Carnegie Mellon University
Research Showcase

Tepper School of Business

2-22-2008

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

Ron N. Borkovsky
Northwestern University

Ulrich Doraszelski
Harvard University

Yaroslav Kryukov
Northwestern University, kryukov@andrew.cmu.edu

Follow this and additional works at: http://repository.cmu.edu/tepper

Part of the Economic Policy Commons, and the Industrial Organization Commons

Recommended Citation
Borkovsky, Ron N.; Doraszelski, Ulrich; and Kryukov, Yaroslav, "A User's Guide to Solving Dynamic Stochastic Games Using the Homotopy Method" (2008). *Tepper School of Business.* Paper 234.
http://repository.cmu.edu/tepper/234

This Working Paper is brought to you for free and open access by Research Showcase. It has been accepted for inclusion in Tepper School of Business by an authorized administrator of Research Showcase. For more information, please contact research-showcase@andrew.cmu.edu.

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

Ron N. Borkovsky† Ulrich Doraszelski‡ Yaroslav Kryukov§

February 22, 2008

Abstract

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.

*We are greatly indebted to Guy Arie, David Besanko, Paul Grieco, Ken Judd, Lauren Lu, Mark Satterthwaite, Karl Schmedders, Che-Lin Su, and Layne Watson for comments and suggestions. Borkovsky and Kryukov thank the General Motors Center for Strategy in Management at Northwestern's Kellogg School of Management for support during this project. Doraszelski gratefully acknowledges financial support from the National Science Foundation under Grant No. 0615615.

†Kellogg School of Management, Northwestern University, Evanston, IL 60208, r-borkovsky@kellogg.northwestern.edu.

‡Department of Economics, Harvard University, Cambridge, MA 02138, doraszelski@harvard.edu.

§Department of Economics, Northwestern University, Evanston, IL 60208, kryukov@northwestern.edu.

1 Introduction

There has been much interest in game-theoretic models of industry evolution and, in particular, in the framework introduced by Ericson & Pakes (1995) that is at the heart of a large and growing literature in industrial organization and other fields (see Doraszelski & Pakes (2007) and the references therein). Ericson & 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.

A particularly important concern in the literature following Ericson & Pakes (1995) is multiplicity of equilibria. The potential for multiplicity in their framework is widely recognized; see p. 570 of Pakes & McGuire (1994) and, more recently, the examples of multiple equilibria in Doraszelski & Satterthwaite (2007) and Besanko, Doraszelski, Kryukov & Satterthwaite (2007).

Multiple equilibria raise at least two issues. First, the existing structural estimation methods for dynamic oligopoly models such as Aguirregabiria & Mira (2007), Bajari, Benkard & Levin (2007), Pakes, Ostrovsky & Berry (2006), and Pesendorfer & Schmidt-Dengler (2003) rely on the assumption that the same equilibrium is being played in all geographic markets and/or time periods. Hence, multiple equilibria cast doubt on the estimation results unless one can convincingly argue that the same equilibrium is indeed being played in all geographic markets and/or time periods. Of course, this is trivially true if the equilibrium is unique. It would therefore be useful to be able to explore the set of equilibria of dynamic oligopoly models to determine whether multiplicity exists in the empirically-relevant subset of the parameter space. The homotopy method offers a way of doing this.

Second, multiple equilibria limit what we can learn from conducting policy experiments. Strictly speaking, the most one can conclude from a policy experiment in the presence of multiple equilibria is that, were the change in policy to occur, one of several equilibria would be played, but which one is not known. However, if we are able to more fully characterize the set of equilibria, then it becomes possible to bound the range of outcomes that may be produced by a change in policy. The homotopy method is again a useful tool for this purpose.

Computing a Markov-perfect equilibrium of a dynamic stochastic game amounts to solving a large system of equations. To date the Pakes & 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. To identify more than one equilibrium (for a given parameterization of the model), the Pakes & McGuire (1994) algorithm must be restarted from different initial guesses. But different initial guesses may or may not lead to different equilibria. A similar remark applies to the stochastic approximation algorithm of Pakes & 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 & McGuire (1994) algorithm from different initial guesses is sure to miss a substantial fraction of them, regardless of how many initial guesses are tried. That is, as shown by Besanko et al. (2007), if a dynamic stochastic game has multiple equilibria, then some of them cannot possibly be computed by the Pakes & McGuire (1994) algorithm. It is therefore important 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 selected parameters of the model. Whenever we can find such a path and multiple equilibria are the result of the path folding 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 computes 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, Sosonkina, Melville, Morgan & Walker (1997) that implements this method. We also discuss potential problems that one may encounter in using HOMPACK90 to solve dynamic stochastic games and offer some guidance as to how to resolve them.

We then 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. (2007), 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 & 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 fact, we offer several such reformulations that may be useful if complications arise.

In sum, this paper provides a step-by-step guide to solving dynamic stochastic games using the homotopy method. Our goal here is not to provide a comprehensive treatment of the theory of the homotopy method (see Zangwill & Garcia (1981) for an excellent introduction to the homotopy method) or the possibilities for implementing this method on a computer (see Allgower & Georg (1992) among others); rather, it is to enable to reader to start using HOMPACK90 as quickly as possible. To this end, we also make the codes for the learning-by-doing and quality ladder models available on our homepages. We include additional detailed instructions on how to set up and use these codes with the codes themselves.

Most of this paper is devoted to explaining how to use the homotopy method to explore the equilibrium correspondence of a dynamic stochastic game in a systematic fashion. To this end, we use the homotopy algorithm to trace out an entire path of solutions to a system of equations by varying a parameter of interest. This type of application is refereed 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. An *all-solutions homotopy* can sometimes be used to obtain all solutions to systems of equations with certain properties; it aims to compute all equilibria. We briefly discuss these applications at the end of the paper.[1]

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). Collecting Bellman equations and optimality conditions for each player in each state, the equilibrium conditions amount to a system of equations of the form

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

where $\mathbf{x}$ is the vector of the unknown values and policies and $\mathbf{0}$ is a vector of zeros, and we use boldface to distinguish between vector and scalars. Hence, computing an equilibrium of a dynamic stochastic game amounts to solving a system of typically highly nonlinear equations.

[1] The homotopy method has also been applied in other contexts, including general equilibrium models with incomplete asset markets (Schmedders 1998, Schmedders 1999). See also Eaves & Schmedders (1999) for a summary of other applications to general equilibrium models, Berry & Pakes (2007) for an application to estimating demand systems, and Bajari, Hong, Krainer & Nekipelov (2006) for an application to estimating static games of incomplete information.

Various methods are available for solving a system of nonlinear equations (see, e.g., Chapter 5 of Judd 1998). Gaussian methods such as the Pakes & McGuire (1994) algorithm most often used to solve for equilibria of dynamic stochastic games are not guaranteed to converge. Moreover, they offer no systematic approach to computing multiple equilibria and, when multiple equilibria exist, they are unable to compute some of them (Besanko et al. 2007). Unlike the Pakes & McGuire (1994) algorithm, some nonlinear solvers – notably Newton's method – are guaranteed to converge provided that the system of equations satisfies certain conditions and the initial guess that the user provides to the algorithm as a starting point is close to the final solution. However, like the Pakes & McGuire (1994) algorithm, these algorithms are limited in their ability to compute multiple equilibria because, to find a particular equilibrium, an initial guess must be supplied that is close (perhaps very close) to it.

The homotopy method allows us to explore the equilibrium correspondence in a systematic fashion. It is therefore especially useful in models that have multiple equilibria. Starting from a single equilibrium that has already been computed for a given parameterization of the model, the homotopy method traces out an entire path of equilibria by varying a parameter of interest. Recall that the equilibrium conditions depend on the parameterization of the model. Making this dependence explicit, the above system of equations becomes

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

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. 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

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

The homotopy method aims to trace out entire paths of equilibria in $\mathbf{H}^{-1}$.

Example. An example is helpful to explain how the homotopy method works. Let $N = 1$ and consider the equation $H(x, \lambda) = 0$, where

$$H(x, \lambda) = -15.289 - \frac{\lambda}{1 + \lambda^4} + 67.500x - 96.923x^2 + 46.154x^3.$$

Here we do not use boldface for x and 0 since they are scalars. This equation implicitly relates a variable x with a parameter λ. The set of solutions $H^{-1} = \{(x, \lambda) | H(x, \lambda) = 0\}$ is graphed in Figure 1. There evidently are multiple solutions to $H(x, \lambda) = 0$, e.g., $x = 0.610$, $x = 0.707$, and $x = 0.783$ at $\lambda = 0.3$. Finding these solutions is trivial with the graph in hand, but the graph is less than straightforward to draw even in this very simple case. Whether one solves $H(x, \lambda) = 0$ for x taking λ as given or for λ taking x as given, the result

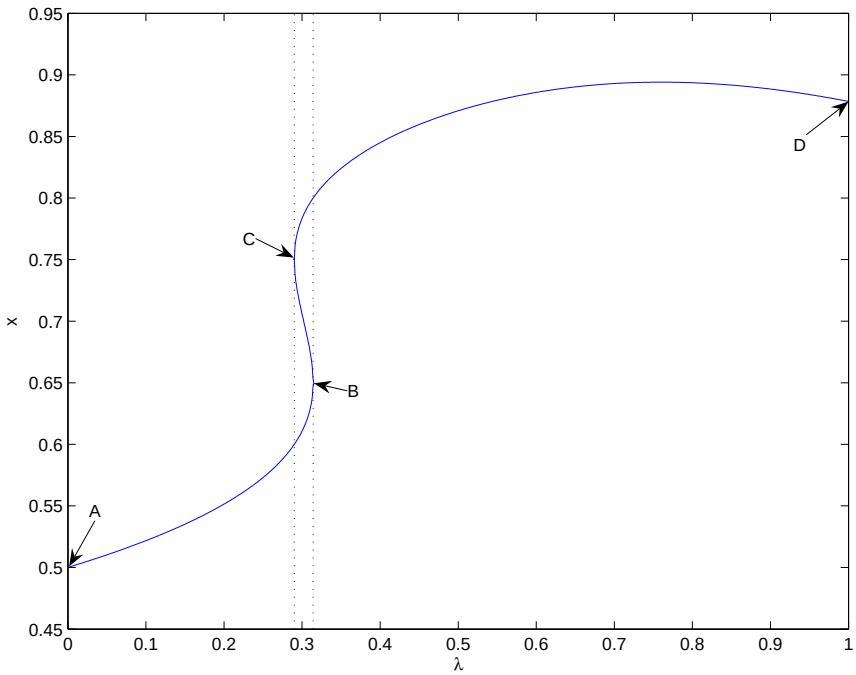

Figure 1: Example.

is a multi-valued correspondence, not a single-valued function.

