Backward Compatibility in Two-Sided Markets*

Richard T. Gretz† † Myongjin Kim‡ ‡ Suman Basuroy§ §

August, 2014
Preliminary and Incomplete
Please do not circulate

Abstract

Often new hardware is backward compatible with software designed for previous generations of hardware. For example, PlayStation 2 can play games designed for PlayStation. Our study contributes to the growing literature on dynamic discrete choice-demand estimation for consumer durables by incorporating backward compatibility into a framework with heterogeneous forward looking consumer who can multi-home (purchase more than one piece of hardware). We apply it using data from the home video game industry on seven consoles and their games spanning two product generations from 1995 - 2005. We use our estimates to obtain the marginal value of console backward compatibility on hardware share. Our results suggest backward compatibility offers hardware firms a significant advantage over non-backward compatible competitors. However, backward compatibility increases the cannibalization of the associated previous generation hardware - an important factor for managers considering profitability throughout the hardware lifespan. Also, we assess the effect of backward compatibility on hardware adoption for different consumer inventories. Interestingly, we find backward compatibility has the greatest effect on non-adopters of the previous generation hardware.

Keywords: two-sided markets, backwards compatibility, dynamic demand, estimation of network effects, videogame industry

JEL: L13, L14, L42, L86

*Richard T. Gretz is the corresponding author.
†Bradley University. Address: Department of Economics, 209 Baker Hall 208, Peoria, IL 61625. E-mail: rgretz@bradley.edu; website: http://www.bradley.edu/academic/departments/economics/faculty
‡University of Oklahoma. Address: Department of Economics, Cate Center Drive, Norman, OK 73972. E-mail: mjkim@ou.edu; website: http://www.myongjinkim.com
§University of Oklahoma. Address: Price College of Business, Division of Marketing & Supply Chain Management 307 W. Brooks, Suite 3 Norman, Oklahoma 73019. E-mail: sumanbasuroy@ou.edu; website: http://faculty-staff.ou.edu/B/Suman.Basuroy-1/
1 Introduction

For many consumer electronics customers choose a product platform to consume the associated product and services, e.g. smart phones, TV set-top boxes or game consoles. Many providers thus target to 'lock' consumers into their platform and make platform switching costly. A particular important time in the product life cycle is the release of a successor product. The hardware provider needs on the one hand a superior product in order to attract new customers but at the same time needs to ensure that not too many existing customer use the opportunity of a new product introduction to switch platforms. Many manufacturers therefore allow their successor product to be backward compatible to the previous generation product, a particular example would be the iPhone which in each new iteration can run the software for the old iPhone but also typically includes new software features. However, manufacturers not always choose to make the product backward compatible. An industry in which we observe both actions is the game console industry. While Sony choose to make the PlayStation 2 backward compatible, Nintendo opted not to do so in the sixth generation hardware (GameCube). This paper therefore studies the rational of allowing backward compatibility in the game console industry.

The main contribution of our paper is to extend Lee (2013) by allowing for backward compatible hardware. This allows us to study the effects of backward compatibility on manufacturers’ profits and consumer welfare. We are able to identify this by keeping track of consumer’s inventory over time. Specifically, consumers purchase the backward compatible hardware even though they do not own the previous generation of the hardware. This offers a source of variation that we use to identify the backward compatibility. We therefore allow that owners of a Playstation 2 derive utility from games available for the Playstation 1. We therefore keep track of the consumers inventory over time.

Our goal in this paper is to examine the benefits and costs to consumers, console makers, and game developers of backward compatibility in a variety of settings. In the counterfactual we examine the effect of backward compatibility by studying the implications of a non-backward compatible PlayStation 2. We find that the it is the non-adopters of Playstation 1 that benefit the most from the backward compatibility. In particular we find that Playstation 2 would have 200,000 fewer adopters a year after it’s introduction if it was not backward compatible. However, this comes with the tradeoff that the Playstation 1 would in fact have had 120,000 more adopters one year after the PlayStation 2 was introduced. Overall, consumer welfare increases through backward compatibility. In another counterfactual we study the effect of forcing all hardware to be backward compatible and the effect on both consumers and firms profit.

This paper is related to the literature on modeling demand for durable goods. Empirically modeling demand for durable goods incorporates several interesting and challenging phenomena. In many industries it is appropriate to assume consumers have heterogeneous preferences, are forward looking and have rational expectations over the expected utility of future products, base their decisions on their current inventory of competing products, and may make repeat purchases. Indeed, not including these realistic aspects of consumer demand can lead to biased estimates (Conlon (2012); Gowrisankaran and Rysman (2009); Lee (2013)). Researchers have recently combined the techniques developed by Berry (1994) and Berry, Levinsohn and Pakes (1995) with the optimal stopping literature pioneered by Rust (1987) to obtain empirical estimations incorporating these aspects of consumer demand. Gowrisankaran and Rysman (2009) estimate demand in the digital camcorder industry allowing for persistent consumer heterogeneity, consumers who can delay
purchases based on rational expectations of the evolution of product characteristics, and repeat purchases by consumers. Lee (2013) extends this methodology with his study of the video game console industry and introduces a new fixed point routine which simultaneously recovers hardware and software utilities.

This paper proceeds as follows: We describe the video game data set in section 2. In section 3, we describe a dynamic model for software and hardware demand and build up a method for incorporating backward compatibility. Then we discuss computation and the specifics of the MPEC techniques employed on the software side of the problem in section 4. We describe instruments and identification strategy in section 5. The dynamic estimation results are presented in section 6. Using the estimated parameters, we recover theoretical policy functions for hypothetical counterfactuals in section 7. Then we conclude in the last section.

2 Data and Industry Background

The game industry has a (surprisingly) long history. Each generation of gaming console hardware is essentially a significant improvement over the previous generation in the hardware quality. Before we provide details on the 5th and 6th generation hardware that are underlying this paper we briefly review the history of gaming consoles that starts in the 1970s. The first generation was created in the early 1970s with Atari being one of the leading producers of arcade video games. One of the first successful games was a simple table tennis game called Pong in 1972. The first generation typically had the game hardwired into microchips. The second generation improved upon this by allowing different software to run on the same hard. Both the first and second generation video game consoles were frequently arcade games that were coin operated and present in many shopping malls. The 3rd generation gaming console arrived in consumers homes in larger numbers than previous generations and arcade games slowly started their decline thereafter. The 3rd generation is represented in particular by the Nintendo Entertainment System (NES) and ended with the discontinuation of the NES. The fourth generation included Mega Drive/Genesis and Super NES and started roughly in 1988. The fourth generation also came along with a continued decline in the popularity of arcades and an increasing penetration of handheld gaming consoles such as the Gameboy. The fifth generation included the Sony Playstation, Sega Saturn and Nintendo 64. The fifth generation is noteworthy for its transition to 3D graphics and use of CDs. Finally, the sixth generation started in the early 2000s and includes the Sega Dreamcast, Sony Playstation 2 and Nintendo Gamecube. Moreover, Microsoft entered the game console industry with the introduction of the Xbox.

Our study examines the fifth and sixth product generation of the home video game industry from May 1995 through October 2005. The data set does not include the next generation of consoles (seventh) which is introduced in November 2005. Our data source is the NPD Group, a market research firm with point of sale data from approximately 65% of U.S. retailers. We have observations on console and game quantity sold and average price over that timeframe. We should note the fifth generation began in 1993 with the introduction of the 3DO and Atari Jaguar, however these consoles were short lived and exited the market rather quickly.

The majority of sales in the fifth product generation are generated by the consoles that we include in our study: Sega Saturn introduced in May 1995, the Sony PlayStation introduced in September 1995, and Nintendo 64 introduced in September 1996. We also include all sixth generation consoles: Sega Dreamcast introduced in September 1999, Sony PlayStation 2 introduced in October 2000, and Nintendo GameCube and Microsoft Xbox both introduced in November
2001. In total, we observe 7 consoles over the timeframe (3 in the fifth generation, 4 in the sixth). However, only 4 consoles survive through 2005 - 1 from the fifth generation and 3 from the sixth generation; Sega Saturn, Nintendo 64, and Sega Dreamcast exit in August 1999, December 2003, and February 2002, respectively.

Each console’s technical characteristics remain unchanged over time, however, new product generations are marked by the introduction of new consoles with a distinct shift in performance ability. For example, the majority of consoles in the 3rd generation feature 32-bit central processing units (CPUs); 128-bit CPUs are the norm for the 4th generation. In general, new video game console generations are introduced every 5 - 7 years. Table 4 summarizes the above points while Table 5 displays descriptive statistics by console, by generation, and in total for the NPD data.

Table 1: Entry and Exit of Fifth and Sixth Generation Consoles

<table>
<thead>
<tr>
<th>Fifth Generation</th>
<th>Processor bit size</th>
<th>Entrance Date</th>
<th>Exit Date (if applicable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sega Saturn</td>
<td>32</td>
<td>May 1995</td>
<td>August 1999</td>
</tr>
<tr>
<td>Sony PlayStation</td>
<td>32</td>
<td>September 1995</td>
<td>–</td>
</tr>
<tr>
<td>Nintendo 64</td>
<td>64</td>
<td>September 1996</td>
<td>December 2003</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sixth Generation</th>
<th>Processor bit size</th>
<th>Entrance Date</th>
<th>Exit Date (if applicable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sega Dreamcast</td>
<td>128</td>
<td>September 1999</td>
<td>February 2002</td>
</tr>
<tr>
<td>Sony PlayStation 2</td>
<td>128</td>
<td>October 2000</td>
<td>–</td>
</tr>
<tr>
<td>Microsoft Xbox</td>
<td>128</td>
<td>November 2001</td>
<td>–</td>
</tr>
<tr>
<td>Nintendo GameCube</td>
<td>128</td>
<td>November 2001</td>
<td>–</td>
</tr>
</tbody>
</table>

Notes: In the market through October 2005

Our study concerns the effect of backward compatibility on adoption and game availability. The fifth and the sixth product generations are ideal candidates for our purposes. Each console in the fifth generation has a successor in the sixth: Sega Saturn - Sega Dreamcast, Sony PlayStation - Sony PlayStation 2, and Nintendo 64 - Nintendo GameCube. However, only the Sony PlayStation 2 had the ability to play games for the original Sony PlayStation. In that sense, Sony PlayStation 2 was backward compatible with Sony PlayStation.

In sum, we observe two console makers (Nintendo and Sega) who introduce consoles in both the fifth and sixth generation and eschew backward compatibility, one console maker (Sony) who introduces consoles in both the fifth and sixth generation and embraces backward compatibility, and one console maker, Microsoft with the Microsoft Xbox, that is a new entrant to the industry in the sixth generation.

These dynamics allow us to consider counterfactuals concerning the effect of backward compatibility on the backward compatible console and its predecessor, incumbent competitors, and new entrants. There are some stylized facts about the industry we should mention before we move forward. First, there is significant consumer heterogeneity around preference for gaming. Second, 

1As noted in Lee (2013), 20% of game players account for 75% of all video game usage.
Table 2: Descriptive Statistics on Monthly Console and Game Prices and Quantities

