

# Firm Demand Learning and Sticky Prices - A Machine Learning Approach

William Greenall

May 18, 2019

## 1 Introduction

A major assumption in economic models is that firms know the demand function they face. It is almost ubiquitous in macroeconomic models, and deviations from this assumption are excursions from standard microeconomics. If the objective function to be maximised is unknown, it is usually considered a function of some random variable, and appropriate expectations can be taken. In other cases the form of a function is known up to some parameter, and the firm learns about the value of this parameter. Thorough analysis of the general case of this approach is conducted by Aghion et al. (1991). Such assumptions are, however, simplifications for ease of analysis. The aim of this thesis is to relax this assumption, and consider the position of the firm that has no knowledge whatsoever of its demand curve, and to consider the learning process of such a firm. I contend that certain features of firm behaviour in this situation have repercussions for economic observations that are ignored if one assumes that firms have learnt demand completely.

A second major assumption used in most economic models is that observation of the revenue function or demand function is instantaneous, in continuous time; or takes one 'period' when the problem is expressed in discrete time. This implies that we assume away any problems that arise due to the time it takes to learn about any given aspect of the firm's environment. I consider the case of the firm learning about its uncertain environment over time. Optimal learning behaviour, under certain assumptions, will be obtained. Because it takes time to learn about prices, it will become clear that stickiness is the norm: prices must remain where they are for some period of time in order to facilitate learning. This facet of the analysis is absent by construction in situations where learning is modelled as instantaneous.



The structure of the thesis is as follows. I first review some of the literature on sticky pricing and learning models to show the contrast between previous approaches and the one presented here. This includes presentation of some empirical evidence on pricing. I continue with some preliminary mathematical exposition of the tools used, as they may not be familiar to an economics audience. I then present a model of a monopolistic firm learning its demand function whilst maximising revenue. The firm essentially faces a multi-armed bandit problem. The prices it offers are the 'arms' and each arm has a payoff, with individual sales accruing according to a Poisson point process. This is the source of the stochastic nature of the returns for a finite period of time at a given price, and hence the multi-armed bandit formulation. The firm wants to maximise a function that depends on the intensity of this Poisson process, itself a function of price. Being continuum-armed, this bandit problem does not generally have a solution. To handle this, I set up a Bayesian optimisation framework, adapted for this problem, and describe an appropriate algorithm to acquire the sequence of prices. This allows us to construct the learning problem, which becomes one of choosing the appropriate time-to-wait until the next price is set. This model therefore offers direct insight into the learning process, and can also be incorporated into broader macroeconomic analysis to better understand firm behaviour.

## **2 Literature Review: Economics**

In the macroeconomic New Keynesian literature, techniques used to induce sticky prices among firms include those found in Calvo (1983), and Rotemberg (1982). The two techniques are ubiquitous in New Keynesian literature, and a good review of the differences between the two can be found in Lombardo and Vestin (2008). The effectiveness of monetary policy on real economic variables depends on the existence of price and wage stickiness, and there is plenty of empirical literature that argues for the existence of sticky prices. Bils and Klenow (2004) find that prices change on average once every 4.3 months; the implication being that prices are observationally non-sticky in quarterly data. On the other hand, Nakamura and Steinsson (2008) find that prices change about once per 7-11 months when not including short term sales. Kehoe and Midrigan (2015) find further evidence of this, and see that sale prices may differ from so-called regular prices. Regular prices are the prices firms tend to return to between sales. Firms switch to these sale prices relatively often, but changes in regular prices occur around once per year. Given that macroeco-

conomic conditions fluctuate much more frequently than these yearly changes, the empirical evidence appears to justify the use of models that assume stickiness in prices.

Whilst harder to embed in macroeconomic models, another way of modelling price stickiness is known as  $(S, s)$  pricing. This kind of pricing is exhibited in Barro (1972) and more fully in Caplin and Spulber (1987). Here, firms are assumed to face some fixed cost of changing prices. As explained by Stokey (2009), such fixed costs imply an 'inaction' region. The state variable of the system is usually some difference between the actual price set and an appropriately defined price index. This state can vary, but action is only taken once the state variable leaves a certain region. The region is then chosen to optimise some measure of revenue. This balances the gain from constantly changing the level of the control variable and the loss from paying the cost of making those changes. Importantly,  $(S, s)$ -type models have very different implications for the distribution of prices, the extent of price distortion, and the aggregate effects of monetary policy to those of Calvo-style models (Caballero and Engel, 2007).

Beyond these models, another approach that is currently developing in the literature is to invoke rational inattention. First considered by Sims (2003), the concept of rational inattention captures the inability of agents to effectively monitor all signals available to them. This inability is modelled as a constraint on information flow, and agents make decisions based on a strict subset of the information available to them. This is developed further by Maćkowiak and Wiederholt (2009). They show that, with idiosyncratic shocks and aggregate shocks, firms can respond only imperfectly to the large set of signals they receive under perfect information. Hence, they respond less than perfectly to aggregate conditions. This leads to persistence in the response of the price level to aggregate shocks, and hence to nominal rigidities in the sense that the price process under rational inattention fails to track the optimal price process which would appear under full information. However, Maćkowiak and Wiederholt are unable to get specific prices to remain sticky; prices change in every period.