To apply the homotopy method, we introduce an auxiliary variable s that indexes each point on the graph starting at point A for $s = 0$ and ending at point D for $s = \bar{s}$. The graph is then just the parametric path given by a pair of functions $(x(s), \lambda(s))$ satisfying $H(x(s), \lambda(s)) = 0$ or, equivalently, $(x(s), \lambda(s)) \in H^{-1}$. While there are infinitely many such pairs, there is a simple way to characterize a member of this family. Differentiate $H(x(s), \lambda(s)) = 0$ with respect to s to obtain

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

This differential equation in two unknowns $x'(s)$ and $\lambda'(s)$ captures the condition that is required to remain "on path." One possible approach for tracing out a path in H^{-1} is thus to solve equation (2) for the ratio $\frac{x'(s)}{\lambda'(s)} = -\frac{\partial H(x(s),\lambda(s))/\partial \lambda}{\partial H(x(s),\lambda(s))/\partial x}$ that indicates the direction of the next step along the path from s to $s + ds$. This approach, however, creates difficulties because the ratio may switch from $+\infty$ to $-\infty$, e.g., at point B in Figure 1. So instead of solving for the ratio, we simply solve for each term of the ratio. This insight implies that the graph of H^{-1} in Figure 1 is the solution to the system of differential equations

$$x'(s) = \frac{\partial H(x(s), \lambda(s))}{\partial \lambda}, \tag{3}$$

$$\lambda'(s) = -\frac{\partial H(x(s), \lambda(s))}{\partial x}. \tag{4}$$

Equations (3) and (4) are the so-called basic differential equations for our example. In our example, note that if $\lambda = 0$, then $H(x, \lambda) = 0$ is easily solved for $x = 0.5$. This provides the initial condition (point A in Figure 1). From there the homotopy method uses the basic differential equations to determine the next step along the path. It continues to follow the path – step-by-step – until it reaches $\lambda = 1$ (point D). Whenever $\lambda'(s)$ switches sign from negative to positive (point B), the path is bending backward and there are multiple solutions. Conversely, whenever the sign of $\lambda'(s)$ switches back from positive to negative (point C), the path is bending forward.[2]

Returning to the general case with $N > 1$, our goal is to explore the equilibrium correspondence $\mathbf{H}^{-1} = \{(\mathbf{x}, \lambda) | \mathbf{H}(\mathbf{x}, \lambda) = \mathbf{0}\}$ that depends on the homotopy parameter λ. Proceeding as in our example, a parametric path is a set of functions $(\mathbf{x}(s), \lambda(s)) \in \mathbf{H}^{-1}$. Differentiating $\mathbf{H}(\mathbf{x}(s), \lambda(s)) = \mathbf{0}$ with respect to s yields the conditions that are required to remain on path

$$\frac{\partial \mathbf{H}(\mathbf{x}(s), \lambda(s))}{\partial \mathbf{x}} \mathbf{x}'(s) + \frac{\partial \mathbf{H}(\mathbf{x}(s), \lambda(s))}{\partial \lambda} \lambda'(s) = \mathbf{0}, \tag{5}$$

where $\frac{\partial \mathbf{H}(\mathbf{x}(s), \lambda(s))}{\partial \mathbf{x}}$ is the $(N \times N)$ Jacobian of $\mathbf{H}$ with respect to $\mathbf{x}$, $\mathbf{x}'(s)$ and $\frac{\partial \mathbf{H}(\mathbf{x}(s), \lambda(s))}{\partial \lambda}$ are $(N \times 1)$ vectors, and $\lambda'(s)$ is a scalar. This system of N differential equations in $N + 1$ unknowns, $x_i'(s)$, $i = 1, \ldots, N$, and $\lambda'(s)$, has a solution that obeys the basic differential equations

$$y_i'(s) = (-1)^{i+1} \det \left(\left[\frac{\partial \mathbf{H}(\mathbf{y}(s))}{\partial \mathbf{y}} \right]_{-i} \right), \quad i = 1, \ldots, N + 1, \tag{6}$$

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 $\frac{\partial \mathbf{H}(\mathbf{y}(s))}{\partial \mathbf{y}}$ of $\mathbf{H}$ with respect to $\mathbf{y}$. Note that equation (6) reduces to equations (3) and (4) if $\mathbf{x}$ is a scalar instead of a vector. For the general case, a proof that the basic differential equations (6) satisfy the conditions in equation (5) that are required to remain on path can be found in Garcia & Zangwill (1979) and on pp. 27–28 of Zangwill & Garcia (1981).

Regularity and smoothness requirements. A closer inspection of the basic differential equations (6) reveals a potential difficulty. If the Jacobian $\frac{\partial \mathbf{H}(\mathbf{y}(s))}{\partial \mathbf{y}}$ is not of full rank at some point $\mathbf{y}(s)$ on the solution path, then the determinant of each of its square submatrices is zero. Thus, according to the basic differential equations (6), $y_i'(s) = 0$, $i = 1, \ldots, N + 1$, 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, $\mathbf{H}$ is regular if $\text{rank} \left(\frac{\partial \mathbf{H}(\mathbf{y})}{\partial \mathbf{y}} \right) = N$ for all $\mathbf{y} \in \mathbf{H}^{-1}$. The regularity requirement – and a certain smoothness

[2] The orientation of the path taken by the homotopy method is arbitrary. Reversing the signs of the basic differential equations implies, perhaps more intuitively, that $\lambda'(s)$ switches sign from positive to negative at point B.

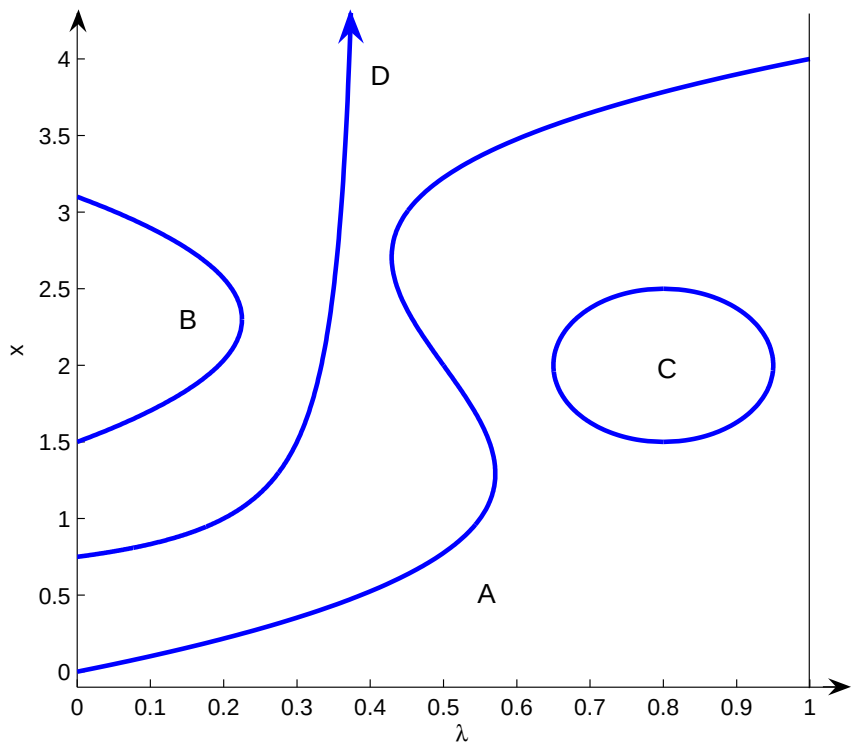

Figure 2: Examples of solution paths if $\mathbf{H}$ is regular.

requirement to be discussed below – ensures that the set of solutions $\mathbf{H}^{-1}$ consists only of continuous paths. Figure 2 shows examples of possible solution paths if $\mathbf{H}$ 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 $\mathbf{x}$ in the case of a vector) tends to $+\infty$ or $-\infty$.[3] Figure 3 shows examples of solution paths that are ruled out by the regularity requirement: (E) isolated equilibria; (F) continua of equilibria; (G) branching point;[4] (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 & 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.

[3]The mathematical literature on the homotopy method rules out paths like (D) by imposing a boundary freeness requirement (see, e.g., Chapter 3 of Zangwill & Garcia 1981).

[4]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 & Schaeffer (1985) for an introduction to bifurcation theory.