<table>
<thead>
<tr>
<th>Console</th>
<th>Variable</th>
<th>Average</th>
<th>St. Dev.</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sega Saturn</td>
<td>Monthly Console Price</td>
<td>$111.63</td>
<td>$74.44</td>
<td>$20.51</td>
<td>$262.77</td>
</tr>
<tr>
<td></td>
<td>Monthly Console Quantity</td>
<td>26563.9</td>
<td>64192.6</td>
<td>335.0</td>
<td>449975.0</td>
</tr>
<tr>
<td></td>
<td>Average Monthly Game Price</td>
<td>$15.68</td>
<td>$9.91</td>
<td>$0.30</td>
<td>$51.67</td>
</tr>
<tr>
<td></td>
<td>Average Monthly Game Quantity</td>
<td>1044.5</td>
<td>2588.1</td>
<td>1.0</td>
<td>77030.0</td>
</tr>
<tr>
<td>Sony PlayStation</td>
<td>Monthly Console Price</td>
<td>$69.96</td>
<td>$44.42</td>
<td>$13.64</td>
<td>$126.62</td>
</tr>
<tr>
<td></td>
<td>Monthly Console Quantity</td>
<td>247139.6</td>
<td>372862.7</td>
<td>200.0</td>
<td>2608893.0</td>
</tr>
<tr>
<td></td>
<td>Average Monthly Game Price</td>
<td>$11.04</td>
<td>$7.10</td>
<td>$0.01</td>
<td>$92.33</td>
</tr>
<tr>
<td></td>
<td>Average Monthly Game Quantity</td>
<td>4356.3</td>
<td>17797.4</td>
<td>1.0</td>
<td>1452720.0</td>
</tr>
<tr>
<td>Nintendo 64</td>
<td>Monthly Console Price</td>
<td>$82.67</td>
<td>$31.87</td>
<td>$30.64</td>
<td>$118.87</td>
</tr>
<tr>
<td></td>
<td>Monthly Console Quantity</td>
<td>133597.5</td>
<td>141149.8</td>
<td>470.0</td>
<td>265134.0</td>
</tr>
<tr>
<td></td>
<td>Average Monthly Game Price</td>
<td>$14.62</td>
<td>$7.10</td>
<td>$2.41</td>
<td>$31.74</td>
</tr>
<tr>
<td></td>
<td>Average Monthly Game Quantity</td>
<td>5494.1</td>
<td>14519.3</td>
<td>9.0</td>
<td>255134.0</td>
</tr>
<tr>
<td>Sony PlayStation 2</td>
<td>Monthly Console Price</td>
<td>$117.18</td>
<td>$38.17</td>
<td>$74.40</td>
<td>$178.19</td>
</tr>
<tr>
<td></td>
<td>Monthly Console Quantity</td>
<td>493117.5</td>
<td>444364.3</td>
<td>187553.0</td>
<td>2686290.0</td>
</tr>
<tr>
<td></td>
<td>Average Monthly Game Price</td>
<td>$13.02</td>
<td>$7.05</td>
<td>$0.01</td>
<td>$68.11</td>
</tr>
<tr>
<td></td>
<td>Average Monthly Game Quantity</td>
<td>9497.4</td>
<td>24242.9</td>
<td>2.0</td>
<td>2053983.0</td>
</tr>
<tr>
<td>Microsoft Xbox</td>
<td>Monthly Console Price</td>
<td>$100.37</td>
<td>$28.70</td>
<td>$74.97</td>
<td>$169.71</td>
</tr>
<tr>
<td></td>
<td>Monthly Console Quantity</td>
<td>277427.0</td>
<td>248033.0</td>
<td>77455.0</td>
<td>1079382.0</td>
</tr>
<tr>
<td></td>
<td>Average Monthly Game Price</td>
<td>$12.66</td>
<td>$7.74</td>
<td>$0.00</td>
<td>$112.58</td>
</tr>
<tr>
<td></td>
<td>Average Monthly Game Quantity</td>
<td>7006.4</td>
<td>29745.4</td>
<td>1.0</td>
<td>1777696.0</td>
</tr>
<tr>
<td>Nintendo GameCube</td>
<td>Monthly Console Price</td>
<td>$69.25</td>
<td>$21.83</td>
<td>$46.45</td>
<td>$113.07</td>
</tr>
<tr>
<td></td>
<td>Monthly Console Quantity</td>
<td>201743.8</td>
<td>231721.4</td>
<td>28435.0</td>
<td>1157122.0</td>
</tr>
<tr>
<td></td>
<td>Average Monthly Game Price</td>
<td>$12.79</td>
<td>$6.46</td>
<td>$0.68</td>
<td>$45.39</td>
</tr>
<tr>
<td></td>
<td>Average Monthly Game Quantity</td>
<td>6504.1</td>
<td>22259.2</td>
<td>2.0</td>
<td>826352.0</td>
</tr>
<tr>
<td>Console Generation</td>
<td>Monthly Console Price</td>
<td>$95.03</td>
<td>$36.35</td>
<td>$30.64</td>
<td>$178.19</td>
</tr>
<tr>
<td></td>
<td>Monthly Console Quantity</td>
<td>305285.0</td>
<td>339423.6</td>
<td>470.0</td>
<td>2686290.0</td>
</tr>
<tr>
<td></td>
<td>Average Monthly Game Price</td>
<td>$13.00</td>
<td>$7.15</td>
<td>$0.00</td>
<td>$112.58</td>
</tr>
<tr>
<td></td>
<td>Average Monthly Game Quantity</td>
<td>8028.9</td>
<td>34889.0</td>
<td>1.0</td>
<td>2053983.0</td>
</tr>
<tr>
<td>Sixth Generation</td>
<td>Monthly Console Price</td>
<td>$84.09</td>
<td>$46.30</td>
<td>$13.64</td>
<td>$262.77</td>
</tr>
<tr>
<td></td>
<td>Monthly Console Quantity</td>
<td>237426.1</td>
<td>331504.2</td>
<td>200.0</td>
<td>2686290.0</td>
</tr>
<tr>
<td></td>
<td>Average Monthly Game Price</td>
<td>$12.57</td>
<td>$7.67</td>
<td>$0.00</td>
<td>$112.58</td>
</tr>
<tr>
<td></td>
<td>Average Monthly Game Quantity</td>
<td>6070.4</td>
<td>27712.1</td>
<td>1.0</td>
<td>2053983.0</td>
</tr>
</tbody>
</table>

Notes: All prices corrected for inflation using the Consumer Price Index (1982 = 100).
game sales are driven by hit titles (Gretz and Basuroy (2014); Lee (2013); Binken and Stremersch (2009)). Similar to movies (Basuroy, Chatterjee and Ravid (2003)), the majority of games released sell relatively little compared to a few superstars or killer applications. Figure 1 shows histograms for each console where y-axis is the number of games and the x-axis is total sales. Notice the histograms each display similar patterns - few games with many units sold, many games with few units sold.

Third, most of the games are exclusive to a single console. This means that gamers may multi-home (i.e. purchase more than one console) to obtain access to titles they cannot get from consoles already in their inventory. Table 3 provides data on the number of games and number of exclusive games released for each console.

Table 3: Installed Base and Number of Games per Console

<table>
<thead>
<tr>
<th>Generation</th>
<th>Console</th>
<th>Console Installed Base</th>
<th>Months on the Market</th>
<th>Number of Games Released for Each Console</th>
<th>Number of Exclusive Games</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fifth Generation</td>
<td>Sega Saturn</td>
<td>1381325</td>
<td>52</td>
<td>251</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>Sony PlayStation</td>
<td>30151032</td>
<td>122</td>
<td>1353</td>
<td>969</td>
</tr>
<tr>
<td></td>
<td>Nintendo 64</td>
<td>17983666</td>
<td>88</td>
<td>290</td>
<td>170</td>
</tr>
<tr>
<td>Sixth Generation</td>
<td>Sega Dreamcast</td>
<td>4007925</td>
<td>30</td>
<td>250</td>
<td>140</td>
</tr>
<tr>
<td></td>
<td>Sony PlayStation 2</td>
<td>30080166</td>
<td>61</td>
<td>1191</td>
<td>520</td>
</tr>
<tr>
<td></td>
<td>Microsoft Xbox</td>
<td>13316494</td>
<td>48</td>
<td>758</td>
<td>221</td>
</tr>
<tr>
<td></td>
<td>Nintendo GameCube</td>
<td>9683704</td>
<td>48</td>
<td>494</td>
<td>124</td>
</tr>
</tbody>
</table>

Notes: Figures based on observations through October 2005. Sony PlayStation, Sony PlayStation 2, Microsoft Xbox, and Nintendo GameCube still on market at the end of the sample timeframe.

Importantly, backward compatibility gives anyone who purchases the next generation console access to all games available on the related previous generation console. This means that purchasers of the previous generation console have access to all their previous generation games and non-purchasers of the previous generation console have access to a large number of (exclusive) games that were not in their previous inventory.

The importance of backward compatibility to the video game industry is up for debate - and is the focus of this study. Anecdotal evidence suggests consumers desire the ability to play previous generation games on new systems. An example from the introduction of the most recent generation of video game consoles, Keith Stuart from the Guardian laments “[i]n the new consoles from Sony and Microsoft are selling in their millions, but customers hoping to play their old games will be disappointed (Stuart (2013)).” Interestingly, similar evidence suggests developers desire backward compatibility as well. Guiseppe Nelva in the industry publication DualSHOCKERS discusses a prominent game developer’s negative reaction upon learning new generation consoles will not be backward compatible and notes “it’s funny to see a prominent game developer so surprised considering the radical difference in architecture between the two generations.”

---

2The developer, Hideki Kamiya, called the lack of backward compatibility for PlayStation 4 malicious (Nelva (2014)).
Figure 1: Total Sale per Console Type
Another example shows the concerns of smaller game producers and developers, “[t]he lack of backwards compatibility really affects more than consumers. It affects the small and independent, developers and publishers as well (Pearman (2013)).” However, console makers may face significant costs, either financially or in crippling the new technology, by including the ability to play previous generation games in their new generation consoles. In a recent Wall Street Journal interview, the head of Microsoft’s interactive entertainment business, Don Mattrick, made this point, “Compatibility between software and hardware is also a challenge. For example, neither the PlayStation 4 nor the Xbox One can play game disks made for their predecessors... Microsoft’s Mattrick says “If you are backwards compatible, you are really backwards (Sherr (2013)).”

3 Model

Our model builds on Lee (2013) and modifies the fixed point routine to incorporate overlapping product generations and backward compatibility. That is, current generation hardware may have the ability to use software designed for previous generations. Also, we employ an MPEC (Mathematical Programming with Equilibrium Constraints) technique derived from by Su and Judd (2012) and Dubé, Fox and Su (2012) on the software side of the problem to speed up computation. Both Su and Judd (2012) and Dubé, Fox and Su (2012) show significant speed improvements with MPEC over the traditional fixed point approach in several settings including logit demand models similar to the one we present below.

3.1 Software demand

Our main contribution lies with the inclusion of multiple product generations of hardware and backward compatibility. It is important to note in this subsection we focus on software designed for a unit of hardware which includes all software originally released specifically to be used with that unit of hardware. For example, PlayStation 2 games were originally designed to be played on PlayStation 2. However, PlayStation 2 is backward compatible and can play games designed for PlayStation.\(^3\) We incorporate backward compatibility in the next subsection. Our goals for this subsection are to find the expected option value for a unit of software designed for hardware \(j\). We employ the same assumption as Lee (2013) where consumers consider software purchase decisions independently (i.e. software are not substitutes for each other).

Several important industry features are incorporated into the software demand side of the problem. First, software must be consumed together with hardware in order to provide utility. In other words, a consumer will only consider purchasing a piece of software if they own the appropriate hardware. Second, consumers face a discrete-choice problem of whether to purchase a piece of software or not. Whereas software is a durable good, consumers never return to the market for a piece of software they have purchased previously. Third, consumers are forward looking and have rational expectations regarding the evolution of software characteristics. That is, they compare the present value of purchasing the unit of software in period \(t\) to the expected value of waiting until the next period to reconsider purchase. We assume consumers have rational expectations over the expected value of waiting until the next period. Fourth, we assume consumers are heterogeneous with respect to their taste for software and their sensitivity to software price.

\(^3\)PlayStation 2 is Sony’s next generation follow up to the PlayStation.
The life-time utility for consumer $i$ who owns hardware $j$ and purchases software $k$ at time $t$ is:

$$u_{j,k,t}^i = \alpha_{sw}^i + \alpha_{sw,c}^i c_{j,k,t} + \alpha_{sw,p}^i P_{j,k,t} + \eta_{j,k,t} + \epsilon_{j,k,t}^i. \quad (3.1)$$

Consumer $i$’s software preference parameter, coefficient on software price, and idiosyncratic utility shock for software $k$ at time $t$ are given by $\alpha_{sw}^i$, $\alpha_{sw,p}^i$, and $\epsilon_{j,k,t}^i$, respectively. Importantly, we assume $\epsilon_{j,k,t}^i$ is IID from the type I extreme value distribution. Software $k$’s observed characteristics, unobserved (to the econometrician) characteristics, and price at time $t$ are given by $c_{j,k,t}$, $\eta_{j,k,t}$, and $P_{j,k,t}$. \(^4\)

Going forward it is useful to distinguish the idiosyncratic utility shock from the rest of the consumer’s life-time utility and rewrite (3.1) as $u_{j,k,t}^i = \zeta_{j,k,t}^i + \epsilon_{j,k,t}^i$ where $\zeta_{j,k,t}^i = \alpha_{sw}^i + \alpha_{sw,c}^i c_{j,k,t} + \alpha_{sw,p}^i P_{j,k,t} + \eta_{j,k,t}$ is interpreted as consumer $i$’s mean life-time utility of purchasing software $k$ at time $t$.

Alternatively, consumer $i$ may choose not to purchase software $k$ at time $t$ and receive utility $u_{j,0,t}^i = \epsilon_{j,0,t}^i$ (i.e. mean life-time utility of not purchasing, $\zeta_{j,0,t}^i$, is normalized to zero). Consumer $i$ who does not purchase software $k$ at time $t$ returns to the market at time $t + 1$ to reconsider the purchase. Therefore, consumer $i$ faces an optimal stopping problem (Rust (1987)) where the consumer has to consider whether or not to make a purchase (i.e. stop coming back to the market). The solution to the optimal stopping problem is described by the policy function

$$W^i(\Omega_t^i, \epsilon_{j,k,t}^i) = \max\{u_{j,k,t}^i, u_{j,0,t}^i + \beta E[W^i(\Omega_{t+1}^i, \epsilon_{j,k,t+1}^i)|\Omega_t^i]\} \quad (3.2)$$

where $\Omega_t^i$ contains all the relevant information (prices, product attributes, etc.) to predict the expected value of returning to the market next period.

It is necessary to make some assumptions about $\Omega_t^i$ in order for the state space to be tractable. Lee (2013) following Gowrisankaran and Rysman (2009) and Melnikov (2013) take the following approach. First, $\Omega_t^i$ evolves according to a first order Markov process \(^5\), $G_j^i(\zeta_{j,k,t+1}^i|\Omega_t^i)$, where $G_j^i$ is consumer and hardware specific (i.e. $G_j^i$ does not differ for software designed for hardware $j$). Second, $\Omega_t^i$ evolves similarly for units of software with the same the mean life-time utility. In other words, $\zeta_{j,k,t}^i$ and the evolution of $\zeta_{j,k,t}^i$ are sufficient statistics for consumers to solve their dynamic optimization problem. \(^6\) This means the state can be described by two variables: whether or not software $k$ was previously purchased and $\zeta_{j,k,t}^i$. This simplifies from including prices and other product attributes in the state space. Lee (2013) also includes month of the year, $m(t)$, as an additional state space variable since video game software and hardware sales are highly seasonal, or formally $G_j^i(\zeta_{j,k,t+1}^i|\zeta_{j,k,t}^i, m(t))$. Third, it is necessary to describe beliefs over the evolution of $\zeta_{j,k,t}^i$. Following Assumption II.2 in Lee (2013), we assume consumers expectations of $\zeta_{j,k,t+1}^i$ are

\(^4\)The coefficient on observed characteristics, $\alpha_{sw,c}$, is assumed to be consumer, time, console, and software independent.

\(^5\)Lee (2013) points out that the first order process may not accurately reflect the underlying process but it is a useful simplification that eases the computational burden of the problem. Further, Hendel and Nevo (2006) show that it might be a reasonable approximation of consumer memory and expectations.

