam, 1887–1966.

Doraszelski, U., M. Satterthwaite. 2010. Computable Markov-perfect industry dynamics. *RAND J. Econom.* **41**(2) 215–243.

Eaves, C., K. Schmedders. 1999. General equilibrium models and homotopy methods. *J. Econom. Dynam. Control* **23**(9–10) 1249–1279.

Ericson, R., A. Pakes. 1995. Markov-perfect industry dynamics: A framework for empirical work. *Rev. Econom. Stud.* **62**(1) 53–82.

Ferris, M., K. Judd, K. Schmedders. 2007. Solving dynamic games with Newton's method. Working paper, University of Wisconsin, Madison.

Garcia, C., W. Zangwill. 1979. An approach to homotopy and degree theory. *Math. Oper. Res.* **4**(4) 390–405.

Golubitsky, M., D. Schaeffer. 1985. *Singularities and Groups in Bifurcation Theory*, Vol. 1. Springer, New York.

Grieco, P. 2008. Numerical approximation of an equilibrium correspondence: Method and applications. Working paper, Northwestern University, Evanston, IL.

Herings, J., R. Peeters. 2010. Homotopy methods to compute equilibria in game theory. *Econom. Theory* **42**(1) 119–156.

Judd, K. 1998. *Numerical Methods in Economics*. MIT Press, Cambridge, MA.

Judd, K., K. Schmedders. 2006. A computational approach to proving uniqueness in dynamic games. Working paper, Hoover Institution, Stanford, CA.

Judd, K., K. Schmedders, S. Yeltekin. 2007. Optimal rules for patent races. Working paper, Hoover Institution, Stanford, CA.

McKelvey, R., A. McLennan. 1996. Computation of equilibria in finite games. H. Amman, D. Kendrick, J. Rust, eds. *Handbook of Computational Economics*. North-Holland, Amsterdam, 87–142.

McKelvey, R., A. McLennan, T. Turocy. 2006. Gambit: Software tools for game theory. Technical report, California Institute of Technology, Pasadena.

Pakes, A., P. McGuire. 1994. Computing Markov-perfect Nash equilibria: Numerical implications of a dynamic differentiated product model. *RAND J. Econom.* **25**(4) 555–589.

Pakes, A., P. McGuire. 2001. Stochastic algorithms, symmetric Markov perfect equilibrium, and the "curse" of dimensionality. *Econometrica* **69**(5) 1261–1281.

Saad, Y., M. Schultz. 1986. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.* **7**(3) 856–869.

Schmedders, K. 1998. Computing equilibria in the general equilibrium model with incomplete asset markets. *J. Econom. Dynam. Control* **22**(8–9) 1375–1401.

Schmedders, K. 1999. A homotopy algorithm and an index theorem for the general equilibrium model with incomplete asset markets. *J. Math. Econom.* **32**(2) 225–241.

Sommese, A., C. Wampler. 2005. *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science*. World Scientific Publishing, Singapore.

Watson, L., S. Billups, A. Morgan. 1987. HOMPACK: A suite of codes for globally convergent homotopy algorithms. *ACM Trans. Math. Software* **13**(3) 281–310.

Watson, L., M. Sosonkina, R. Melville, A. Morgan, H. Walker. 1997. Algorithm 777: HOMPACK90: A suite of Fortran 90 codes for globally convergent homotopy algorithms. *ACM Trans. Math. Software* **23**(4) 514–549.

Zangwill, W., C. Garcia. 1981. *Pathways to Solutions, Fixed Points, and Equilibria*. Prentice Hall, Englewood Cliffs, NJ.