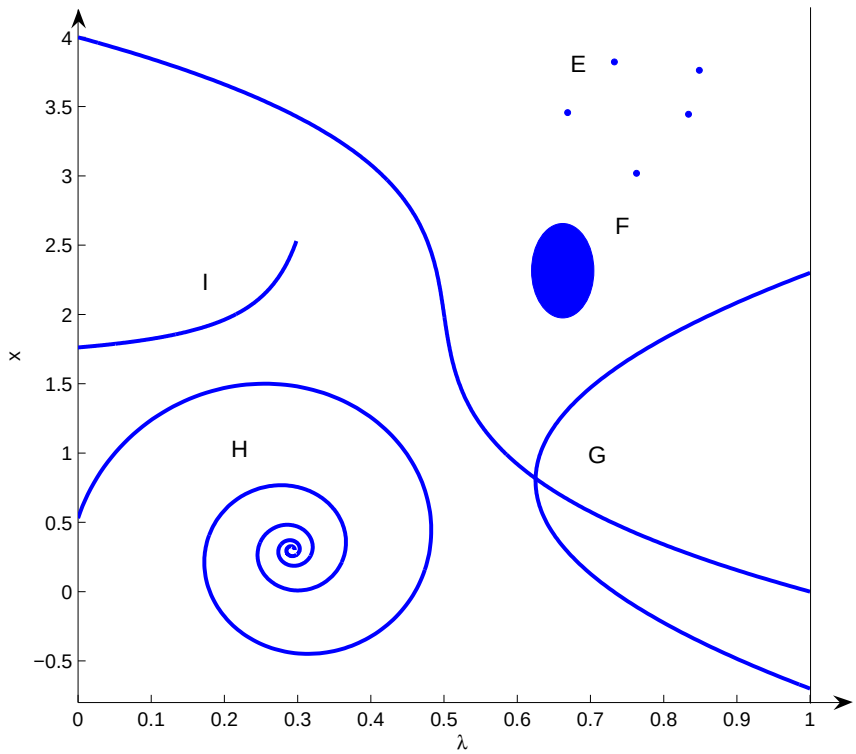

Figure 3: Examples of solution paths if $\mathbf{H}$ is not regular.

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 $\mathbf{H}$ 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 & Garcia 1981). Moreover, for a path to be described by the basic differential equations (6) it must be the case that $\mathbf{H}$ is twice continuously differentiable in addition to regular. This result is known as the BDE theorem (see, e.g., pp. 27–28 of Zangwill & Garcia 1981).

The smoothness requirement is non-trivial and easily violated, for example, by non-negativity constraints on components of $\mathbf{x}$, say because investment cannot become negative, or by distributions with non-differentiable 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.

To understand why smoothness is necessary, consider an analogy: Imagine that the solution path is lined with railroad tracks and that the homotopy algorithm follows these tracks just as a train would. Like a train the homotopy algorithm can follow a curve in the tracks, perhaps at a reduced speed, but the train derails if the tracks take a sudden turn.

There is a subtle difference between the homotopy method, a mathematical theory, and

	dense Jacobian	sparse Jacobian
ODE based	FIXPDF	FIXPDS
normal flow	FIXPNF	FIXPNS
augmented Jacobian	FIXPQF	FIXPQS

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

the homotopy algorithm, a computational 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 Section 5.5.

3 The HOMPACK90 Software Package

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

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

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 $\lambda \in [0, 1]$. The detailed description of HOMPACK90 is given in Watson, Billups & Morgan (1987) and Watson et al. (1997). Here we just give a brief overview that is meant to enable the reader to start using HOMPACK90 as quickly as possible.

In order 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 Section 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[5], and an exit flag that indicates a normal ending or several kinds of

[5]This functionality was added by us and is not a part of the original HOMPACK90.

failure. We discuss some of the potential problems in Section 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 $\left\{s^k, \mathbf{y}^k\right\}$, 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 at the solution $\mathbf{y}^{k+1}$; the corrector phase then improves that guess using a version of Newton's method. The difference between algorithms lies in the implementation of the predictor and corrector phases.

ODE based. The predictor phase of the ordinary differential equation (ODE) based path-following algorithm is a direct application of the system of differential equations (5). It first solves the system of linear equations

$$\frac{\partial \mathbf{H}\left(\mathbf{y}^k\right)}{\partial \mathbf{y}} \Delta \mathbf{y} = 0 \tag{7}$$

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

Normal flow. The predictor phase uses a Hermite cubic extrapolation from the previous two points. While the Hermite cubic extrapolation is much easier to compute than solving the system of linear equations (7), it is also much cruder. The corrector phase is thus necessary at every step.

Augmented Jacobian. This path-following algorithm is similar to the normal flow algorithm except that it takes a more sophisticated approach to the corrector phase.

3.2 Jacobian

HOMPACK90 requires the user to provide a routine that returns the Jacobian $\frac{\partial \mathbf{H}(\mathbf{y})}{\partial \mathbf{y}}$ at a given point $\mathbf{y}$. 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 with most elements being zero. 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 non-zero elements with corresponding row and column indices rather than as a dense matrix that consists of the entire set of $N\left(N+1\right)$

elements. 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 increases the time needed to complete the entire path significantly; 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 use ADIFOR, a program that analytically differentiates Fortran code. ADIFOR is described in Bischof, Khademi, Mauer & Carle (1996); here we just give a brief overview.[6]

The input to ADIFOR is the Fortran90 code that returns $\mathbf{H}(\mathbf{y})$ at a given point $\mathbf{y}$. ADIFOR analyses this code and from it generates the 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.

Dense vs. sparse Jacobian. 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 Section 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 non-zero elements. In practice, this means going through the system of equations and identifying the elements of $\mathbf{y}$ that are involved 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; Section 5.6 gives more details and some solutions.

[6] ADIFOR can be obtained at http://www-unix.mcs.anl.gov/autodiff/ADIFOR/.

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 the 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, Schmedders & Yeltekin 2002).

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 & McGuire (1994) algorithm, other nonlinear solvers (see Ferris, Judd & Schmedders 2007), and artificial homotopies (see Section 6.1). Finally, one can use a solution obtained by tracing out a path along a different parameter as an initial condition (see Section 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. (2007) because it is particularly well-suited for the homotopy method; as explained above, the dynamic programming problem that a firm has to solve leads to a system of equations that satisfies the smoothness requirement.

4.1 Model

The description of the model is abridged. Please refer to Besanko et al. (2007) for economic motivation and greater detail on some derivations.

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, while 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 $\rho \in (0,1]$:

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

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. We let

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

denote the probability that firm n loses a unit of know-how through organizational forgetting. We refer to $\delta \in [0,1]$ as the forgetting rate. If $\delta > 0$, then $\Delta(e_n)$ is increasing and concave in e_n; $\delta = 0$ corresponds to the absence of organizational forgetting.

Demand. Each period one (non-strategic) 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, a firm's stock of know-how moves up or down or remains constant depending on realized demand $q_n \in \{0,1\}$ and

organizational forgetting $f_n \in \{0,1\}$. The transition probabilities are

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

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

$$V_n(\mathbf{e}) = \max_{p_n} D_n(p_n, p_{-n}(\mathbf{e}))(p_n - c(e_n)) + \beta \sum_{k=1}^{2} D_k(p_n, p_{-n}(\mathbf{e}))\overline{V}_{nk}(\mathbf{e}), \tag{8}$$

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

$$\begin{aligned} \overline{V}_{n1}(\mathbf{e}) &= \sum_{e_1'=e_1}^{e_1+1} \sum_{e_2'=e_2-1}^{e_2} V_n(\mathbf{e}')\Pr(e_1'|e_1,1)\Pr(e_2'|e_2,0), \\ \overline{V}_{n2}(\mathbf{e}) &= \sum_{e_1'=e_1-1}^{e_1} \sum_{e_2'=e_2}^{e_2+1} V_n(\mathbf{e}')\Pr(e_1'|e_1,0)\Pr(e_2'|e_2,1). \end{aligned}$$

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 (8). Differentiating $h_n(\cdot)$ with respect to p_n we obtain the first-order condition

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

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 focus attention on 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 probability that the buyer purchases from firm $k \in \{1,2\}$ in state $\mathbf{e}$.

parameter	M	m	β
value	30	15	$\frac{1}{1.05} = 0.9524$

Table 2: Parameter values. Learning-by-doing model.

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. Accordingly, we explore the full range of values for the progress ratio ρ and the forgetting rate δ while holding fixed the remaining parameters at the values shown in Table 2.

Besanko et al. (2007) 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

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

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 $\rho \in [0.01, 1]$.

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

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

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

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

$$H^2_{\mathbf{e}}(\mathbf{x},\lambda) = 1 - \left(1 - D_1(\mathbf{e})\right)\left(p(\mathbf{e}) - c(e_1)\right) - \beta\overline{V}_1(\mathbf{e}) + \beta \sum_{k=1}^{2} D_k(\mathbf{e})\overline{V}_k(\mathbf{e}) = 0. \tag{10}$$

The collection of equations (9) and (10) for all states $\mathbf{e} \in \{1, \ldots, M\}^2$ can be written more compactly as

$$\mathbf{H}(\mathbf{x},\lambda) = \begin{bmatrix} H^1_{(1,1)}(\mathbf{x},\lambda) \\ H^1_{(2,1)}(\mathbf{x},\lambda) \\ \vdots \\ H^2_{(M,M)}(\mathbf{x},\lambda) \end{bmatrix} = \mathbf{0}, \tag{11}$$

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

$\lambda \in [0, 1])$.[7]

Code. A set of code that allows the user to compute equilibria of the learning-by-doing model using the homotopy method is available on the authors' homepages. It includes (i) Matlab code that implements the Pakes & McGuire (1994) algorithm that we use to compute a starting point for the homotopy algorithm (see Besanko et al. (2007) for a detailed description); (ii) Fortran90 code that includes HOMPACK90 and the implementation of the learning-by-doing model; and (iii) additional Matlab code that analyzes the output of the homotopy algorithm. More detailed information is included within the code itself.