\(^6\)This is similar to the Inclusive Value Sufficiency assumption (Assumption 1) in Gowrisankaran and Rysman (2009).
described as:

\[ \zeta_{j,k,t+1} = \gamma_{j,0} + \gamma_{j,1} \zeta_{j,k,t} + \gamma_{j,2} \zeta_{j,k,t}^2 + \sum_{m=1}^{11} \gamma_{j,m+2} \chi_{m(t)} + v_{j,k,t} \]  (3.3)

where the first term is a constant, the second and third terms are the linear and quadratic effects of \( \zeta_{j,k,t} \), the summation represents month dummies, and the last term is an error.

With these assumptions, we can analytically integrate (3.2) over all \( \varepsilon_{j,k,t} \) to obtain consumer \( i \)'s expected option value of being able to purchase software designed for hardware \( j \) with mean life-time utility \( \zeta_{j,k,t} \) in month \( m(t) \):

\[
EW_{j}^{i}(\zeta_{j,k,t}, m(t)) = \int \varepsilon_{j,k,t} W^{i}(\zeta_{j,k,t}, m(t), \varepsilon_{j,k,t}) f(\varepsilon) \\
= \ln \left( e^{\zeta_{j,k,t}} + e^{\beta E[EW_{j}^{i}(\zeta_{j,k,t+1}, m(t+1)) | \zeta_{j,k,t}, m(t)]} \right)  \tag{3.4}
\]

where \( \beta < 1 \) is a discount parameter and \( E[g] \) is the expectation operator (we leave out the constant of integration since it does not affect consumer purchasing decisions).

### 3.2 Software benefits on hardware \( j \) and backward compatibility

In this subsection we discuss the software available on hardware \( j \) for which the consumer values the option to purchase, we incorporate the benefits of the option to purchase backward compatible software, we describe consumer beliefs and values over future software release, and finally our goal, we define the expected discounted benefit consumer \( i \) obtains from having the option to purchase all relevant current and future software available on hardware \( j \).

Importantly, this benefit depends on the hardware units already in the consumer’s inventory. Consumer’s inventory is given by \( \iota \in I_{1} \cup I_{2} \) where \( I_{1} \) and \( I_{2} \) represent the inventory states of first and second generation hardware, respectively. For our problem, \( I_{1} \equiv \{0, 1\}^{3} \) represents inventory states for N64, PlayStation, and Saturn (0 for not purchased, 1 for purchased) in the first hardware generation and \( I_{2} \equiv \{0, 1\}^{4} \) represents inventory states for Dreamcast, GameCube, PlayStation 2, and Xbox in the second hardware generation. In all, there are 128 (= 2^7) possible inventory states given 7 units of hardware. It is helpful to group hardware in consumer inventory into similar product generations in order to compare with hardware not in the inventory. Let \( \iota_{j} \) and \( \iota_{-j} \) be the hardware in inventory \( \iota \) in the same and different generation as hardware \( j \), respectively.

We assume that consumers value the option to purchase two types of software available for hardware \( j \) given their existing inventory. First, consumers value unique software, or formally, let \( K_{j,t}(\iota) \) be the set of software available for hardware \( j \) and not available on any other hardware in inventory \( \iota \). This assumption follows Lee (2013) where consumers only value software options that are not already available in their current inventory.

Second, for any inventory \( \iota \) consumers value software available on hardware \( j \) that is not available on any other same generation hardware in their inventory \( \iota_{j} \) but is available on different product generation hardware in their inventory \( \iota_{-j} \). This is a modification of Lee (2013) that takes into account software competition across product generations. The underlying assumption is that there is sufficient differentiation between hardware product generations for consumers to value the option of purchasing software available for hardware \( j \) that is also available for different product generation hardware in their inventory.
For example, similar games (i.e. same title, software developer, plot, characters, player interaction, etc.) can be released on N64 (first generation hardware) and GameCube (second generation hardware). The game on GameCube is likely more intricate, has more features, and is visually more appealing than the similar release on the previous generation hardware; the game on N64 is likely available for a discount compared to the enhanced game on the next generation hardware. In either case, given the games are not perfect substitutes, we assume consumers value these purchase options when considering their hardware decisions. For convenience, we call this intergenerational software. Formally, let $\mathcal{K}_{j,t}(\ell \rightarrow j)$ be the set of intergenerational software - software available on hardware $j$ at time $t$, in the inventory of different generation hardware $\ell \rightarrow j$, and not in the inventory of same generation hardware.

We want to incorporate the option values of backward compatible software - software designed for a previous generation system that can be used on a current generation system. This includes two more sets of software similar to $\mathcal{K}_{j,t}(t \rightarrow \ell)$ and $\mathcal{K}_{j,t}(\ell \rightarrow j)$ but modified for backward compatibility - essentially we find consumer benefit for the option to purchase software, $k'$, designed for a previous generation hardware, $j'$, which can be used on hardware $j$. For backward compatible hardware $j$ consumers will also value $\mathcal{K}_{j',t}(t \cup j)$ and $\mathcal{K}_{j',t}(\ell \rightarrow j)$. The first term is the set of unique software on hardware $j'$ - software not available in the inventory or hardware $j$. The second term is the set of intergenerational software on hardware $j'$ - software available on hardware $j'$, in the inventory of hardware in a different generation than $j'$ (note this inventory includes $j$), and not in the inventory of same generation hardware as $j'$. Note, $\mathcal{K}_{j',t}(t \cup j)$ and $\mathcal{K}_{j',t}(\ell \rightarrow j)$ are empty sets if hardware $j$ is not backward compatible.

The option values associated with existing software in $\mathcal{K}_{j,t}(t)$, $\mathcal{K}_{j,t}(\ell \rightarrow j)$, $\mathcal{K}_{j',t}(t \cup j)$, and $\mathcal{K}_{j',t}(\ell \rightarrow j)$ make up the total utility received from purchasing hardware $j$. We assume consumers receive utility from software available at time $t$ as well as expected future software releases. We let the utility from software available at time $t$ for hardware $j$, $\Lambda^C_{j,t,t}$, be the sum of the option values at time $t$ where

$$
\Lambda^C_{j,t,t} = \sum_{k \in \mathcal{K}_{j,t}(t)} EW^C_{j,k} (\Omega^C_t) + \pi \left( \sum_{k \in \mathcal{K}_{j,t}(\ell \rightarrow j)} EW^C_{j,k} (\Omega^C_t) \right) 
$$

(3.5a)

for non-backward compatible hardware $j$ and

$$
\Lambda^C_{j,t,t} = \sum_{k \in \mathcal{K}_{j,t}(t)} EW^C_{j,k} (\Omega^C_t) + \phi \Lambda^C_{j',t,t \cup j} + \pi \left( \sum_{k \in \mathcal{K}_{j,t}(\ell \rightarrow j)} EW^C_{j,k} (\Omega^C_t) \right) 
$$

(3.5b)

for backward compatible hardware $j$. Note $\Lambda^C_{j',t,t \cup j}$ in (3.5b) is simply a restatement of (3.5a) for hardware $j'$ including hardware $j$ in the inventory. The first summation in (3.5a) and (3.5b) aggregates the option values for unique software on hardware $j$ while the second summation totals the option values for intergenerational software on hardware $j$. Simply put, backward compatible hardware provides the expected utility of a non-backward compatible hardware plus the option value of using the software of the previous generation. We incorporate the scalars $\pi$ and $\phi$ to

---

Note our model can be easily modified to circumstances where not all the software on $j'$ can be used on hardware $j$. This is not a concern in our dataset so we restrict the theory we display for brevity.
allow for differing effects of intergenerational and backward compatible software, respectively.

The utility from future software releases is

$$F_j \Lambda_{j,t} = \sum_{\tau=1}^{T-t} (\beta T)^\tau \left( \sum_{k \in K^R_{j,t+\tau}(\iota)} EW_{j,k}(\Omega^i_{t+\tau}) \right) + \varpi \left( \sum_{\tau=1}^{T-t} (\beta T)^\tau \left( \sum_{k \in K^R_{j,t+\tau}(t-j)} EW_{j,k}(\Omega^i_{t+\tau}) \right) \right)$$

(3.6a)

for non-backward compatible hardware $j$ and

$$F_j \Lambda'_{j,t} = \sum_{\tau=1}^{T-t} (\beta T)^\tau \left( \sum_{k \in K^R_{j,t+\tau}(\iota)} EW'_{j,k}(\Omega^i_{t+\tau}) \right) + \varpi \left( \sum_{\tau=1}^{T-t} (\beta T)^\tau \left( \sum_{k \in K^R_{j,t+\tau}(t-j)} EW'_{j,k}(\Omega^i_{t+\tau}) \right) \right) + \phi \Lambda'_{j',t,t} \bigcup_j$$

(3.6b)

for backward compatible hardware $j$. $\Lambda'_{j',t,t} \bigcup_j$ is a restatement of (3.6a) for hardware $j'$ including hardware $j$ in the inventory. $K^R_{j,t}(\iota)$ and $K^R_{j,t}(t-j)$ denote the sets of unique software and intergenerational software, respectively, on hardware $j$ released at time $t$. For each of the double sum terms in (3.6a) and (3.6b) the inside sum aggregates the option values of software released in a future period after $t$. This value is discounted by $\beta T$. The outside sum totals the discounted future release option values from time $t + 1$ to some terminal time $T$ (i.e. the date after which no new software is released for hardware $j$). The first double summation is for unique future releases on hardware $j$ while the second is for intergenerational future releases on hardware $j$. Similar to the case of existing software, the expected benefit from future software for backward compatible hardware is the same as for non-backward compatible hardware plus the expected benefit of future software release for a previous generation hardware.

With (3.5a), (3.5b), (3.6a), and (3.6b) we can obtain consumer $i$’s expected present value for purchasing all available software for hardware $j$ at time $t$ given inventory $\iota$, $\Gamma^i_{j,t}$:

$$\Gamma^i_{j,t}(\alpha^i_{sw}, \alpha^i_{sw,p}; \iota) = \Lambda^i_{j,t,t} + E \Lambda^F_{j,t,t}(\Omega^i_{t}).$$

(3.7)

Note, (3.7) depends on the expected value of future software releases. We assume consumers have rational expectations over future software utility and obtain the second term by regressing (3.6a) and (3.6b) on (3.5a) and (3.5b), among other variables, for each consumer. Specifically, we impose the following functional form on consumer beliefs:

$$F_j \Lambda^i_{j,t+1,t} = \varsigma^i_{0,t} \chi_t + \varsigma^i_{j,t,t} \Lambda^i_{j,t,t} + \sum_{m=1}^{11} \varsigma^i_{0,0,m} \chi_m(t) + \varsigma^i_{\Lambda} X_\Lambda + \mu^i_{j,t}$$

(3.8)

where the first term is an inventory specific constant; the second term is the effect of utility from software available at time $t$ for hardware $j$ given inventory $\iota$; the summation indicates month dummies; the fourth term indicates other observables including hardware age, age squared, the number of available games on hardware $j$, and hardware installed base in levels and logs; the last term is an error. Note the coefficient on utility from software available at time $t$, $\varsigma^i_{j,t}$, is independent of inventory - inventory effects are captured by a constant.

We estimate (3.8) separately for each consumer by stacking observations for every possible
inventory where \( j \) can be purchased. We use inventory dummies and restrict coefficients on software utility, month dummies, and other observables to be the same for each inventory. This has the benefit of drastically decreasing the computational burden. Also, there are several inventories where inventory specific coefficients on the predictors cannot be identified due to a lack of observations. This occurs in our data set as a new hardware unit enters the market shortly before another hardware unit leaves; little overlap translates into few observations where the new hardware is in the consumer’s inventory and the exiting hardware is not.

When considering purchasing hardware \( j \) consumers with more hardware in their inventory will likely have a lower value of \( \Gamma_{i,j,t}^j(g) \), ceteris paribus, given only unique software on \( j \) is included along with intergenerational software when calculating \( \Gamma_{i,j,t}^j(g) \). Critically, the value of \( \Gamma_{i,j,t}^j(g) \) for backward compatible hardware depends on whether or not the relevant previous generation hardware is already in the consumer’s inventory. For consumers who do not own the previous generation hardware we calculate \( \Gamma_{i,j,t}^j(g) \) using (3.5b) and (a prediction of) (3.6b). For consumers who own the previous generation hardware we could either calculate \( \Gamma_{i,j,t}^j(g) \) using (3.5b) and (3.6b) or use (3.5a) and (3.6a). Using the latter implies the consumer only considers benefit from hardware \( j \) as hardware \( j' \) is already in their inventory - the underlying assumption is that the consumer will use software \( k' \) on hardware \( j' \) and not on hardware \( j \). The former implies the consumer considers benefits of upgrading from hardware \( j' \) to hardware \( j \) - here we assume the consumer will use software \( k' \) on hardware \( j \) and not on hardware \( j' \). Essentially, using the former implies a consumer who purchases hardware \( j \) eliminates hardware \( j' \) from their inventory and is never on the market for \( j' \) again. We use the former methodology in our estimations below.\(^8\)

This was done for two reasons. First, we believe it is closer approximation to the reality of industries with backward compatible hardware. A main reason a hardware firm invests in backward compatibility is so consumers who own the previous generation hardware can use their existing inventory of software on the new hardware. A common complaint levied by consumers on hardware that eschew backward compatibility is the inability to use their existing software library on the new system.\(^9\) Second, it allows richer hypotheses tests concerning the importance of backward compatibility to consumers who own and those who do not own the previous generation hardware. Otherwise, the value of backward compatibility is essentially restricted to those who do not own the previous generation hardware. For example, a consumer who owns the previous generation system will not take into account backward compatibility when weighing a purchase of backward compatible hardware versus non-backward compatible hardware.