4.2 Equilibrium Correspondence

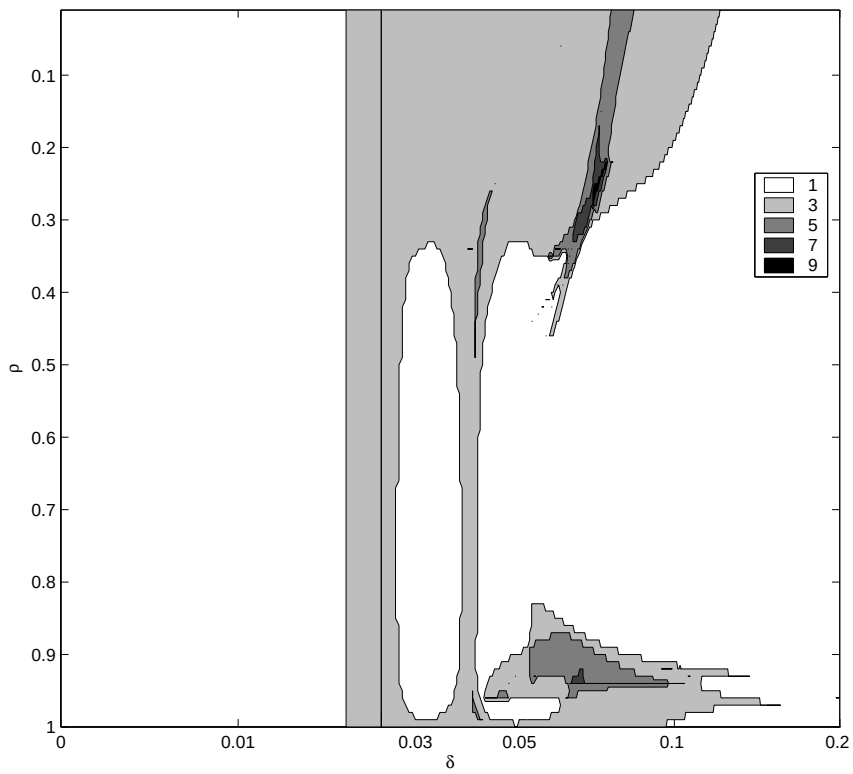

Figure 4: Number of equilibria. Learning-by-doing model.

Figure 4 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. It is not a coincidence that the number of equilibria at each parameterization is odd; see the discussion in Besanko et al. (2007).

[7] A slightly modified version of Proposition 2 in Doraszelski & Satterthwaite (2007) establishes that such an equilibrium always exists.

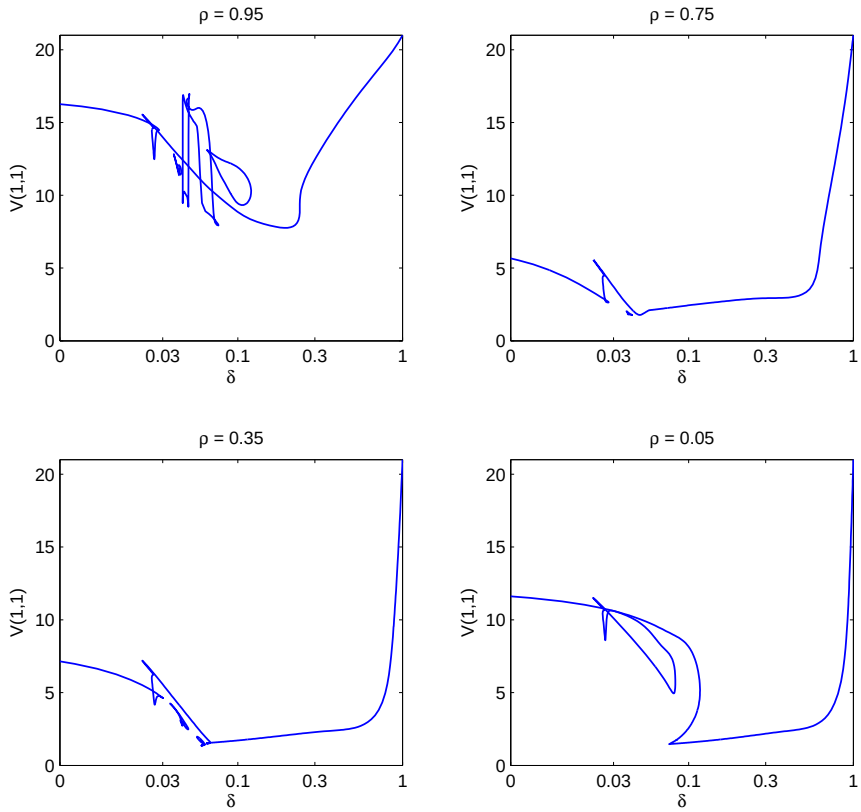

Figure 5: Initial firm value $V(1,1)$. Learning-by-doing model.

Multiplicity is especially pervasive for progress ratios and forgetting rates that are broadly consistent with empirical studies of learning-by-doing and organizational forgetting ($\rho \geq 0.7$ and $\delta \leq 0.1$). Moreover, Besanko et al. (2007) show that these 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. Consequently, in addition to the parameterization, the equilibrium itself is an important determinant of pricing behavior and industry dynamics. This reinforces our earlier point that it is important to explore the equilibrium correspondence of structurally estimated models in order to determine whether multiplicity exists in the empirically relevant subset of the parameter space.

Recall that we made the forgetting rate δ a function of the homotopy parameter λ and applied the homotopy method repeatedly for a series of values for the progress ratio ρ. 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. As we are also interested in long-run industry concentration, we further compute the expected Herfindahl index. We proceed in two steps. First, we use the policy function to construct the probability distribution

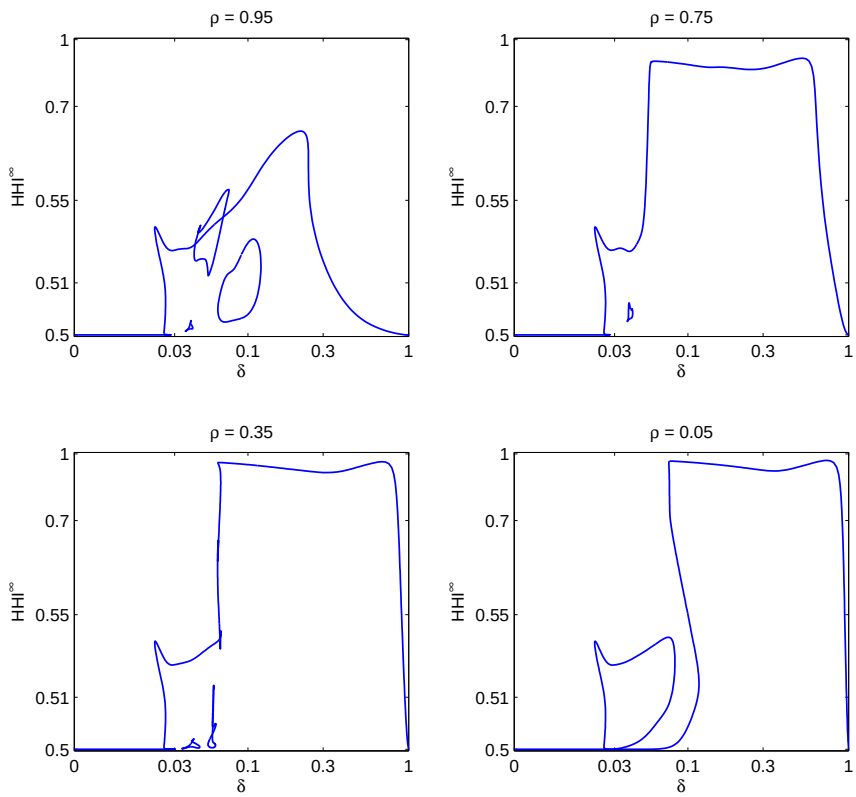

Figure 6: Limiting expected Herfindahl index HHI^{∞}. Learning-by-doing model.

over 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

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

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

Figures 5 and 6 visualize the equilibrium correspondence either in terms of $V(1,1)$ or in terms of HHI^{∞} for a variety of different progress ratios ρ. As mentioned above, Besanko et al. (2007) prove that the model has unique equilibria at $\delta = 0$ and $\delta = 1$. Hence, if the system of equations that characterizes the equilibria is regular, then there must be a path connecting them. We observe multiple equilibria whenever this path bends back on itself. Moreover, we have been able to identify one or more loops that are disjoint from this path.

In some places the various solution paths appear to intersect each other. A case in point is the loop in the upper left panel of Figure 6 that appears to twice intersect the path from $\delta = 0$ and $\delta = 1$. While such an intersection seemingly resembles the branching point (G) in Figure 3, 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 5 and 6 do not violate the regularity requirement.

While we have been able to identify some loops, we note that other loops may exist because, in order to trace out a loop, we must somehow compute at least one equilibrium on the loop. Unfortunately, there is no sure fire of way of doing so. Figures 4–6 are therefore not necessarily a complete mapping of the equilibria.

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. Below we present the results of a series of experiments that are designed to highlight the implications of these choices for the performance of HOMPACK90. We have traced out the main path of the equilibrium correspondence from $\delta = 0$ to $\delta = 1$ for a progress ratio of $\rho = 0.75$ (as shown the upper right panel of Figure 6). We set the precision in HOMPACK90 to 10^{-10} (see Section 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 1GHz AMD Athlon CPU and 4GB 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 non-trivial 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 slowest and the augmented Jacobian algorithm as the least robust but fastest.

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-

algorithm / Jacobian	time (h:m)	#steps	time/step (s)
ODE based / dense, ana.	22:50	1596	51.5
normal flow / dense, ana.	28:59	2197	47.5
aug. Jacobian / dense, ana.	25:25	2250	40.7
ODE based / sparse, ana.	1:28	1579	03.4
normal flow / sparse, ana.	1:44	2197	02.9
aug. Jacobian / sparse, ana.	2:43	2195	04.5

Table 3: Performance. Path-following algorithms and dense vs. sparse Jacobian. Learning-by-doing model.

algorithm / Jacobian	"complicated" ($\delta \in [0, 0.03]$)			"simple" ($\delta \in (0.03, 1]$)		
	time (h:m)	#steps	time/step (s)	time (h:m)	#steps	time/step (s)
ODE based / sparse, ana.	0:31	507	3.7	0:57	1072	3.2
normal flow / sparse, ana.	0:18	292	3.9	1:26	1905	2.8
aug. Jacobian / sparse, ana.	0:26	290	5.4	2:17	1905	4.3

Table 4: Performance. "Complicated" vs. "simple" segment of path. Learning-by-doing model

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, thereby leading to an overall decrease of computation time. In contrast, the sparse augmented Jacobian algorithm takes more time for each step but requires fewer steps, thereby leading to an overall increase in computation time. While the sparse augmented Jacobian algorithm takes only two fewer steps than the sparse normal flow algorithm in Table 3, Borkovsky, Doraszelski & Satterthwaite (2007) have found that in some applications both the dense and the sparse versions of the augmented Jacobian algorithm sometimes take up to 20 percent 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

[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.

algorithm / Jacobian	time (h:m)	#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

Table 5: Performance. Path-following algorithms and numerical vs. analytical Jacobian. Learning-by-doing model.

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, while 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 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.

5 Example 2: The Quality Ladder Model

We next consider the quality ladder model of Pakes & McGuire (1994). The quality ladder model presents a complication that stems from the non-negativity constraint on investment. The problem that a firm has to solve is formulated using a complementary slackness condition, a combination of equalities and inequalities, rather than a first-order condition, an equation, as in the learning-by-doing model in Section 4. However, the homotopy method operates on a system of equations. We show how to resolve this problem by reformulating the complementary slackness condition as a system of equations.

5.1 Model

The description of the model is abridged; please see Pakes & McGuire (1994) for details. To simplify the exposition, we restrict attention to a duopoly without entry and exit in what follows.[9]

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 $\omega = (\omega_1, \omega_2) \in \{1, \ldots, M\}^2$ and we use $\omega^{[2]}$ 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 industry-wide depreciation shock which 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 $x_n \geq 0$ and $\eta \in \{0, 1\}$ is an industry-wide 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 $\frac{\alpha x_n}{1+\alpha x_n}$, 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.

[9] It is straightforward to extend the quality ladder model to allow for entry and exit. The key is to do this in a way that guarantees the existence of an equilibrium; see Doraszelski & Satterthwaite (2007) for details. The Online Appendix of Besanko et al. (2007) contains a formal derivation of the learning-by-doing model with entry and exit. Setup costs and scrap values are drawn from triangular distributions that yield cumulative distribution functions that are once but not twice continuously differentiable, yet Besanko et al. (2007) did not encounter a problem. If a problem is encountered in another application, we suggest using a Beta(k, k) distribution with $k \geq 3$ to ensure that the system of equations is at least twice continuously differentiable.

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

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

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, which 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

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

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 ω are

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

where $p_{-n}(\omega)$ is the price charged by the other firm and $c \geq 0$ is the marginal cost of production. Given a state ω, there exists a unique Nash equilibrium of the product market game (Caplin & 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, $\pi_n(\omega)$ can be computed before the Markov-perfect equilibria of the dynamic stochastic game are computed via the homotopy method. This allows us to treat $\pi_n(\omega)$ as a primitive in what follows.

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

$$V_n(\omega) = \max_{x_n \geq 0} \pi_n(\omega) - x_n + \beta\left(\frac{\alpha x_n}{1 + \alpha x_n} W_n^1(\omega) + \frac{1}{1 + \alpha x_n} W_n^0(\omega)\right), \tag{12}$$

where $\beta \in (0, 1)$ is the discount factor and $W_n^{\tau_n}(\omega)$ 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

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

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

The policy function $\mathbf{x}_n : \{1,\ldots,M\}^2 \to [0,\infty)$ specifies the investment of firm n in state ω. Solving the maximization problem on the right-hand side of the Bellman equation (12), we obtain the complementary slackness condition

$$
\begin{aligned}
-1+\beta\frac{\alpha}{(1+\alpha x_n)^2}\left(W_n^1(\omega)-W_n^0(\omega)\right) &\leq 0, \\
x_n\left(-1+\beta\frac{\alpha}{(1+\alpha x_n)^2}\left(W_n^1(\omega)-W_n^0(\omega)\right)\right) &= 0, \\
x_n &\geq 0.
\end{aligned}
\tag{13}
$$

The investment decision $x_n(\omega)$ is uniquely determined by the solution to complementary slackness condition. It follows that

$$
x_n(\omega) = \max\left\{0, \frac{-1+\sqrt{\beta\alpha\left(W_n^1(\omega)-W_n^0(\omega)\right)}}{\alpha}\right\} \tag{14}
$$

if $W_n^1(\omega) - W_n^0(\omega) \geq 0$ and $x_n(\omega) = 0$ otherwise.

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

Parameterization. As explained above, the homotopy algorithm traces out an entire path of equilibria by varying one or more parameters of interest. We allow α and δ to vary, while holding the remaining parameters fixed at the values shown in Table 6. 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 & Doraszelski 2004, Besanko, Doraszelski, Lu & Satterthwaite 2006). Note that in the quality ladder model the equilibrium may not be unique at either $\delta = 0$ or $\delta = 1$.

parameter	M	m	c	ω^*	β
value	18	5	5	12	0.925

Table 6: Parameter values. Quality ladder model.

Taken together, we make the vector comprising α and δ a function of the homotopy parameter λ:

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

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 $\delta(1) = 1$, holding all other parameter values fixed. Setting different starting and ending values for one or more of these parameters allows us to explore the set of equilibria by moving through the parameter space in various directions. 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.

System of equations. Due to the non-negativity constraint on investment, we obtained a complementary slackness condition instead of a first-order condition as in the learning-by-doing model in Section 4. To apply the homotopy method, we must reformulate the combination of equalities and inequalities in (13) as equalities. In the next section we describe one way to do this.

Code. A set of code that allows the user to compute equilibria of the quality ladder model using the homotopy method is available on the authors' homepages.

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

The homotopy method operates on equations. Therefore, a model that includes a complementary slackness condition, a combination of equalities and inequalities, must be reformulated as a system of equations.

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

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

where $A(x)$ and $B(x)$ are functions of x. A complementary slackness condition may arise from an optimization problem with a non-negativity constraint as in the quality ladder

model in Section 5.1. It may also arise if a model contains min or max operators. For example, the equation $x = \min\{a(x), b(x)\}$ is equivalent to

$$\begin{aligned} x - a(x) &\leq 0, \\ x - b(x) &\leq 0, \\ (x - a(x))\,(x - b(x)) &= 0. \end{aligned}$$

Note that the equation $x = \min\{a(x), b(x)\}$ has a kink when $a(x) = b(x)$ and hence does not satisfy the smoothness requirement of the homotopy method.

Zangwill & Garcia (1981) offer a reformulation of the complementary slackness condition that consists entirely of equations that are continuously differentiable to an arbitrary degree (see pp. 65–68).[10] 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

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

Using the fact that $\max\{0, -\zeta\}\max\{0, \zeta\} = 0$ 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. Hence, by choosing k large enough, we can satisfy the smoothness requirement of the homotopy method.

Example: The quality ladder model. An example is helpful in understanding how the Zangwill & Garcia (1981) reformulation works. Consider the complementary slackness condition (13) in the quality ladder model in Section 5. Using the fact that we focus on symmetric equilibria in order to eliminate firm indices and multiplying through by $(1 + \alpha x(\omega))^2$ to simplify the expressions that arise in what follows, the complementary slackness condition (13) can be restated as

$$\begin{aligned} -(1 + \alpha x(\omega))^2 + \beta\alpha\left(W^1(\omega) - W^0(\omega)\right) &\leq 0, \\ x(\omega)\left(-(1 + \alpha x(\omega)^2 + \beta\alpha\left(W^1(\omega) - W^0(\omega)\right)\right) &= 0, \\ x(\omega) &\geq 0. \end{aligned} \tag{19}$$

[10] As Zangwill & Garcia (1981) is out of print, a more easily accessible source may be Charnes, Garcia & Lemke (1977).

Applying the Zangwill & Garcia (1981) reformulation to the complementary slackness condition (19) yields the equations

$$
\begin{aligned}
-\left(1+\alpha x(\omega)\right)^2+\beta\alpha\left(W^1(\omega)-W^0(\omega)\right)+\left[\max\left\{0,\zeta(\omega)\right\}\right]^k &= 0, && (20)\\
-x(\omega)+\left[\max\left\{0,-\zeta(\omega)\right\}\right]^k &= 0. && (21)
\end{aligned}
$$

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

$$
-V(\omega)+\pi_1(\omega)-x(\omega)+\beta\left(\frac{\alpha x(\omega)}{1+\alpha x(\omega)}W^1(\omega)+\frac{1}{1+\alpha x(\omega)}W^0(\omega)\right)=0, \qquad (22)
$$

where we substitute for $W^{\tau_1}(\omega)$ using the definition

$$
\begin{aligned}
W^{\tau_1}(\omega) \;=\; & \sum_{\eta\in\{0,1\},\tau_2\in\{0,1\}}\delta^{\eta}(1-\delta)^{1-\eta}\left(\frac{\alpha x(\omega^{[2]})}{1+\alpha x(\omega^{[2]})}\right)^{\tau_2}\left(\frac{1}{1+\alpha x(\omega^{[2]})}\right)^{1-\tau_2}\\
& \times V\Big(\max\left\{\min\left\{\omega_1+\tau_1-\eta,M\right\},1\right\},\max\left\{\min\left\{\omega_2+\tau_2-\eta,M\right\},1\right\}\Big).
\end{aligned}
$$

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 non-zero elements. Also note that each column of the Jacobian corresponding to a slack variable has only one non-zero element. We have found that such a Jacobian tends to cause HOMPACK90's sparse linear equation solver to fail; this is discussed further in Section 5.6.

We address these problems by solving equation (21) for $x(\omega)$ and then substituting for $x(\omega)$ in equations (20) and (22).[11] 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 non-zero 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

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

[11] We thank Karl Schmedders for suggesting this approach.