With (3.7) we can relate the benefits of software to the hardware side of the problem and move on to hardware demand.

### 3.3 Hardware demand

Our goal for this subsection is to find the expected option value of being on the market with any inventory and hardware available for purchase. We have several important assumptions similar to the software side of the problem which echo industry features.

First, consumers face a discrete-choice problem of whether to purchase a piece of hardware

---

\(^8\) It is important to note we ran our estimations using both methodologies. Our parameter estimates were not qualitatively different.

\(^9\) “...The biggest complaint is the lack of backward-compatibility with PlayStation 2 and PlayStation 3 games, a possible deal breaker for some...” as stated in the article Upgrade select PS3 games for use on the PS4, http://www.cnet.com, November 12, 2013.
or not. Like software, hardware is a durable good. In each period consumers can purchase any hardware they have not purchased previously. However, we modify this to account for backward compatibility. Consumers who purchase any hardware that is backward compatible with another unit of hardware will never return to the market for either. For example, a consumer who purchases PlayStation 2 has no need to purchase PlayStation given the former can play all the games designed for the latter.

Second, consumers are forward looking and have rational expectations regarding the evolution of hardware characteristics conditional on their current inventory. That is, consumers compare the present value of purchasing hardware and returning to the market in the next period with their new inventory to the expected value of waiting until the next period and returning to the market with the same inventory. We assume consumers have rational expectations over the expected value of waiting until the next period.

Third, we assume consumers are heterogeneous with respect to their sensitivity to hardware price. Finally, for simplicity we assume consumers can purchase at most one unit of hardware per period.

The life-time expected utility consumer \( i \) with inventory \( \imath \) receives from purchasing hardware \( j \) at time \( t \) is:

\[
u_{j,t,\imath}^i = \alpha_{\text{hw},c} c_{j,t} + \alpha_{\text{hw},p} P_{j,t} + \alpha_T \Gamma_{j,t}^i (\alpha_{\text{sw}}, \alpha_{\text{sw},p}; t) + D_{1,2}(t) + D_2(t) + \xi_{j,t} + \varepsilon_{j,t,\imath}^i. \tag{2.9}
\]

Observed product characteristics, prices, and unobserved (to the econometrician) characteristics are given by \( c_{j,t}, P_{j,t}, \) and \( \xi_{j,t} \). Coefficients on observed hardware characteristics, \( \alpha_{\text{hw},c} \), are independent of consumer type while the coefficient on price, \( \alpha_{\text{hw},p} \), is consumer specific. The expected present value for purchasing all available software for hardware \( j \), \( \Gamma_{j,t}^i (\alpha_{\text{sw}}, \alpha_{\text{sw},p}; t) \), is given by (3.7) and depends on consumer \( i \)'s preference for software, sensitivity to software price, and inventory. The coefficient on expected present value for purchasing all available software, \( \alpha_T \), can also be thought of as a scalar which aids in the comparison of hardware and software utility (Train (n.d.); Lee (2013)).

We include dummy variables to assess complementarities of hardware in each generation where \( D_{1,2}(t) = D_{1,2}, \) a constant, if consumer \( i \) has any hardware in their inventory, 0 otherwise. Similarly, \( D_2(t) = D_2 \) if consumer \( i \) has any second generation hardware in their inventory, 0 otherwise. The effect of owning any first generation hardware is given by \( D_{1,2} \) and the effect of owning any second generation hardware is given by \( D_{1,2} + D_2 \); positive values indicate the consumer is more likely to purchase hardware if they already own hardware, negative values indicate they are less likely to purchase hardware. Finally, \( \varepsilon_{j,t,\imath}^i \) is an idiosyncratic utility shock for consumer \( i \) purchasing hardware \( j \) at time \( t \) not currently in inventory \( \imath \). Again, as with the software side of the problem, we assume \( \varepsilon_{j,t,\imath}^i \) is IID from the type I extreme value distribution. Similar to the software side, it is useful to distinguish the idiosyncratic utility shock from the rest of (2.9) and define the mean expected life-time utility consumer \( i \) receives from purchasing hardware \( j \) at time \( t \) given inventory \( \imath \) as

\[
\delta_{j,t,\imath}^i = \alpha_{\text{hw},c} c_{j,t} + \alpha_{\text{hw},p} P_{j,t} + \alpha_T \Gamma_{j,t}^i (\alpha_{\text{sw}}, \alpha_{\text{sw},p}; t) + D_{1,2}(t) + D_2(t) + \xi_{j,t}.
\]

Also, the consumer can choose to not purchase any hardware and receive \( u_{0,t,\imath}^i = \varepsilon_{0,t,\imath}^i \).

Consumer \( i \) faces an optimal stopping problem where the choice is to purchase one unit of hardware not already in their inventory or not purchase any new hardware - in either case, the
consumer returns to the market the next period. With $\varepsilon^i_t$ representing the set of idiosyncratic shocks from hardware purchases or non-purchases in each possible inventory state, the solution to the optimal stopping problem is described by the policy function

$$V^i(\iota, \Omega^i_t, \varepsilon^i_t) = \max \left\{ u^i_{0,t,t} + \beta E[W^i(\iota, \Omega^i_{t+1}, \varepsilon^i_{t+1})|\Omega^i_t], \max_{j \notin \Omega^i_t} \left\{ u^i_{[j],t,t} + \beta E[V^i(\iota \cup \{j\}, \Omega^i_{t+1}, \varepsilon^i_{t+1})|\Omega^i_t]\right\} \right\}$$

(2.10)

and $\{j\}$ represents a hardware unit purchased in period $t$. The first term represents consumer $i$ purchasing the best hardware available in period $t$ and returning to the market in period $t+1$ with their new inventory, the second term represents consumer $i$ not purchasing hardware in period $t$ and returning to the market in period $t+1$ with the same inventory. Similar to the software side, $\Omega^i_t$ contains all the relevant information (prices, product attributes, etc.) to predict the expected value of returning to the market next period.

Again, we use some assumptions to make the state space tractable. $\Omega^i_t$ evolves according to a first order Markov process, $F^i_j(\delta^i_{j,t+1,i} | \Omega^i_t)$, where $F^i_j$ is consumer and hardware specific. We assume $\Omega^i_t$ includes information on hardware mean utilities for all hardware available at time $t$ not included in the consumer’s inventory, $\{\delta^i_{j,t,i}\}_{j \notin \Omega^i_t}$, the set of hardware available for purchase at time $t$, $J_t$, the consumer’s inventory state $\iota$, and month of the year. Specifically, we use the functional form

$$\delta^i_{j,t+1,i} = \vartheta^i_{0,i} \chi_t + \sum_{(j \notin \Omega^i_t) \cap J_t} \vartheta^i_{j,t} \delta^i_{j,t} + \sum_{m=1}^{11} \vartheta^i_{0,0,m} \chi_{m(t)} + \psi^i_{j,t}$$

(2.11)

where the first term is an inventory specific constant effect, the first summation represents the effect of current mean life-time utilities of hardware not in the consumer’s inventory but available for purchase at time $t$, the second summation represents the effect of month dummies, and the last term is an error. Similar to (3.8), we estimate (2.11) by stacking each inventory where hardware $j$ is available for purchase and restricting the coefficients on hardware utility and month dummies to be the same for each inventory. This dramatically reduces the computational burden of the problem. Also, we accommodate situations where hardware options change for a given inventory by zeroing out $\delta^i_{j,t}$ in (2.11) when hardware $j'$ is not available. As with our discussion of (3.8), this occurs in our data set as hardware units enter and leave the market at different points over time (i.e. a consumer with the same inventory may have different hardware choices over time).

With these assumptions, we can analytically integrate (2.10) over all $\varepsilon^i_t$ and obtain our goal for this subsection, consumer $i$’s expected option value of being on the market to purchase available hardware with any inventory $\iota$ at time $t$:

$$EV^i(\iota, \{\delta^i_{j,t,i}\}_{(j \notin \Omega^i_t) \cap (j \in J_t)}, m(t)) = \int_{\varepsilon^i_t} V^i(\iota, \{\delta^i_{j,t,i}\}_{(j \notin \Omega^i_t) \cap (j \in J_t)}, m(t), \varepsilon_t^i) f(\varepsilon)$$

$$= \ln \left( \sum_{(j \notin \Omega^i_t) \cap (j \in J_t)} \left( e^{\delta^i_{j,t,i} + \beta E[V^i(\iota \cup \{j\}_{(j \notin \Omega^i_t) \cap (j \in J_{t+1})}, m(t+1)) | \iota, \{\delta^i_{j,t,i}\}_{(j \notin \Omega^i_t) \cap (j \in J_t)} m(t)]} \right) \right)$$

$$+ e^{\beta E[V^i(\iota, \{\delta^i_{j,t+1,i}\}_{(j \notin \Omega^i_{t+1}) \cap (j \in J_{t+1})}, m(t+1)) | \iota, \{\delta^i_{j,t,i}\}_{(j \notin \Omega^i_t) \cap (j \in J_t)} m(t)]}$$

(2.12)
The summation shows the value of being on the market depends on which hardware units are available for purchase at time $t$ and not currently in a consumer’s inventory. The first term in the summation represents the present value of purchasing console $j$ while the second term in the summation represents the option value of coming back to the market next period with console $j$ in consumer $i$’s inventory. The last term represents the option value of coming back to the market to the market with the same inventory next period.

3.4 State Space

The assumptions outlined in the above subsections drastically reduce the state space of the problem. Similar to Lee (2013), in each period consumer $i$ will condition their hardware and software purchase decisions on hardware utilities, software utilities, their hardware inventory, their software inventory, and the month. As mentioned above, we modify hardware inventory to take into account backward compatibility. Consumers who purchase any hardware that is backward compatible with another unit of hardware will never return to the market for either.

3.5 Other Assumptions

We should note that we simply from several potentially strategic concerns. First, consumers and firms have perfect information regarding hardware exit and entry. Also, exit and entry decisions by hardware and software firms are taken as exogenous and are not strategic. Finally, software firm’s platform decisions (i.e. which hardware does a software firm release their software on) is also exogenous. In the next section we estimate our model by finding the parameter set that minimizes an objective function constructed from these unobservables.

4 Estimation

The goal of this section is to describe the computation algorithm used to rationalize predicted market shares with actual market shares and recover the sets of product unobservables for all software, $\{\eta_{j,k,t}\}$, and all hardware, $\{\xi_{j,t}\}$, for a given set of parameters. We start by discussing our approach to incorporating consumer heterogeneity. Then we lay a general roadmap of the algorithm and proceed to move into subsections which delve into greater detail for portions of the computation. In turn we look at the software side of the problem, the hardware side of the problem, and the interface between the two. When looking at the software and hardware side, we show how the expected value functions in (3.4) and (2.12) are used to construct the probability that a consumer will purchase software and hardware, respectively, in each period. When discussing the interface between the two, we show how market shares on the hardware side effect the available market on the software side. Importantly, we also incorporate backward compatibility into the hardware and software interface. Finally we construct the objective function. The details of the computational algorithm are provided in the Appendix.

4.1 Consumer heterogeneity

In the model consumers are heterogeneous with respect to their taste for software and their sensitivity to software price. We assume consumer tastes for software are independent and normally distributed with mean zero and standard deviation $\sigma_{sw}$.\(^{10}\) We assume price sensitivity is a function of income level and impose the functional form $\alpha_{r,P} = \alpha_{r,P} - \sigma_{r,P} y_i$ where $y_i$ is consumer $i$’s

\(^{10}\)In our estimations we include software specific constants to capture the effect of software quality. This effect cannot be separately identified from mean software preference so we assume the later is zero.
income, $\alpha_{r,P}$ and $\sigma_{r,P}$ are the mean and standard deviation for price sensitivity, and $r = hw, sw$ for hardware and software respectively. We follow Berry, Levinsohn and Pakes (1995) and draw $y^i$ from an independent and log normal distribution with the first and second moments from the March 2001 Current Population Survey.

In practice we discretize the distributions of consumer preferences. Lee (2013) following Judd (1992) and Heiss and Winschel (2008) uses independent (with respect to software preference and price sensitivity) Gauss-Hermite quadrature to determine initial population shares of each type of consumer. We do the same and obtain 9 values of $\alpha^i_{sw} = \alpha^i_{sw,quad} \sigma_{sw}$, where $\alpha^i_{sw,quad}$ is a quadrature point, and 7 values of $y^i$ for sets of $\{\alpha^i_{hw,p}, \alpha^i_{sw,p}\}$ for 63 different consumer types, indexed by $i$, in our estimation.

4.2 The Recovery of Unobservables and The Objective Function

Our goal is to minimize an objective function constructed from the hardware and software unobservables with respect to the model’s parameters. We adopt the approach of Lee (2013) and construct the objective function from the innovations in the unobservables rather than the unobservables themselves (the assumption is that the innovations in unobservables can be described by an AR(1) process with mean zero independent errors). This approach is robust to software and hardware releases being strategically timed and correlations between observable and unobservable characteristics. We define the innovations in unobservables as $\nu_{j,t} = \xi_{j,t} - \rho_{hw} \xi_{j,t-1}$ and $w_{j,k,t} = \eta_{j,k,t} - \rho_{sw} \eta_{j,k,t-1}$, for hardware and software respectively where $\rho_{hw}$ and $\rho_{sw}$ are parameters to be estimated.

First, we find the estimates $\alpha_{hw,c}$ and $\alpha_{sw,c}$ via a regression of $\delta_{j,t} - \alpha_{hw,p} P_{j,t} - \alpha_{sw} \Gamma_{j,t}(t = 0) - D_{1,2} - D_2$ and $\xi_{j,k,t} = \alpha_{sw,p} P_{j,k,t}$ on observed hardware and software characteristics ($c_{j,t}$ and $c_{j,k,t}$), respectively. With estimates of $\alpha_{hw,c}$ and $\alpha_{sw,c}$ we construct innovations in hardware and software unobservables as follows:

$$\nu_{j,t} = \xi_{j,t} - \rho_{hw} \xi_{j,t-1}$$

(4.1)

and

$$w_{j,k,t} = \eta_{j,k,t} - \rho_{sw} \eta_{j,k,t-1}$$

(4.2)

We should note that the inclusion of $(D_{1,2}$ and $D_2)$ alters the interpretation of these coefficients from the original stated in (2.9). $(D_{1,2}$ and $D_2)$ as well as $D_{1,2}$ are included when we calculate (4.1) for generation 1 and 2 hardware, respectively. Therefore, $D_{1,2}$ captures the effect of purchasing first generation hardware while $D_{1,2}$ and $D_2$ (summed) captures the effect of purchasing second generation hardware given a consumer does not own any hardware. Positive effects for $(D_{1,2}$ and $D_2)$ indicate an increased benefit of purchase if no other hardware is owned (i.e. hardware is substitutable) while negative effects for $(D_{1,2}$ and $D_2)$ indicate the opposite (i.e. hardware is complementary).

---

11We use a GMM estimation for both hardware and software separately where the instruments are described in the next subsection.
We construct the conditional moments with instruments for both hardware and software, \( Z_{hw}^{j,t} \) and \( Z_{sw}^{j,k,t} \), discussed below. The conditional moments are:

\[
E[Z_{hw}^{j,t} \nu_{j,t}] = 0, \tag{4.3}
\]

\[
E[(Z_{hw}^{j,t} - Z_{j,t-1}^{hw})(\nu_{j,t} - \nu_{j,t-1})] = 0, \tag{4.4}
\]

\[
E[Z_{sw}^{j,k,t} w_{j,k,t}] = 0, \tag{4.5}
\]

and

\[
E[(Z_{sw}^{j,k,t} - Z_{j,k,t-1}^{sw})(w_{j,k,t} - w_{j,k,t-1})] = 0. \tag{4.6}
\]

We include (4.4) and (4.6) as conditional moments in addition to (4.3) and (4.5) as in Lee (2013) because this approach has been shown to improve estimation results (Arelleno and Bover (1995); Blundell and Bond (1998)). The GMM estimator is:

\[
\arg \min_\theta G(\theta)'(Z'Z)^{-1}G(\theta),
\]

where

\[
Z = \begin{bmatrix}
Z_{hw}^{j,t} & 0 & 0 \\
(Z_{hw}^{j,t} - Z_{j,t-1}^{hw}) & 0 & 0 \\
0 & Z_{sw}^{j,k,t} & 0 \\
0 & 0 & (Z_{sw}^{j,k,t} - Z_{j,k,t-1}^{sw})
\end{bmatrix}
\]

and \( \theta = \{\theta_1, \theta_2, \theta_3\} \).

We do not constrain parameters to be the same for each generation and so the non-linear parameters are given by

\[
\theta_1 = \{\beta, \phi, \sigma_{hw,p}^g, \sigma_{sw,p}^g, \sigma_w^g, \alpha_1^g, \phi, \varpi\} \quad \text{and} \quad \theta_2 = \{\alpha_{hw,p}^g, \rho_{hw}^g, D_{1,2}, D_2, \alpha_{sw,p}^g, \rho_{sw}^g\}
\]

for generation, \( g = 1, 2 \) while the linear parameters are:

\[
\theta_3 = \{\alpha_{hw,c}, \alpha_{sw,c}\}.
\]

They are a function of the non-linear parameters.

Our minimization algorithm proceeds by finding the Gradient of \( G(\theta)'(Z'Z)^{-1}G(\theta) \) using finite differences and minimizing. With this approach, it is useful to distinguish \( \theta_1 \) as it includes all of the parameters that enter the Hardware and Software Interface Loop. We highlight these parameters because the Hardware Software Interface Loop takes a long time to converge - more parameters in this set dramatically increases the time to calculate the Gradient. This is why we manipulated several functions within the loops in order to diminish the number of parameters in this set (i.e. we are able to move \( \alpha_{hw,p}, \alpha_{sw,p}, D_{1,2}, \) and \( D_2 \) into \( \theta_2 \)), though this resulted in slightly

\[\text{12}\] We constrain the search range to reasonable possible values for certain parameters (e.g. positive variances, positive discount factors, etc.) in order to increase the speed with which the algorithm finds a minimum.
different interpretations for $D_{1,2}$, and $D_2$ than originally outlined in (2.9).

4.3 Instruments and Identification

Unless otherwise specified, we include 1 and 2 period lags of each instrument described in this section. For the first difference portions of $Z$ (i.e. $Z_{j,t}^{hw} - Z_{j,t-1}^{hw}$ and $Z_{j,k,t}^{sw} - Z_{j,k,t-1}^{sw}$) the maximum number of lags for any variable is 2.

4.3.1 Identification of the Backward Compatible Software Parameter and Intergenerational Software Parameter

One major innovation in our paper is the inclusion of the backward compatible parameter, $\phi$. The evolution of consumer inventory over time provides a method of identification. The key point is that there are some consumers who purchase the backward compatible hardware with the relevant previous generation hardware in their inventory and others without. The consumers without the previous generation hardware are newly on the market for the previous generation software. Variation in hardware sales for these consumers relative to the option values for previous generation software can be used to identify $\phi$.

In the extreme, if hardware sales do not vary as these consumers have access to different option values for previous generation software then the effect of backward compatibility would be captured by $\phi = 0$. In other words, the option values these consumers have for this software does not affect hardware sales and therefore there is no benefit from backward compatibility. On the other hand, we expect $\phi > 0$ if access to these software options influences hardware sales. That is, they must receive some benefit from the option to purchase backward compatible software. As instruments, we use the current and a 1 period lag of predicted future portions of $\Gamma(g)$ for backward compatible software for hardware with backward compatible capability.

Similarly, variation in hardware sales can be used to identify $\omega$ as consumers have access to different option values of intergenerational software. We expect $\omega = 0$ if option values for intergenerational software do not affect hardware sales, $0 < \omega < 1$ if intergenerational software is a substitute for similar software on the different generation hardware, and $\omega > 1$ if it is a complement. As an instrument, we use the current intergenerational software portions of $\Gamma(g)$ for backward compatible hardware. As the GMM objective function is derived from the unobservables for the mean consumer with no inventory, backward compatible hardware offers an opportunity to identify $\omega$ because similar software is available on both product generations this hardware can access.

4.3.2 Identification and Instruments for Other Parameters

Unless otherwise specified, we interact instruments with product generation dummies to allow for different generational effects.

Variations in sales month to month and as products age are used to identify the effect of the exogenous observables ($\alpha_{hw,c}$ and $\alpha_{sw,c}$). We include month of year dummies and restrict the effect to be the same for all hardware. We also include hardware age, hardware age squared, hardware generation age, hardware generation age squared, and hardware dummies. For software, we include month dummies but allow months to have different effects on software on different hardware. We also include software age, software age squared, and software dummies. The effects of software age are restricted to be the same for all software on the same product generation.

Variation is hardware sales with current and future software quality are used to identify $\alpha_\Gamma$, $\beta$, and $\beta_\Gamma$. A myopic model is rejected if expected future releases of software have no impact on
current hardware sales; a model with no discounting is rejected if hardware sales do not change with the impending release of high quality software. Finally, a model where software options do not affect hardware sales would imply \( \alpha = 0 \). Current and lagged future portions of \( \Gamma(g) \) are used as instruments.

Variation in hardware and software sales as consumer inventory changes overtime provides a means to identify \( \alpha_{sw,p}, \sigma_{sw,p}, \alpha_{hw,p}, \sigma_{hw,p}, \) and \( \sigma_{sw} \). Consider an extreme case where all consumers have the same preference for software and sensitivity to hardware and software price. Then, the introduction of a high quality game will have the same effect on hardware adoption at the beginning of the hardware lifecycle and the end of the hardware lifecycle. Similarly, reductions in hardware and software price will have the same effect on adoption at different times over the lifecycle. On the other hand, if consumers differ in their preference for software and sensitivity to price, we would expect different reactions to the same movements over time. That is, consumers with a higher preference for software and low sensitivity to price are likely early purchasers of hardware. Since hardware and software are durable goods, we expect increases in hardware adoption due to the introduction of high quality software to be less as the hardware ages (i.e. those who have a high preference for software are already on board). Similarly, we expect consumers to be more sensitive to price as hardware and software age as less price sensitive consumers likely purchase earlier.

We use one and two period lagged prices along with current prices as instruments. Current prices are not sufficient as firms may quickly adjust prices due to changes in unobservables. However, the lags are valid instruments as long as firms cannot make accurate forecast of innovations in unobservables.

We use competitor characteristics that affect margins including installed base size, household penetration percentage, and current software option values (equations (3.5a) and (3.5b)) as instruments on hardware and software price as in Lee (2013). It is important to note we do not consider hardware introduced by the same firm as competitors. For example, when constructing these variables PlayStation is not viewed as a competitor to PlayStation 2. This controls for shocks correlated within a firm. For software price, we use hardware installed base size and hardware household penetration percentage for current and different generation competitors.

Further, we use the monthly average Japanese and U.S. exchange rate, both current and lagged as cost shifters as in Berry (1994), Berry, Levinsohn and Pakes (1995), and Nevo (2000). This is valid as most consoles are manufactured in Japan. However, we augment this by including current and lagged producer price indexes for Electronic Computer Manufacturing, Computer Storage Device Manufacturing, and Audio and Video Equipment Manufacturing, Magnetic and Optical Recording Media Manufacturing (MORMM), and Game Software Publishing (GSP) as additional cost side instrument for on the hardware price. We use similar instruments for software price however we only include current and lagged MORMM and GSP producer price indexes. These producer price indexes have been used as valid instruments in other studies of the video game industry (e.g. Gretz and Basuory (2013); Gretz (2010)).

Additionally, to control for any cost effect not captured by age, we instrument for software price in period \( t \) of software released in period \( t_0 \) with the average price of all software \( t - t_0 \) periods old released prior to period \( t_0 \). For example, the instrument for software price in December 1999 for software released in October 1999 is the average 3 month old price of all software released before October 1999. This has the benefit of being uncorrelated with an innovation in the unobservable in period \( t \) by construction (Lee (2013)). Also, we include the average price of software of a different genre on competing hardware in both the same and different generation.
Also, variations in hardware sales as exclusive and non-exclusive software is released can be used to identify $D_{1,2}$ and $D_2$. For example, if the option values of exclusive software have the same impact on hardware sales as the option values of non-exclusive software then all hardware sales must be to consumers who have no other hardware in their inventory. This must be true since software already in a consumer’s inventory is not taken into account when consumers are back on the market for new hardware. This implies very high values of $D_2$ as consumers without any inventory are likely purchasing. Conversely, if exclusive software has a relatively larger impact on hardware sales than non-exclusive software then consumers with existing inventory are likely purchasing. This indicates negative values of $D_2$ as consumers without inventory are less likely to purchase relative to consumers with inventory.

We include the difference between actual and predicted total household penetration of all video game hardware and total household penetration of the later generation as additional moments in the estimation to help identify $D_{1,2}$ and $D_2$. Lee (2013) reports household penetration 44.1 million for all hardware which translates to a penetration rate of 0.4007 by the end of our sample; we find evidence of a penetration rate of 0.33 for the later generation from Nielsen (2012).

Finally, we use one and two period lagged mean hardware ($\delta_{j,t}$) and software utility ($\zeta_{j,k,t}$) to identify $\rho_{hw}$ and $\rho_{sw}$, respectively. We review the results of the estimation below.

5 Estimation Results

The results of the parameter definitions and the results of the minimization routine are displayed in Table 4, Table 5, and Table 6, respectively.

By and large results from Tables 5 and 6 conform to priors and are similar to the results in Lee (2013). The GMM objective is 948.6418. Discount parameters are positive and significant ($\beta > 0$ and $\beta_T > 0$), there is significant heterogeneity in preferences for software ($\sigma_{sw} > 0$), option values for software have a positive impact on hardware demand ($\alpha_T > 0$), prices negative and significantly affect utility ($\alpha_{sw,p} < 0$ and $\alpha_{hw,p} < 0$), and there is little evidence of hardware complementarity or substitutability ($D_{1,2} \approx 0$ and $D_2 \approx 0$). One contrast, we find significant heterogeneity in price preferences, $\sigma_{sw,p}$ and $\sigma_{hw,p}$, (except for software price in the 1st generation) which Lee (2013) could not identify.

We include hardware generation age and its squared term in the estimation as well. It is interesting to note that this effect is increasing at a decreasing rate. This suggests that demand for hardware increases and then falls throughout the product generation lifecycle independent of the age of individual hardware units. This is similar to results in static models (Gretz and Basuroy (2013)) which highlight the importance of generation lifecycle effects on demand.