The equations in state ω are

$$H_{\omega}^{1}(\mathbf{x},\lambda) = -V(\omega) + \pi_1(\omega) - x(\omega) + \beta\left(\frac{\alpha x(\omega)}{1+\alpha x(\omega)}W^1(\omega) + \frac{1}{1+\alpha x(\omega)}W^0(\omega)\right) = 0, \tag{23}$$

$$H_{\omega}^{2}(\mathbf{x},\lambda) = -(1+\alpha x(\omega))^2 + \beta\alpha\left(W^1(\omega) - W^0(\omega)\right) + \left[\max\{0,\zeta(\omega)\}\right]^k = 0, \tag{24}$$

where we substitute for $W^{\tau_1}(\omega)$ using the definition

$$\begin{aligned} W^{\tau_1}(\omega) \quad = \quad & \sum_{\eta\in\{0,1\},\tau_2\in\{0,1\}} \delta^{\eta}(1-\delta)^{1-\eta}\left(\frac{\alpha x(\omega^{[2]})}{1+\alpha x(\omega^{[2]})}\right)^{\tau_2}\left(\frac{1}{1+\alpha x(\omega^{[2]})}\right)^{1-\tau_2} \\ & \times V\Big(\max\{\min\{\omega_1+\tau_1-\eta,M\},1\},\max\{\min\{\omega_2+\tau_2-\eta,M\},1\}\Big), \end{aligned} \tag{25}$$

and for $x(\omega)$ using

$$x(\omega) = \left[\max\{0,-\zeta(\omega)\}\right]^k, \tag{26}$$

obtained from equation (21). Note that (23) and (24) are equations that are used to construct the system of homotopy equations, while (25) and (26) are simply definitional shorthands for terms that appear in equations (23) and (24). The collection of equations (23) and (24) for all states $\omega \in \{1,\ldots,M\}^2$ can be written more compactly as

$$\mathbf{H}(\mathbf{x},\lambda) = \begin{bmatrix} H_{(1,1)}^{1}(\mathbf{x},\lambda) \\ H_{(2,1)}^{1}(\mathbf{x},\lambda) \\ \vdots \\ H_{(M,M)}^{2}(\mathbf{x},\lambda) \end{bmatrix} = \mathbf{0}, \tag{27}$$

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 $\lambda \in [0,1]$).[12] The equilibrium investment decision $x(\omega)$ in state ω is recovered by substituting the equilibrium slack variable $\zeta(\omega)$ into definition (26).

In general, our approach of replacing a model variable with a slack variable can be taken only if one of the equations in the Zangwill & Garcia (1981) formulation admits a closed-form solution for a model variable (in case of the quality ladder model, we solved equation (21) for the investment decision $x(\omega)$). This is always the case if a model variable is constrained to be above/below a constant, as with the non-negativity constraint in the quality ladder model. However, it is possible that none of the equations in the Zangwill & 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., $x_n(\omega) + x_{-n}(\omega) \leq L(\omega)$, in the quality ladder model (say because firms are competing for

[12] A simplified version of Proposition 3 in Doraszelski & Satterthwaite (2007) establishes that such an equilibrium always exists.

a scare resource).[13] Then in solving an equation corresponding to (16) or (17) for $x_n(\omega)$, one finds that $x_n(\omega) = f(\zeta_n(\omega), x_{-n}(\omega)) = f(\zeta_n(\omega), x_n(\omega^{[2]}))$. That is, the closed-form solution for firm n's policy in state ω, $x_n(\omega)$, is a function of its rival's policy in state ω, $x_{-n}(\omega)$, and thus its own policy in state $\omega^{[2]}$, $x_n(\omega^{[2]})$. In this case, it is impossible to find a closed-form solution for $x_n(\omega)$ as a function of only $\zeta_n(\omega)$ and thus it is impossible to eliminate the model variable $x_n(\omega)$. On the other hand, in this case, the Jacobian of the system formulated using the "pure" version of the Zangwill & 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

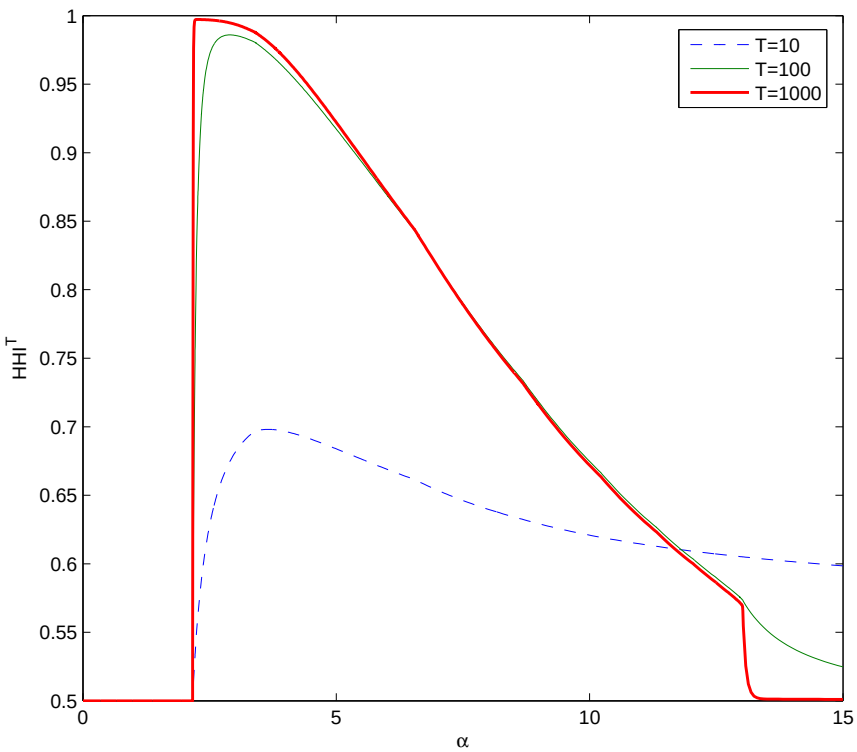

Figure 7: Transient expected Herfindahl index HHI^T at $T \in \{10, 100, 1000\}$ along α with $\delta = 0.7$. Quality ladder model.

We set $k = 2$ in the Zangwill & Garcia (1981) formulation of the quality ladder model. It follows that the system of homotopy equations is continuously differentiable.[14] We explore the equilibrium correspondence by allowing two parameters to vary: the effectiveness of investment α and the rate of depreciation δ. To visualize the equilibrium correspondence

[13] See Besanko et al. (2006) for a more concrete example.

[14] In general, if HOMPACK90 encounters problems when $k = 2$, we recommend setting $k > 2$; this ensures that the system of equations is at least twice continuously differentiable.

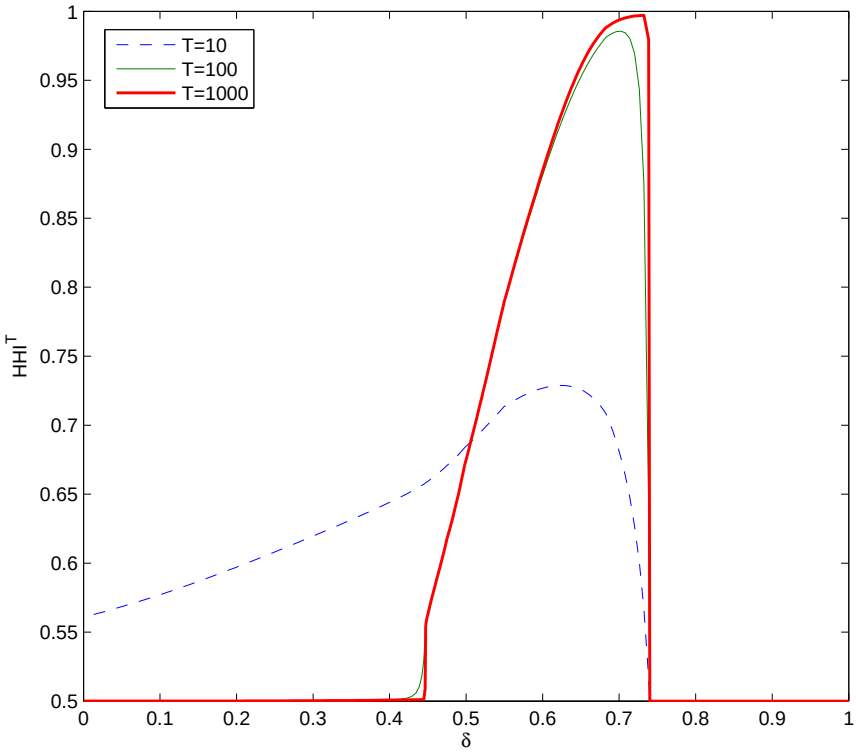

Figure 8: Transient expected Herfindahl index HHI^T at $T \in \{10, 100, 1000\}$ along δ with $\alpha = 3$. Quality ladder model.

we graph the expected Herfindahl index

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

where μ^T is the transient distribution over states in period T, starting from state $(1,1)$ in period 0. We use a transient distribution rather than the limiting distribution as in the learning-by-doing model because there may be several closed communicating classes.[15] When there are multiple closed communicating classes, one cannot compute a single limiting distribution; rather, one must compute a separate limiting distribution for each closed communicating class. So, instead, we compute the transient distribution at various points in time $T \in \{10, 100, 1000\}$. The transient distribution accounts for the probability of reaching any one of the closed communicating classes. In addition, given a discount factor of $\beta = 0.925$ we take a period to be one year; therefore, anything that happens beyond a certain point in time may be considered economically irrelevant.

We present the results in Figures 7 and 8. The industry concentration, both in the short- and in the long-run, is affected by α and δ in non-trivial ways. While the homotopy algorithm computes continuous solution paths, the expected Herfindahl indexes in Figures

[15] A closed communicating class is a subset of states that the industry never leaves once it has entered it.

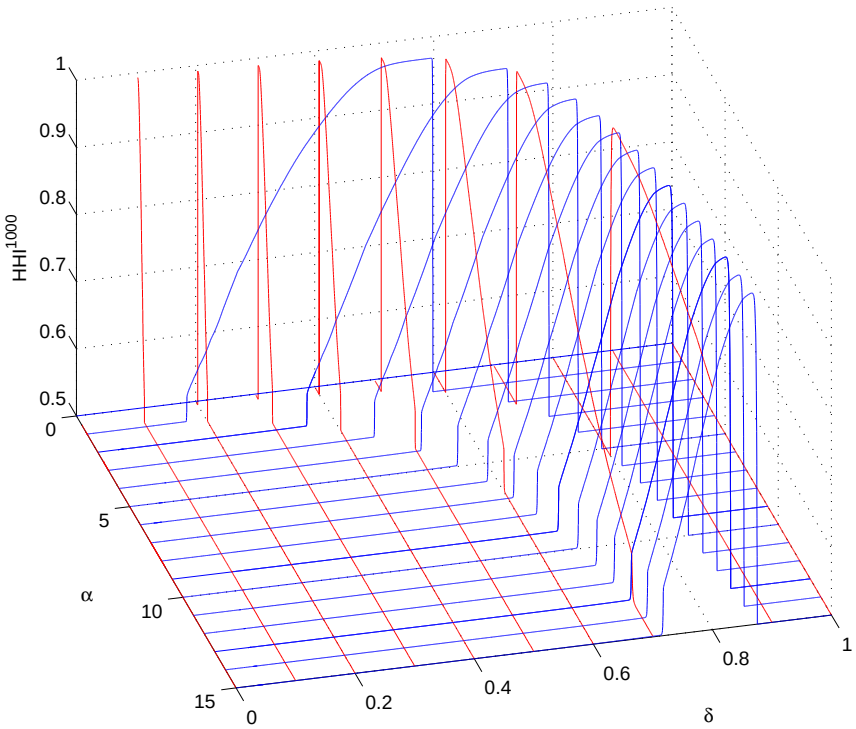

Figure 9: Transient expected Herfindahl index HHI^{1000} along α and δ. Quality ladder model.

7 and 8 appears 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 Figure 7; with $\alpha = 3$ investment drops from positive to zero around $\delta = 0.74$.

Figure 9 illustrates the ability of the homotopy algorithm to criss-cross the parameter space. It combines several slices through the equilibrium correspondence to show how the expected Herfindahl index HHI^{1000} depends jointly on the effectiveness of investment α and the rate of depreciation δ.

Despite our best efforts, we have not uncovered any multiple equilibria in the quality ladder model; this does not necessarily mean that they do not exist.

5.4 Scalability

We use the quality ladder model to assess the scalability of HOMPACK90. We change the number of quality levels M, and thus the number of equations $R = 2M^2$, and adjust the

M	ω^*	#equations	time (h:m:s)	#steps	time/step (s)
9	6	162	0:00:15	931	0.02
18	12	648	0:24:27	7608	0.19
27	18	1458	5:08:27	21457	0.86

Table 7: Scalability. Normal flow algorithm with sparse analytic Jacobian. Quality ladder model.

quality cutoff ω^* accordingly. We trace out the equilibrium correspondence along $\alpha \in [0, 15]$ with $\delta = 0.7$ held fixed.

The results are presented in Table 7. It appears that the total computation time increases approximately as a third-order polynomial in the number of equations R. It is to be expected that the time per step increases in the number of equations since solving the system of linear equations becomes more burdensome. The rate of increase is approximately proportional to $R^{\frac{3}{2}}$. More surprisingly, the number of steps increases in the number of equations. Again the rate of increase is approximately proportional to $R^{\frac{3}{2}}$.

The reason for this latter result is the following. Recall from Section 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 & 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 & 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 less 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 2) 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.[16] 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 would successfully follow a path to its end even if the smoothness requirement were violated. In general, however, it is advisable to formulate the problem such that the smoothness requirement is satisfied (see Sections 5.2 and 5.7).

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). While 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 Section 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

[16] A matrix is singular if its condition number is infinite. A large condition number signifies that a matrix is nearly singular, see pp. 67–70 of Judd (1998).

setting and/or decrease the maximum step size in order to force the homotopy algorithm to remain close to the current 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 $\frac{\partial \mathbf{H}(\mathbf{y})}{\partial \mathbf{y}}$ (see, e.g., equation (7) in case of the ODE based algorithm).[17] 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, while 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 & Schultz 1986) coupled with incomplete LU (ILU) preconditioning. Thus, HOMPACK90 solves $\left(\mathbf{Q}^{-1}\mathbf{A}\right)\mathbf{z} = \left(\mathbf{Q}^{-1}\mathbf{b}\right)$, where $\mathbf{Q}$ is the ILU preconditioner of $\mathbf{A}$. $\mathbf{Q}$ is chosen to make $\left(\mathbf{Q}^{-1}\mathbf{A}\right)$ 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 Section 5.2, constructing a system of equations for the quality ladder model using the "pure" version of the Zangwill & Garcia (1981) formulation yields such a sparse Jacobian. Our proposal is to reduce the size and sparsity of the Jacobian by eliminating variables. Since this proposal may not be applicable or successful in other applications, we discuss in Section 5.7 a number of additional reformulations of the complementary slackness conditions.

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 (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.[18]

[17] The Jacobian $\frac{\partial \mathbf{H}(\mathbf{y})}{\partial \mathbf{y}}$ is a $(N \times (N+1))$ 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 case the ODE-based and normal flow algorithms and the augmented Jacobian algorithm uses a vector that is tangent to the solution path.

[18] We thank Layne Watson for some of these suggestions. We warn the reader that implementing some of

5.7 Other Reformulations of the Complementary Slackness Condition

The user has considerable freedom in formulating the system of equations that characterizes the equilibria of a dynamic stochastic game and some formulations work better than others in some applications. In what follows we present additional reformulations of the complementary slackness conditions. The additional reformulations of the complementary slackness conditions are applicable to problems with simple inequality constraints that force a variable to be above or below a constant.[19]

First reformulation. Recall that in the quality ladder model the investment decision is[20]

$$x(\omega) = \max\left\{0, \frac{-1+\sqrt{\beta\alpha\left(W^1(\omega)-W^0(\omega)\right)}}{\alpha}\right\}$$

if $\left(W^1(\omega)-W^0(\omega)\right) \geq 0$ and $x(\omega)=0$ otherwise. Taken together, the above can be restated as

$$x(\omega) = \frac{-1+\sqrt{\max\left\{1, \beta\alpha\left(W^1(\omega)-W^0(\omega)\right)\right\}}}{\alpha}. \tag{28}$$

It follows that the complementary slackness condition (19) is equivalent to equation (28). However, we cannot simply replace the complementary slackness condition (19) with equation (28) because the max operator in the argument of the square root introduces a kink, thereby violating the smoothness requirement of the homotopy method. But we can eliminate this kink through a change of variables. To see this, let

$$\xi^k(\omega) = \beta\alpha\left(W^1(\omega)-W^0(\omega)\right)-1, \tag{29}$$

where $k \in \mathbb{N}$ is odd.[21] This allows us to restate the investment decision as

$$x(\omega) = \frac{-1+\sqrt{\left[\max\left\{0, \xi(\omega)\right\}\right]^k+1}}{\alpha}. \tag{30}$$

Thus we can replace the complementary slackness condition (19) with equations (29) and (30). By setting $k \geq 3$ we ensure that the system of equations is twice continuously differentiable.

We can further eliminate the model variable $x(\omega)$ by using equation (30) to substitute for it in equations (29) and (22) to construct a system of $2M^2$ equations in the $2M^2$ unknowns $V(1,1), \ldots, V(M,M)$ and $\xi(1,1), \ldots, \xi(M,M)$.

Besides the quality ladder model, this reformulation of the complementary slackness con-

them requires in-depth knowledge of HOMPACK90.

[19] More general formulations for simple inequality constraints other than the non-negativity constraint in the quality ladder model are available from the authors upon request.

[20] We have eliminated the firm indices because we restrict attention to symmetric equilibria.

[21] Note if k is odd, then $\xi(\omega) = (\beta\alpha(W^1(\omega)-W^0(\omega))-1)^{1/k}$, which follows from equation (29), is always well-defined.

dition can also be applied in other models of investment wherein investment is constrained to be above or below a constant and the first-order condition is quadratic.

Second reformulation. Consider an unconstrained version of the quality ladder model. The investment decision is

$$\theta(\omega) = \frac{-1 + \sqrt{\beta\alpha\left(W^1(\omega) - W^0(\omega)\right)}}{\alpha} \tag{31}$$

if $W^1(\omega) - W^0(\omega) \geq 0$ whereas otherwise the problem has no solution. We can recover the investment decision in the constrained version of the model as follows.

$$x(\omega) = \max\left\{0, \theta(\omega)\right\} \tag{32}$$

Hence, the complementary slackness condition (19) is equivalent to equations (31) and (32). Unfortunately, the max operator in equation (32) introduces a kink, thereby violating the smoothness requirement. We address this problem by letting $\phi(\omega)^k$ instead of $\theta(\omega)$ be the investment decision in the unconstrained version of the model, where $k \in \mathbb{N}$ is odd. This yields the following system of equations:

$$[\phi(\omega)]^k - \frac{-1 + \sqrt{\beta\alpha\left(W^1(\omega) - W^0(\omega)\right)}}{\alpha} = 0 \tag{33}$$

$$x(\omega) - \max\{0, [\phi(\omega)]^k\} = 0 \tag{34}$$

We can also eliminate the model variable $x(\omega)$ by using equation (34) to substitute for it in equations (33) and (22) to construct a system of $2M^2$ equations in the $2M^2$ unknowns $V(1,1), \ldots, V(M,M)$ and $\phi(1,1), \ldots, \phi(M,M)$.