Importantly, we find that backward compatible software has a positive and significant impact on hardware demand ($\phi > 0$). We should draw attention to the fact that the coefficient on backward compatible software is significantly greater than 1. This means that backward compatible software is more valuable on the next generation hardware than the hardware for which it was originally designed. To see this, consider the policy function described in (A.15) but only allow the consumer three choices: (1) do not purchase hardware, (2) purchase the next generation backward compatible hardware, and (3) purchase the previous generation hardware associated with the backward compatible hardware. Given that $\phi > 1$, the unconditional probability for (2) will increase more than the unconditional probability for (3) given the same increase in the option value for

\[13\text{We obtain a p-value less than .01 using standard T-test.}\]
Table 4: Parameter Definitions

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi$</td>
<td>Backward Compatibility</td>
</tr>
<tr>
<td>$\omega$</td>
<td>Intergenerational Competition</td>
</tr>
<tr>
<td>$\beta$</td>
<td>Discount Factor</td>
</tr>
<tr>
<td>$\beta_\Gamma$</td>
<td>Discount Factor for Expected Future Software Releases</td>
</tr>
<tr>
<td>$\alpha_\Gamma$</td>
<td>Coefficient on the Expected Present Value for the Option of Purchasing Available Software</td>
</tr>
<tr>
<td>$D_{1,2}$</td>
<td>Complementarity/Substitutability Coefficient - Effect on Hardware Demand if No Hardware is in Consumer Inventory</td>
</tr>
<tr>
<td>$D_2$</td>
<td>Complementarity/Substitutability Coefficient - Effect on Hardware Demand if No 2nd Generation Hardware is in Consumer Inventory</td>
</tr>
<tr>
<td>$\sigma_{sw}$</td>
<td>Standard Deviation of Software Preferences</td>
</tr>
<tr>
<td>$\alpha_{sw,p}$</td>
<td>Mean Software Price Effect</td>
</tr>
<tr>
<td>$\sigma_{sw,p}$</td>
<td>Standard Deviation of Software Price Preferences</td>
</tr>
<tr>
<td>$\alpha_{hw,p}$</td>
<td>Mean Hardware Price Effect</td>
</tr>
<tr>
<td>$\sigma_{hw,p}$</td>
<td>Standard Deviation of Hardware Price Preferences</td>
</tr>
<tr>
<td>$\rho_{hw}$</td>
<td>Coefficient on Hardware AR(1) Process</td>
</tr>
<tr>
<td>$\rho_{sw}$</td>
<td>Coefficient on Software AR(1) Process</td>
</tr>
</tbody>
</table>
backward compatible software. We explore this further in the counterfactuals in next section.

Finally, we find evidence that intergenerational software does not affect hardware sales given \( \omega = 0 \). This means that consumers do not tend to purchase titles similar to ones they already own but are available on different generation hardware. This may have important managerial implications given our results suggest consumers do not value next generation software releases when they own similar software compatible with previous generation hardware. Perhaps a counterintuitive result; managers should shy away from releasing on new hardware software similar to what was available on previous generation hardware given that this software does not significantly influence new hardware adoption. Instead, software different from what was available on previous generation hardware is valued.
Table 5: Estimation Results - Hardware

<table>
<thead>
<tr>
<th>Global Variable</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta$</td>
<td>0.44552</td>
<td>(\star\star\star)</td>
</tr>
<tr>
<td></td>
<td>(0.00021)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Non-Linear Hardware Coefficients</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi$</td>
<td>1.99975</td>
<td>(\star\star\star)</td>
</tr>
<tr>
<td></td>
<td>(0.00143)</td>
<td></td>
</tr>
<tr>
<td>$\omega$</td>
<td>0.00074</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(0.00164)</td>
<td></td>
</tr>
<tr>
<td>$D_{1,2}$</td>
<td>1.05273</td>
<td>(351659)</td>
</tr>
<tr>
<td>$D_2$</td>
<td>0.95954</td>
<td>(371400)</td>
</tr>
<tr>
<td>1st Generation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2nd Generation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\alpha_{hw,p}$</td>
<td>-0.03907</td>
<td>(\star\star\star)</td>
</tr>
<tr>
<td></td>
<td>(0.00071)</td>
<td></td>
</tr>
<tr>
<td>$\sigma_{hw,p}$</td>
<td>0.00758</td>
<td>(\star\star\star)</td>
</tr>
<tr>
<td></td>
<td>(0.00006)</td>
<td></td>
</tr>
<tr>
<td>$\alpha_\Gamma$</td>
<td>0.03201</td>
<td>(\star\star\star)</td>
</tr>
<tr>
<td></td>
<td>(0.00004)</td>
<td></td>
</tr>
<tr>
<td>$\beta_\Gamma$</td>
<td>0.07927</td>
<td>(\star\star\star)</td>
</tr>
<tr>
<td></td>
<td>(0.01321)</td>
<td></td>
</tr>
<tr>
<td>$\rho_{hw}$</td>
<td>0.95263</td>
<td>(\star\star\star)</td>
</tr>
<tr>
<td></td>
<td>(0.00369)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Linear Hardware Coefficients</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1st Generation</td>
<td>2nd Generation</td>
</tr>
<tr>
<td>Hardware Age</td>
<td>-0.47862</td>
<td>(\star\star\star)</td>
</tr>
<tr>
<td></td>
<td>(0.00950)</td>
<td></td>
</tr>
<tr>
<td>Hardware Age$^2$</td>
<td>0.00342</td>
<td>(\star\star\star)</td>
</tr>
<tr>
<td></td>
<td>(0.00011)</td>
<td></td>
</tr>
<tr>
<td>Hardware Generation Age</td>
<td>0.53035</td>
<td>(\star\star\star)</td>
</tr>
<tr>
<td></td>
<td>(0.01319)</td>
<td></td>
</tr>
<tr>
<td>Hardware Generation Age$^2$</td>
<td>-0.00391</td>
<td>(\star\star\star)</td>
</tr>
<tr>
<td></td>
<td>(0.00012)</td>
<td></td>
</tr>
</tbody>
</table>

Notes: Standard errors in parentheses
* $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$
Number of Hardware Observations = 435.
Month and hardware dummies included for hardware but not displayed for brevity.
Table 6: Estimation Results - Software

<table>
<thead>
<tr>
<th>Non-Linear Software Coefficients</th>
<th>1st Generation</th>
<th>2nd Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_{sw}$</td>
<td>1.89294***</td>
<td>1.79049***</td>
</tr>
<tr>
<td></td>
<td>(0.00099)</td>
<td>(0.00065)</td>
</tr>
<tr>
<td>$\alpha_{sw,p}$</td>
<td>-0.07491***</td>
<td>-0.07712***</td>
</tr>
<tr>
<td></td>
<td>(0.00001)</td>
<td>(0.00001)</td>
</tr>
<tr>
<td>$\sigma_{sw,p}$</td>
<td>0.00005</td>
<td>0.17497***</td>
</tr>
<tr>
<td></td>
<td>(0.00019)</td>
<td>(0.00011)</td>
</tr>
<tr>
<td>$\rho_{sw}$</td>
<td>0.90584***</td>
<td>0.71960***</td>
</tr>
<tr>
<td></td>
<td>(0.00001)</td>
<td>(0.00004)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Linear Software Coefficients</th>
<th>1st Generation</th>
<th>2nd Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Age</td>
<td>-0.18827***</td>
<td>-0.22442***</td>
</tr>
<tr>
<td></td>
<td>(0.00003)</td>
<td>(0.00001)</td>
</tr>
<tr>
<td>Software Age$^2$</td>
<td>0.00087***</td>
<td>0.00092***</td>
</tr>
<tr>
<td></td>
<td>(0.00000)</td>
<td>(0.00000)</td>
</tr>
</tbody>
</table>

Notes: Standard errors in parentheses
* $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$
Number of Software Observations =160741.
Month dummies for software on each hardware included but not displayed for brevity.
6 Counterfactuals

In this section we recover theoretical parameters for hypothetical situations. We perform three counterfactuals as follow:

1. No backwards compatibility For this counterfactual we study a scenario under which Sony – the only hardware manufacturer with a backward compatible console – looses the backward compatibility of the Playstation 2. We follow Lee and study a ‘partial equilibrium’ in the sense that we do not account for other equilibrium responses such as changes in prices, adjustment of quality or characteristics and available quantity. We assume that the paths for these variables are the same as in the data. What follows now is a description of the computational routine

- in what follows we take prices, product characteristics and homing choices from the data. The following description builds on parts of the ’Computation Overview’ section.

(a) Compute Consumer welfare and profits for the benchmark, i.e. everything as in the data. Given the estimates of the model we can compute consumer welfare and profits for the observations in the data. For the benchmark essentially this was already computed in the estimation step to obtain parameters.
   i. Hardware sales (predicted): Compute the predicted number of consoles sold for the estimated parameters.
   ii. Software sales (predicted): Compute the predicted number of consoles sold for the estimated parameters (as you describe in the computational routing this essentially taking as given the hardware purchases of the consoles – ”distribution of consumers types on each hardware unit in each period”).
   iii. Utility: Given parameter estimates compute the lifetime utility for each consumer type, compute aggregate welfare by summing up utility over consumer types. Essentially this is equation 3.2 in this document for each consumer.

(b) Compute the same as above under the counterfactual assumption, i.e. no backward compatibility. Now this requires a bit more computation. First of all, take the parameter estimates from before but set the parameters related to backward compatibility to zero, i.e. $\phi = 0$.
   i. Hardware sales (predicted): For each consumer given the parameters compute each period the hardware purchasing decision. This given you the distribution of consumer types on each hardware unit in each period. Compute how many consoles of each type would have been sold in total.
   ii. Software sales (predicted): Given the hardware choices compute the software choices consumers would make in each period. Compute how many software titles would have been sold.
   iii. Utility: Once hardware and software purchases are determined one can again compute the utility.
   iv. Having computed sales and utility under the benchmark and the counterfactual we can then compute easily which manufacturer benefitted and how consumer benefitted/suffered. This can then essentially be presented like table 8 in Lee.
• For the counterfactual we need to find new equilibrium decisions of the consumers. We solve for a new consumer equilibrium in the counterfactual, that is given the consumers value function in section 4 and the estimated parameters we determine the decisions the consumers would have made if backward compatibility was absent (essentially a 0 coefficient on all estimates involving backward compatibility). For doing so we take price path and characteristics from the data and compute the consumers value function that tells us for each point in time the purchasing decision a consumer would make. Once we have obtained the decisions for consumers we can as for the benchmark compute welfare and profits. We can then compare the implications of 'no backward compatibility' to the benchmark. Essentially much of the existing code can be 're-used'. Instead of having a parameter guess that is updated over iterations however, one takes the parameters as given. This is essentially what we discussed in our earlier email. That is for each consumer at time t we determine the purchasing decision given prices and characteristics but this might lead to different inventory states and utility/profit.

2. **The effect of backward compatibility on software manufacturers’ profit** The second counterfactual is similar in to the first one in that we consider ‘no backward compatibility’ but in the previous case we took software homing decision as in the data. In this counterfactual however we extend the analysis by allowing software developers to change their homing decision.

This counterfactual requires that we do Lee’s software homing (page 26/27 in his paper). this would allow us to also study the effect on software makers. Equation 18 I belief is separately estimated and thus should not take too much time to do and more importantly should not change the existing results. Since I do not see an easy third counterfactual, adding software homing would at least allow us to study the effect of backward compatibility on software manufacturers and the effect of allowing them to change the homing decision.

3. **Force everyone to be backward compatible** In this counterfactual we examine the effect of forcing all hardware manufacturers to product backward compatible consoles.

7 Conclusion

In this paper we studied the effect of backward compatibility in the game console industry. We extend the existing literature by allowing consumers to derive utility from previously purchased games on newly introduced consoles. We find that backward compatibility has a positive and significant effect on hardware demand. In particular, backward compatible software is more valuable for the following generation hardware than for the current generation hardware.

In the counterfactuals we first study the effect of prohibiting backward compatibility. In a second step we allow software producers to adjust their platform homing decision, taking into account hardware manufacturers choice for backward compatibility. In the final counterfactual, we study the effect of forcing all hardware manufacturers to introduce backward compatible consoles. In all counterfactuals we study the effect on hardware manufacturers and software makers profits, and consumers welfare.
References


Nelva, Giuseppe. 2014. “Hideki Kamiya Finds Out that the PS4 Isn’t Backwards Compatible, Calls it ‘Malicious.’.”


Pearman, Antwand. 2013. “Backwards Compatibility For PS4 and Xbox?”


Sherr, Ian. 2013. “Microsoft and Sony Diverge on Gaming ‘Cloud’.”

Stuart, Keith. 2013. “PS4 and Xbox One: So why aren’t they backwards compatible?”


A Appendix

The goal of this Appendix is to derive the Jacobian of the constraint defined in (A.12) in the software loop. We begin by discussing the sparcity pattern of the Jacobian and the diagonals. Then we discuss the off diagonals and present a formula for obtaining non-zero off diagonals.

A.1 Sparcity and the Diagonals

The Jacobian for (A.12) is relatively sparse since we assume software sales are independent of each other. That is:

$$\frac{\partial \ln(s_{j,k,t})}{\partial \zeta_{j,k,t}} = 0.$$  

Also, for any software unit k the upper triangle of the Jacobian is 0. This is because

$$\frac{\partial \ln(s_{j,k,t})}{\partial \zeta_{j,k,t+1}} = 0.$$  

While this might seem counter intuitive since we are dealing with a dynamic problem and consumers can wait until next period to purchase, recall the consumers are not conditioning their decision on \(\zeta_{j,k,t+1}\) but a forecast of \(\zeta_{j,k,t+1}\) from \(\zeta_{j,k,t}\).