In general, this formulation can be used when the first-order condition of the unconstrained problem has a solution. Hence, to present this formulation within the context of the quality ladder model, we have to assume that the investment decision in the unconstrained version of the model is always well defined. Although we have not proven this, a sufficient condition is that the value function is nondecreasing in a firm's state, and this does seem to be the case for the quality ladder model (and many other applications). Moreover, the first-order condition of the unconstrained problem must be expressed in a manner that ensures that its unique solution is the desired optimum, i.e., the second-order condition must hold at the unique solution to the chosen formulation of the first-order condition.

6 Artificial and All-Solutions Homotopies

Below we discuss some other uses of the homotopy method. We first introduce the distinction between natural-parameter homotopies, which trace out an entire path of solutions by varying a parameter of interest, and artificial homotopies, which obtain a solution for a

particular parameterization. Then we present all-solutions homotopies that aim to obtain all solutions to systems of equations with certain properties.

6.1 Artificial Homotopies

While the homotopy method can be used to trace out an entire path of solutions by varying a parameter of interest, it has another important application, namely obtaining a solution to a system of equations for a particular parameterization. Consider the system of equations

$$\mathbf{F}(\mathbf{x}) = \mathbf{0} \tag{35}$$

where $\mathbf{F} : \mathbb{R}^N \to \mathbb{R}^N$ and $\mathbf{0} \in \mathbb{R}^N$ is a vector of zeros and define

$$\mathbf{H}(\mathbf{x}, \lambda) = \lambda \mathbf{F}(\mathbf{x}) + (1 - \lambda)(\mathbf{x} - \mathbf{a}), \tag{36}$$

where $\mathbf{a} \in \mathbb{R}^N$ is a vector. An artificial homotopy traces out a path from $\lambda = 0$, where the solution to $\mathbf{H}(\mathbf{x}, 0) = \mathbf{0}$ is $\mathbf{x} = \mathbf{a}$, to $\lambda = 1$. As $\mathbf{H}(\mathbf{x}, 1) = \mathbf{F}(\mathbf{x})$, the choice of $\mathbf{a}$ does not matter; as long as it reaches $\lambda = 1$, the homotopy algorithm finds a solution to the system of equations (35). So, if the goal is to solve a dynamic stochastic game for a single parameterization (e.g., to obtain an initial condition as in Section 3.3), then HOMPACK90 offers the functionality to do so.

An artificial homotopy tends to be more robust than a homotopy that follows a "natural" parameter of the model. Watson et al. (1997) prove that if $\mathbf{F}$ is twice continuously differentiable and the Jacobian of $\mathbf{F}$ has full rank at any solution to the system of equations (35), then almost all starting points $\mathbf{a}$ will result in a path that has finite length and satisfies regularity at every point. Thus, the artificial homotopy will succeed in tracing out the entire path and, therefore, in solving the system of equations (35) with probability one. In practice, if the homotopy algorithm strays off the solution path to some point $\left(\bar{\mathbf{x}}, \bar{\lambda}\right)$ where $\mathbf{H}(\bar{\lambda}, \bar{\mathbf{x}}) \neq 0$, then it can change the value of the starting point from $\mathbf{a}$ to $\bar{\mathbf{a}}$ such that

$$\bar{\lambda}\mathbf{F}(\bar{\mathbf{x}}) + \left(1 - \bar{\lambda}\right)(\bar{\mathbf{x}} - \bar{\mathbf{a}}) = 0.$$

The homotopy algorithm then returns to the task of tracing out a solution path, starting from point $(\bar{\mathbf{a}}, 0)$, until it finds a solution to the system of equations (35). Note that in changing $\mathbf{a}$ to $\bar{\mathbf{a}}$, the homotopy algorithm simply elects to proceed along a different path, in particular, one that passes through the point $(\bar{\lambda}, \bar{\mathbf{x}})$ to which it has strayed.

Devising a globally convergent algorithm for solving dynamic stochastic games is complicated by the fact that, while we can generally guarantee that the system of equations is twice continuously differentiable, we cannot establish regularity (although, of course, successful usage of natural-parameter homotopies suggests that regularity holds for most – if not all – parameterizations). It remains to be seen how reliable artificial homotopies for

dynamic stochastic games are, especially in comparison to the Pakes & McGuire (1994) algorithm and other nonlinear solvers (see Ferris et al. 2007). Moreover, while an artificial homotopy is extremely robust, it may prove to be less efficient than nonlinear solvers that are based on Newton's method.

6.2 All-Solutions Homotopies

The problem of finding all solutions to a system of equations is largely unresolved in the mathematics literature. Indeed, as already noted, there is no guarantee that the homotopy method finds all the equilibria of a dynamic stochastic game.

In some cases, it is possible to exploit the structure of the system of equations. For example, the system of equations that characterizes the Nash equilibria of a finite game is polynomial (see, e.g., McKelvey & McLennan 1996). For polynomial systems, in turn, there are methods that are sure to find all solutions. These so-called all-solutions homotopies have been implemented in the freely-available software package Gambit (McKelvey, McLennan & Turocy 2006) and used by Bajari, Hong & Ryan (2004) in the context of static games. Judd & Schmedders (2004) use all-solutions homotopies to 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 Chapter 18 of Zangwill & Garcia (1981) for further details on all-solutions homotopies.

While there is little reason to believe that all-solutions homotopies can be extended to general classes of dynamic stochastic games, we emphasize that both natural-parameter and artificial homotopies can and have been used to identify multiple solutions. Our experience suggests that following different parameters (in the case of natural-parameter homotopies) is often a successful strategy. Wolf & Sanders (1996) provide a number of additional suggestions, including using different starting points $\mathbf{a}$ (in the case of artificial homotopies); using a complex homotopy parameter or multiple real homotopy parameters; and allowing the homotopy algorithm to proceed beyond $\lambda = 1$ in the hope that it will bend back and find another solution at $\lambda = 1$. While none of these suggestions is foolproof, in our view, striving to find some solutions is at least a first step toward finding all solutions.

References

Aguirregabiria, V. & Mira, P. (2007), 'Sequential estimation of dynamic discrete games', *Econometrica* **75**(1), 1–54.

Allgower, E. & Georg, K. (1992), 'Continuation and path following', *Acta Numerica* pp. 1–64.

Bajari, P., Benkard, L. & Levin, J. (2007), 'Estimating dynamic models of imperfect competition', *Econometrica* **75**(5), 1331–1370.

Bajari, P., Hong, H., Krainer, J. & Nekipelov, D. (2006), Estimating static models of strategic interactions, Working paper, University of Minnesota, Minneapolis.

Bajari, P., Hong, H. & Ryan, S. (2004), Identification and estimation of discrete games of complete information, Working paper, Duke University, Durham.

Berry, S. & Pakes, A. (2007), 'The pure characteristics demand model', *International Economic Review* **48**(4), 1193–1225.

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

Besanko, D., Doraszelski, U., Kryukov, Y. & Satterthwaite, M. (2007), Learning-by-doing, organizational forgetting, and industry dynamics, Working paper, Harvard University, Cambridge.

Besanko, D., Doraszelski, U., Lu, L. & Satterthwaite, M. (2006), Capacity investment dynamics in oligopolistic industries, Working paper, Northwestern University, Evanston.

Bischof, C., Khademi, P., Mauer, A. & Carle, A. (1996), 'ADIFOR 2.0: Automatic differentiation of Fortran 77 programs', *IEEE Computational Science and Engineering* **3**(3), 18–32.

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

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

Charnes, A., Garcia, C. & Lemke, C. (1977), 'Constructive proofs of theorems relating to f(x)=y, with applications', *Mathematical Programming* **12**(1), 328–343.

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

Doraszelski, U. & Satterthwaite, M. (2007), Computable Markov-perfect industry dynamics: Existence, purification, and multiplicity, Working paper, Harvard University, Cambridge.

Eaves, C. & Schmedders, K. (1999), 'General equilibrium models and homotopy methods', *Journal of Economic Dynamics and Control* **23**(9–10), 1249–1279.

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

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

Garcia, C. & Zangwill, W. (1979), 'An approach to homotopy and degree theory', *Mathematics of Operations Reserach* **4**(4), 390–405.

Golubitsky, M. & Schaeffer, D. (1985), *Singularities and groups in bifurcation theory*, Vol. 1, Springer, New York.

Judd, K. (1998), *Numerical methods in economics*, MIT Press, Cambridge.

Judd, K. & Schmedders, K. (2004), A computational approach to proving uniqueness in dynamic games, Working paper, Hoover Institution, Stanford.

Judd, K., Schmedders, K. & Yeltekin, S. (2002), Optimal rules for patent races, Working paper, Hoover Institution, Stanford.

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

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

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.

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

Pakes, A., Ostrovsky, M. & Berry, S. (2006), 'Simple estimators for the parameters of discrete dynamic games (with entry/exit examples)', *Rand Journal of Economics* **forthcoming**.

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

Saad, Y. & Schultz, M. (1986), 'GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems', *SIAM Journal on Scientific and Statistical Computing* **7**(3), 856–869.

Schmedders, K. (1998), 'Computing equilibria in the general equilibrium model with incomplete asset markets', *Journal of Economic Dynamics and Control* **22**, 1375–1401.

Schmedders, K. (1999), 'A homotopy algorithm and an index theorem for the general equilibrium model with incomplete asset markets', *Journal of Mathematical Economics* **32**(2), 225–241.

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

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

Wolf, D. & Sanders, S. (1996), 'Multiparameter homotopy methods for finding DC operating points of nonlinear circuits', *IEEE Transactions on Circuits and Systems – I: Fundamental Theory and Applications* **43**(10), 824–838.

Zangwill, W. & Garcia, C. (1981), *Pathways to solutions, fixed points, and equilibria*, Prentice Hall, Englewood Cliffs.