Also note that

$$\frac{\partial c_{p_{j,k,t}}}{\partial \lambda_{j,k,t}} = 0 \quad \text{and} \quad \frac{\partial \lambda_{j,k,t}}{\partial \zeta_{j,k,t}} = 0$$

since the utility one player type obtains from software does not affect the purchase decision of another player type. Given this, it is straightforward to find the diagonals:

$$\frac{\partial \ln(s_{j,k,t})}{\partial \zeta_{j,k,t}} = \frac{1}{s_{j,k,t}} \sum_{i} \left( \frac{\partial c_{p_{j,k,t}}}{\partial \lambda_{j,k,t}} \lambda_{j,k,t} \right) \quad \text{(A.1)}$$

A.2 The Lower Triangle

Our goal for this subsection is to obtain \(\frac{\partial \ln(s_{j,k,t+1})}{\partial \zeta_{j,k,t}}\), \(\frac{\partial \ln(s_{j,k,t+2})}{\partial \zeta_{j,k,t}}\), and so forth. We will start by finding \(\frac{\partial \ln(s_{j,k,t+1})}{\partial \zeta_{j,k,t}}\) and then modify to find the derivative for \(t + 2\) and beyond. Note \(\frac{\partial c_{p_{j,k,t+1}}}{\partial \zeta_{j,k,t}} = 0\) since \(c_{p_{j,k,t+1}}\) is a function of \(\zeta_{j,k,t+1}\) not a forecast of \(\zeta_{j,k,t+1}\) from \(\zeta_{j,k,t}\). Given this, we have to find

$$\frac{\partial \ln(s_{j,k,t+1})}{\partial \zeta_{j,k,t}} = \frac{1}{s_{j,k,t+1}} \sum_{i} \left( \frac{\partial \lambda_{j,k,t+1}}{\partial \zeta_{j,k,t}} c_{p_{j,k,t+1}} \right) \quad \text{(A.2)}$$

The important part of (A.2) is to recognize that

$$\frac{\partial \lambda_{j,k,t+1}}{\partial \zeta_{j,k,t}} = \sum_{i} \frac{\partial \lambda_{j,k,t+1}}{\partial \zeta_{j,k,t}}.$$
In other words, we have to find the derivative of $\lambda_{j,k,t+1}^i$ with respect to the utilities of all consumer types in period $t$. It helps to define

$$\lambda_{j,k,t+1}^i = \frac{fr_{j,k,t+1}^i}{\sum_i fr_{j,k,t+1}^i} \tag{A.3}$$

and

$$fr_{j,k,t+1}^i = (1 - cp_{j,k,t}^i)fr_{j,k,t}^i + NHP_{j,t+1}^i \tag{A.4}$$

Then by the chain rule

$$\frac{\partial \lambda_{j,k,t+1}^i}{\partial \zeta_{j,k,t}^i} = \frac{\partial \lambda_{j,k,t+1}^i}{\partial fr_{j,k,t+1}^i} \frac{\partial fr_{j,k,t+1}^i}{\partial cp_{j,k,t}^i} \frac{\partial cp_{j,k,t}^i}{\partial \zeta_{j,k,t}^i}. \tag{A.5}$$

By taking the appropriate derivatives of (A.3) and (A.4) and substituting $\sum_i fr_{j,k,t+1}^i = \frac{fr_{j,k,t+1}^i}{\lambda_{j,k,t+1}^i}$ we can rewrite

$$\frac{\partial \lambda_{j,k,t+1}^i}{\partial \zeta_{j,k,t}^i} = -fr_{j,k,t}^i (1 - \lambda_{j,k,t+1}^i) \frac{\lambda_{j,k,t+1}^i}{fr_{j,k,t+1}^i} \frac{\partial cp_{j,k,t}^i}{\partial \zeta_{j,k,t}^i} \tag{A.5}$$

Next we have to find $\frac{\partial \lambda_{j,k,t+1}^i}{\partial \zeta_{j,k,t}^i}$. This has an answer different from zero because the share of consumers on the market for software $k$ in period $t + 1$ who are type $i$ will be influenced by the purchase decisions of all consumers in period $t$. For example, if a larger share of consumers of type $i'$ purchase in period $t$ then there are fewer of these types of consumers on the market in time $t + 1$. Consequently, there a greater share of consumers who on the market in period $t + 1$ are type $i$. That is:

$$\frac{\partial \lambda_{j,k,t+1}^i}{\partial \zeta_{j,k,t}^{i'}} = \frac{\partial \lambda_{j,k,t+1}^i}{\partial fr_{j,k,t+1}^{i'}} \frac{\partial fr_{j,k,t+1}^{i'}}{\partial cp_{j,k,t}^{i'}} \frac{\partial cp_{j,k,t}^{i'}}{\partial \zeta_{j,k,t}^{i'}}. \tag{A.6}$$

Similar to the solution for $\frac{\partial \lambda_{j,k,t+1}^i}{\partial \zeta_{j,k,t}^i}$, we take the appropriate derivates of (A.4) and (A.3) and substitute to obtain

$$\frac{\partial \lambda_{j,k,t+1}^i}{\partial \zeta_{j,k,t}^{i'}} = fr_{j,k,t}^{i'} \frac{(\lambda_{j,k,t+1}^i)^2 \partial cp_{j,k,t}^{i'}}{fr_{j,k,t+1}^{i'}} \tag{A.6}$$

We can now substitute (A.5) and (A.6) into (A.2) and simplify to obtain the first goal of this subsection:

$$\frac{\partial \ln(s_{j,k,t+1})}{\partial \zeta_{j,k,t}} = \frac{1}{s_{j,k,t+1}} \sum_i \left( cp_{j,k,t+1}^i (\lambda_{j,k,t+1}^i)^2 \left( \sum_i \left( fr_{j,k,t}^i \frac{\partial cp_{j,k,t}^i}{\partial \zeta_{j,k,t}^i} - \frac{fr_{j,k,t}^i}{\lambda_{j,k,t+1}^i} \frac{\partial cp_{j,k,t}^i}{\partial \zeta_{j,k,t}} \right) \right) \right). \tag{A.7}$$
We will use a similar methodology to find
\[
\frac{\partial \ln(s_{j,k,t+2})}{\partial \zeta_{j,k,t}} = \frac{1}{s_{j,k,t+2}} \sum_i \frac{\partial \lambda^i_{j,k,t+2}}{\partial \zeta^i_{j,k,t}} cp^i_{j,k,t+2}.
\]

The major difference is that
\[
\frac{\partial \lambda^i_{j,k,t+2}}{\partial \zeta^i_{j,k,t}} = \frac{\partial \lambda^i_{j,k,t+2}}{\partial fr^i_{j,k,t+2}} \frac{\partial fr^i_{j,k,t+1}}{\partial fr^i_{j,k,t+2}} \frac{\partial cp^i_{j,k,t}}{\partial fr^i_{j,k,t}} \frac{\partial cp^i_{j,k,t}}{\partial \zeta^i_{j,k,t}}.
\]

From (A.5) it is easy to see that \( \frac{\partial fr^i_{j,k,t+2}}{\partial fr^i_{j,k,t+1}} = 1 - cp^i_{j,k,t+1} \). Incorporating this result and following the same procedure used to find \( \frac{\partial ln(s_{j,k,t+1})}{\partial \zeta_{j,k,t}} \) we obtain
\[
\frac{\partial \ln(s_{j,k,t+2})}{\partial \zeta_{j,k,t}} = \frac{1}{s_{j,k,t+2}} \times \sum_i \left( cp^i_{j,k,t+2} \left( \frac{\lambda^i_{j,k,t+2}}{fr^i_{j,k,t+2}} \right)^2 \left( \sum_i \left( fr^i_{j,k,t+1} \frac{\partial cp^i_{j,k,t}}{\partial \zeta^i_{j,k,t}} \right) - \frac{fr^i_{j,k,t}}{\lambda^i_{j,k,t+2}} (1 - cp^i_{j,k,t+1}) \frac{\partial cp^i_{j,k,t}}{\partial \zeta^i_{j,k,t}} \right) \right). \tag{A.8}
\]

Further, a share of consumers, i.e. type \( i \), on the market in period \( t + 3 \) is
\[
\frac{\partial \lambda^i_{j,k,t+3}}{\partial \zeta^i_{j,k,t}} = \frac{\partial \lambda^i_{j,k,t+3}}{\partial fr^i_{j,k,t+3}} \frac{\partial fr^i_{j,k,t+2}}{\partial fr^i_{j,k,t+3}} \frac{\partial fr^i_{j,k,t+1}}{\partial fr^i_{j,k,t+2}} \frac{\partial cp^i_{j,k,t}}{\partial fr^i_{j,k,t}} \frac{\partial cp^i_{j,k,t}}{\partial \zeta^i_{j,k,t}}.
\]

This pattern is consistent for \( t + 4 \) and so on. Using this, we can define \( \frac{\partial \ln(s_{j,k,T})}{\partial \zeta_{j,k,t}} \) for any \( T > t + 1 \).

It is convenient to let \( A^i_T = fr^i_{j,k,t} \frac{\partial cp^i_{j,k,t}}{\partial \zeta^i_{j,k,t}} \prod_{\tau=t+1}^T (1 - cp^i_{j,k,\tau}) \). Then the solution is
\[
\frac{\partial \ln(s_{j,k,T})}{\partial \zeta_{j,k,t}} = \frac{1}{s_{j,k,T}} \sum_i \left( cp^i_{j,k,T} \frac{\lambda^i_{j,k,T}}{fr^i_{j,k,T}} \left( \sum_i (A^i_T) - \frac{A^i_T}{\lambda^i_{j,k,T}} \right) \right). \tag{A.9}
\]

The elements of the diagonal and lower triangle of the Jacobian are given by (A.1) and (A.9), respectively.

### A.3 Computational Details

The estimation algorithm consists of three major loops. First there is an inner loop each for hardware and software to rationalize predicted market shares with observed market shares. Second, there is an outer loop which governs interaction between the two inner loops until convergence. This loop starts with a guess at the benefit from having the option to purchasing all software (and expected future software) available for a unit of hardware, \( \Gamma^i_{j,t}(\alpha^i_{sw}, \alpha^i_{sw,p}; t_{1,2}) \). This guess is used in the hardware loop to determine the distribution of consumer types on hardware over time. We use this distribution in the software loop to determine utility for each unit of software available for a piece of hardware. Finally, individual software utility is used to construct another guess at \( \Gamma^i_{j,t} \) and we check for convergence; the process begins if convergence does not occur. Once the
outer loop converges, we recover product unobservables software, \( \{ \eta_{j,k,t} \} \), and hardware, \( \{ \xi_{j,t} \} \) and construct the objective function.

### A.3.1 Software Loop

A key contribution of this article lies in our methodology for rationalizing predicted software market share with observed software market share. We employ some techniques developed in Su and Judd (2012) and Dubé, Fox and Su (2012) rather than the fixed point approach of Berry (1994) and Berry, Levinsohn and Pakes (1995). The main reason for this approach is to improve calculation speed. We set up a constraint on the software side of the problem where predicted shares equal observed shares and solve it numerically, while it is beyond the scope of this article to employ the full MPEC approach.\(^{14}\) We provide an analytical Jacobian for the problem which dramatically decreases computation time.

This subsection proceeds by using a guess at software utility for the mean consumer to find the software utility for each consumer type. Next we find the conditional probability each consumer type purchases a unit of software given they have not purchased it previously. Then we find the percentage of each consumer type who has not purchased the unit of software. Multiplying these two together gives the unconditional probability for software purchase for each consumer type. We aggregate for all consumer types - this value is our predicted market share. Finally, we set up our constraint (predicted market shares must equal observed market shares) and find its analytical Jacobian.

First, it is useful to define the mean life-time software utility for the mean consumer as \( \zeta_{j,k,t} = \alpha_{sw} + \alpha_{c,j,k,t} + \alpha_{sw,p}P_{j,k,t} + \eta_{j,k,t} \). Using this along with the definition of mean life-time software utility for consumer \( i \) from (3.1) we can obtain \( \zeta_{i,j,k,t} = \zeta_{j,k,t} + \alpha_{sw} - \alpha_{sw} + (\alpha_{sw,p} - \alpha_{sw,p})P_{j,k,t} \). We can simplify \( \zeta_{i,j,k,t} \) by noting \( \alpha_{sw,p} = \alpha_{sw,p} - \sigma_{sw,p}y_{i} \) and \( \alpha_{sw} = 0 \) (software preference is assumed to be distributed with mean zero). Substituting we obtain:

\[
\zeta_{i,j,k,t} = \zeta_{j,k,t} + \alpha_{sw} - \sigma_{sw,p}y_{i}P_{j,k,t} \tag{A.10}
\]

With (3.4) and the distribution assumption of \( \varepsilon_{i,j,k,t} \) stated above, the conditional probability that consumer \( i \) purchases software \( k \) at time \( t \) given software \( k \) has not been previously purchased is

\[
CP_{j,k,t}^i = \frac{e^{\zeta_{i,j,k,t}}}{e^{\zeta_{i,j,k,t}} + e^{\beta E[EW_{j}^i(\zeta_{j,k,t+1},m(t+1))|\zeta_{i,j,k,t},m(t)]}} \tag{A.11}
\]

where \( \zeta_{i,j,k,t} \) is calculated from (A.10). We obtain an approximation of \( \beta E[EW_{j}^i(\zeta_{i,j,k,t+1},m(t+1))|\zeta_{j,k,t},m(t)] \) using the functional form of consumer beliefs defined in (3.3) and estimate (3.4) using the Galerkin method mentioned in Dubé, Fox and Su (2012) and described in Judd (1992).\(^{15}\)

We need to obtain the distribution of consumer types on board hardware \( j \) who have not yet purchased software \( k \) by time \( t \). Let \( fr_{j,k,t} \) be the fraction of consumers who are type \( i \) and who

\(^{14}\)With 160,741 software observations, the number of constraints became computationally unwieldy when attempting to employ a full MPEC approach (including constraints for market share for all consumer types, value functions for all consumer types, evolution of consumer types on the market for each software unit over time, etc.) for the software side of the problem. Instead we employ a constraint on observed and predicted market share with a more complex Jacobian than would be used in a full MPEC.

\(^{15}\)In practice we obtain a reasonable estimate compared to traditional value function iteration using a grid of 40 Chebyshev nodes to generate a 10 degree polynomial approximation. For our purposes, the Galerkin method increased computation speed by roughly a factor of 2 over traditional value function iteration.
are on board hardware \( j \) at time \( t \) but have not yet purchased software \( k \). Also, let \( NH \pi^{i}_{j,t} \) be the fraction of consumers who are type \( i \) who purchase hardware \( j \) in time \( t \) (i.e. new hardware purchasers). It follows then that 
\[
\frac{N H \pi^{i}_{j,k,t} = (1 - cp^{i}_{j,k,t-1}) \frac{N H \pi^{i}_{j,k,t-1}}{N H \pi^{i}_{j,t}}}
\]
where the first part of the equation are those consumers who were on the market for software \( k \) in the last period but did not purchase and the second part are consumers who are newly on the market for software \( k \) in this period.\(^{16}\) Note that if software \( k \) enters the market in period \( t \) then \( cp^{i}_{j,k,t-1} = 0 \) and the possible purchasers of software \( k \) are all consumers who purchased hardware \( j \) up to and including period \( t \). We then obtain the share of consumers who are type \( i \) on board hardware \( j \) at time \( t \) and have not purchased software \( k \), \( \lambda^{i}_{j,k,t} = \frac{f_{j,k,t}}{\sum_{i} f_{j,k,t}}. \)

We find the predicted market share of software \( k \) by multiplying (A.10) by \( \lambda^{i}_{j,k,t} \) and summing:
\[
s_{j,k,t} = \sum_{i} cp^{i}_{j,k,t} \lambda^{i}_{j,k,t}. \]
Finally, we set the constraint that predicted shares have to equal observed shares. We numerically find \( \zeta_{j,k,t} \) which solves
\[
ln(\text{Observed Share}_{j,k,t}) - ln(s_{j,k,t}) = 0. \tag{A.12}
\]
The Jacobian of (A.12) is necessary to enjoy the increased computational speed of this methodology over the traditional fixed point algorithm.\(^{17}\) This derivation is shown in Appendix A.

### A.3.2 Hardware Loop

The goal of the hardware loop is to find the mean hardware utility, \( \delta_{j,t} \), which rationalizes predicted hardware market shares with observed hardware market shares. We proceed in the usual way where we start with an initial guess at \( \delta_{j,t} \) and iterate until convergence.\(^{18}\) Mean hardware utility is the expected discounted benefit the mean consumer with no hardware in their inventory receives from purchasing hardware \( j \). Formally,
\[
\delta_{j,t} = \alpha_{hw,c} c_{j,t} + \alpha_{hw,p} p_{j,t} + \alpha \tau \Gamma_{j,t}(0, \alpha_{sw,p}; t = 0) + \xi_{j,t} \tag{A.13}
\]
where \( t = 0 \) is defined as the inventory containing no hardware and \( \Gamma_{j,t}(0, \alpha_{sw,p}; t = 0) \) is the expected discounted benefit the mean consumer obtains from having the option to purchase all current and future software available on hardware \( j \). Recall that \( \alpha^{i}_{hw,p} = \alpha_{hw,p} + \sigma_{hw,p} y^{i} \) for any consumer \( i \). Using this along with the (A.13) we can express the expected discounted benefit consumer \( i \) obtains for purchasing hardware \( j \) at time \( t \) with inventory \( \iota \) as
\[
\delta^{i}_{j,t,\iota} = \delta_{j,t} + \sigma_{hw,p} y^{i} p_{j,t} + \alpha \tau (\Gamma^{i}_{j,t}(\alpha^{i}_{sw}, \alpha^{i}_{sw,p}; \iota) - \Gamma_{j,t}(0, \alpha_{sw,p}; t = 0)). \tag{A.14}
\]

The first step in the hardware loop is to use an initial guess at \( \delta_{j,t} \) to find \( \delta^{i}_{j,t,\iota+1,2} \) using (A.14). This approach has two advantages. First, \( \Gamma^{i}_{j,t}(\alpha^{i}_{sw}, \alpha^{i}_{sw,p}; \iota) \) and \( \Gamma_{j,t}(0, \alpha_{sw,p}; t = 0) \) do not change throughout the iterative process and can be found prior to entering the loop. Second, (A.14)

---

\(^{16}\)Note the second part of the equation links the hardware part of the problem to the software side.

\(^{17}\)We found constraining logs rather than levels sped had two benefits. First, for the same solution tolerance, the log approach was generally quicker. Second, the log approach achieved greater accuracy when both approaches were compared to solutions from traditional fixed point iteration.

\(^{18}\)The number of constraints necessary to employ a full MPEC approach was too large given number of observations and possible states. We also explored using an MPEC type technique for the hardware loop similar to our approach for software. However, this proved computationally more burdensome than the traditional fixed point approach given the interconnectedness of hardware purchase decisions - the Jacobian did not have a straightforward analytical solution and was cumbersome to calculate numerically.
depends only on the parameters $\sigma_{hw,p}$ and $\alpha$, not on $\alpha_{hw,p}$, $D_{1,2}(t)$, or $D_2(t)$ which decreases the number of parameters we have to search for with our minimization routine.\footnote{We are able to incorporate the effect of $D_{1,2}(t,\tau)$ and $D_2(t)$ when we recover the hardware unobservables after convergence in $\Gamma^{\tau}_{j,t}$.

\footnote{The latter method proved too computationally unwieldy as the size of the state space became prohibitively large even with a grid of few points for each of the 128 possible inventories, 7 hardware units, 63 consumer types, and 12 months.}}

Second, we obtain forecasts of $\delta_{j,t+1,t}^i$ with (2.11). These are used to construct consumer $i$’s expected option value of being on the market to purchase available hardware with any inventory $\iota$ at time $t$, $EV^i(t,\{\delta_{j,t+1,t}^i\}_{(j\notin\iota)\cap(\iota\in J_t)},m(t))$, given in (2.12). However, we follow Lee (2013) in constructing (2.12) by forecasting 10 sample paths from each $\delta_{j,t+1,t}^i$ and assume all forecasts decay to zero either when hardware exit is observed or 3 months after the sample period ends for hardware still on the market at that time. We do this in place of using standard value function convergence on a discretized state space grid.\footnote{The latter method proved too computationally unwieldy as the size of the state space became prohibitively large even with a grid of few points for each of the 128 possible inventories, 7 hardware units, 63 consumer types, and 12 months.}

Third, we find the unconditional probability that consumer $i$ purchases hardware $j$ at time $t$ given inventory $\iota$ given by

$$s_{j,t}^i = \frac{e^{\Delta_{j,t+1,t}^i}}{e^{\Delta_0,t+1,t} + \sum_{(j\notin\iota)\cap(\iota\in J_t)} e^{\Delta_{j,t+1,t}^i}}$$

(A.15)

where $\Delta_{j,t+1,t}^i$ is the mean utility of purchasing console $j$ and returning to the market next period with hardware $j$ in the inventory and $\Delta_0,t+1,t$ is the value not purchasing any hardware and returning to the market the next period with the same inventory where

$$\Delta_{j,t+1,t}^i = \delta_{j,t+1,t}^i + BE[V^i(t \cup j, \{\delta_{j,t+1,t,\cup j}\}_{(j\notin\iota)\cap(\iota\in J_t+1)},m(t+1)|t,\{\delta_{j,t+1,t}^i\}_{(j\notin\iota)\cap(\iota\in J_t)},m(t)]$$

and

$$\Delta_0,t+1,t = BE[V^i(t,\{\delta_{j,t+1,t}^i\}_{(j\notin\iota)\cap(\iota\in J_t+1)},m(t+1)|t,\{\delta_{j,t+1,t}^i\}_{(j\notin\iota)\cap(\iota\in J_t)},m(t)]$$

respectively.

We can specify the distribution of consumer types on the market with inventory $\iota$ at time $t$ with (A.15) and an initial distribution of consumer types on the market. It helps to define

$$s_{t}^i = \frac{\sum_{(j\notin\iota)\cap(\iota\in J_t)} e^{\Delta_{j,t+1,t}^i}}{e^{\Delta_0,t+1,t} + \sum_{(j\notin\iota)\cap(\iota\in J_t)} e^{\Delta_{j,t+1,t}^i}}$$

as the probability that consumer $i$ at time $t$ with inventory $\iota$ purchases any available hardware on the market that is not currently in their inventory. Given this, at any time $t$, we specify

$$\lambda_{t,t}^i = \frac{(1 - s_{t}^i) \lambda_{t-1,t}^i + \sum_{t'} s_{j,t+1,t}^i \lambda_{t-1,t'}^i}{\sum_{t,t'} (1 - s_{t}^i) \lambda_{t-1,t}^i + \sum_{t'} s_{j,t+1,t}^i \lambda_{t-1,t'}^i}$$

(A.16)
where \( t' \) is any inventory that is one hardware unit away from inventory \( t \). The first part of the numerator are the consumers who were in the inventory last period and did not migrate because no purchase was made; the second part of the numerator are the consumers who migrated to this inventory because they purchased the one hardware unit that was missing from this inventory in the previous period. The denominator represents all the consumers with any inventory who are on the market in period \( t \) and thus normalizes so \( \sum_{i,t} \lambda_{i,t}^j = 1 \) (we assume consumers who purchase all available hardware leave the market).

Finally, with (A.15) and (A.16) we obtain the predicted market share given by \( s_{j,t} = \sum_t \sum_i s_{j,t,i}^i \lambda_{i,t}^j \) and check for convergence with observed market share. We proceed in the usual fashion and iterate until convergence using the traditional Berry (1994) and Berry, Levinsohn and Pakes (1995) fixed point procedure.

### A.3.3 Hardware and Software Interface

This subsection describes the outer loop that iterates back and forth between the hardware and software loops until convergence in \( \Gamma_{j,t}^i(\alpha_{sw}, \alpha_{sw,p}; t) \).

First we use an initial guess at \( \Gamma_{j,t}^i(\alpha_{sw}, \alpha_{sw,p}; t) \) to find the mean hardware utilities that rationalize predicted and observed market shares via the hardware loop. With results from the hardware loop we can obtain the distribution of consumer types on each hardware unit in each period. Essentially we use a function analogous to the numerator of (A.16) where the fraction of consumers who are type \( i \) with inventory \( j \) at time \( t \) is:

\[
\bar{\lambda}_{i,t}^j = (1 - s_{i,t}^j) \bar{\lambda}_{i-1,t}^j + \sum_{t'}^t s_{j(j \not\in t') \cap (j \in t),t',t} s_{j,t',t}^i \bar{\lambda}_{i-1,t'}^j.
\]

The first part is the fraction of consumers who are type \( i \) with the same inventory last period who did not make a purchase and the second is the fraction of consumers who are type \( i \) who migrated to this inventory by making a purchase (they purchased the one hardware unit they were missing). Importantly, in the first period all consumer types begin in \( t = 0 \) (no hardware ownership) with the distribution of the initial fraction of consumers who are type \( i \), \( \bar{\lambda}_{i,0,t=0}^j \) determined by the quadrature method described previously.

We then obtain the fraction of consumers who are type \( i \) on board hardware \( j \) at time \( t \) by summing \( \bar{\lambda}_{i,t}^j \) for all inventories that include \( j \). Formally, we define the cumulative fraction of consumers who are type \( i \) and own hardware \( j \) at time \( t \) as \( H_{j,t}^i = \sum_{t \leq j \in t} \bar{\lambda}_{i,t}^j \). However, we have to adjust it for backward compatibility. For hardware \( j' \), for which hardware \( j \) is backward compatible, we let \( H_{j',t}^i = \sum_{t \leq j' \in t} \bar{\lambda}_{i,t}^j + \sum_{t \leq (j \in t)\cap(j' \notin t)} \bar{\lambda}_{i,t}^j \) where the first part of the equation includes all consumers who are type \( i \) and own hardware \( j' \) at time \( t \) and the second part of the equation includes all consumers who are type \( i \) and own hardware \( j \), but not hardware \( j' \), at time \( t \). The second part of the equation is essentially new consumers to hardware \( j' \) because they did not have that hardware in their inventory prior to their purchase of hardware \( j \). Now these consumers are able to use hardware on \( j' \) because \( j \) is backward compatible.

Importantly, we have to adjust the log of observed share of \( j' \), \( \ln(\text{Observed Share}_{j',k',t}) \), because it will change as the number of consumers who are able to purchase software \( k' \) (software originally made for hardware \( j \)), changes with the addition of new potential consumers from hardware \( j \) owners. We take the observed number of software purchases and divide it by the predicted
number of consumers who are on the market for \( k' \). Without backward compatibility, this value would never change; it would simply be the cumulative number of \( k' \) sales subtracted from the total number of hardware sales (those who own the hardware and who have not yet purchased \( k' \)). With backward compatibility it changes as we have to add in our prediction of new consumers who have access to \( k' \), not because they purchased \( j' \) or have \( j' \) in their inventory, but because they purchase \( j \); this is the second part of \( HP_{j',t}^i \).

Once we have \( HP_{j,t}^i \), along with any adjustments for backward compatibility, we use the software loop to find the mean software utilities that rationalize predicted and observed market share. New hardware purchasers are simply the difference in \( HP_{j,t}^i \) from period to period, \( NHP_{j,t}^i = HP_{j,t}^i - HP_{j,t-1}^i \), and is used in finding \( fr_{j,k,t}^i \) on the software side. Note that, from the software loop, \( fr_{j,k,t}^i = NHP_{j,t}^i \) for all \( k \) units of software for the first period the hardware is on the market.

Finally we calculate the next guess at \( \Gamma_{j,t}^i (\alpha_{sw}^i, \alpha_{sw,p}^i; \iota) \) from mean software utilities found and iterate until convergence.\(^{21}\) These connections between hardware and software necessitate simultaneous estimations of both sides of the problem. Lee (2013) shows that convergence in \( \Gamma_{j,t}^i (\alpha_{sw}^i, \alpha_{sw,p}^i; \iota) \) ensures predicted and observed software and hardware market shares are rationalized simultaneously.

\(^{21}\) We use a tolerance level of \( 1e^{-7} \).