### **3 Literature Review: Operations Research**

Modelling demand as a Poisson process is used in the operations research literature for inventory problems and other dynamic pricing models, but my use of them in economic theory means my thesis consists a novel contribution.

The main example is that of Gallego and van Ryzin (1994). A firm is endowed with a finite inventory, and the firm must price to maximise revenue over a finite horizon. Demand takes the form of a Poisson process; the intensity of the process is a combination of a fixed, known arrival rate and a consumer reserve price distribution. Expected sales (as opposed to customer arrivals) in the time horizon are given by  $\lambda \bar{F}(p)$ ; where  $\lambda$  is the basic intensity and  $\bar{F}$  represents the tail probability of the distribution of consumer reserve prices; the net sales process is thus a marked Poisson process (see Kingman, 1992). This framework is common in most of the operations research literature on Poisson-demand problems. In their paper, Gallego and van Ryzin state that they are only aware of one extant paper at the time using a Poisson demand model, that of Li (1988).

Extending the approach by Gallego and van Ryzin, Aviv and Pazgal (2002) introduce learning about the parameter  $\lambda$  in a finite-horizon inventory problem. The uncertainty in the parameter is used to justify a certainty equivalent heuristic; the firm acts as though the parameterised Bayesian belief over the parameter is the truth. This could plausibly be considered the policy of an unsophisticated seller. The authors show that dynamic pricing can lead to important gains over naive policies when learning is involved. Both the approaches discussed thus far imply that price shifts under an optimal policy are due to the finite-horizon component of the model, since high remaining inventory near the end of the period provides an incentive to sell more quickly and thus lower the price as the alternative revenue from having positive inventory at the end of the period is suboptimal. The infinite horizon framework I develop will not have this feature.

The extension to an infinite-horizon model is taken up by Araman and Caldentey (2006). Here, a finite inventory is sold over an infinite horizon, and the firm learns about the rate parameter  $\lambda \in \{\lambda_1, \lambda_2\}$ , such that  $\lambda_1 < \lambda_2$  whilst maximising discounted average lifetime revenue. Again, the relationship between price and intensity is known up to the parameter. This introduces an exploration-exploitation tradeoff due to the gains of learning whether the product being sold is 'high revenue' or 'low revenue'. The firm here has the choice to either to hold the price and earn the revenues it can (exploitation), or change the price and potentially learn more about the parameter, at the risk of lower revenue (exploration). Comparison between these two possibilities and an outside fixed revenue option lead to the necessity of learning the parameter well in order to acquire good performance.

In a similar vein, Farias and Van Roy (2010) enrich the analysis by al-

lowing the scale parameter  $\lambda$  to take many different values, and describe a new heuristic that they describe as 'decay balancing'. The heuristic uses the same value function formulation as Araman and Caldentey but improves performance by imposing that the policy fulfils an optimality condition from a balance equation between rates of decrease and increase on the value function. Effectively, they impose an Euler equation style property on the approximated value function. The approach I take in this thesis, however, generalises to any form of the function describing the sales intensity  $\lambda\bar{F}(p)$ .

## 4 Mathematical Preliminaries

I now introduce some mathematical tools that I will use in the sequel.

## 5 Poisson Processes

Assume a probability space  $(\Omega, \mathcal{F}, \mathbb{P})$ . A Poisson process is a stochastic process  $X_t$ ,  $t \in \mathbb{R}_+$ , adapted to the filtration  $\mathcal{F}_t$  such that  $\mathcal{F}_1 \subset \mathcal{F}_2 \subset \mathcal{F}_t \subset \mathcal{F}$ . Define the counting function  $N(0, t]$  to be the number of points in the interval  $(0, t]$ . Then, the Poisson process is defined by the following property:

$$\mathbb{P}\{N(t, t + dt) = 0\} = (1 - \lambda) dt + o(dt) \quad (1)$$

$$\mathbb{P}\{N(t, t + dt) = 1\} = \lambda dt + o(dt) \quad (2)$$

$$\mathbb{P}\{N(t, t + dt) \geq 2\} = o(dt) \quad (3)$$

for  $\lambda > 0$ , called the intensity parameter, and where  $o(dt)$  means a term such that  $\lim_{h \rightarrow 0} \frac{o(h)}{h} = 0$ .  $X_t$  then represents the sequence  $\{T_i\}_{i=0}^{N(0,t]}$ , the sequence of point arrival times up to time  $t$ . Then,

$$\mathbb{P}\{N(0, t) = k\} = \frac{e^{-\lambda t} (\lambda t)^k}{k!} \quad (4)$$

i.e.,  $N(0, t] \sim \text{Poisson}(\lambda)$ . An implication of this is that interarrival times  $\tau_i \equiv T_i - T_{i-1} \sim \text{Exponential}(\lambda)$ . This fact will be central to the approach taken in the model. When  $\lambda$  is constant over time, this process is called a *homogeneous* Poisson process. A non-homogeneous Poisson process depends on  $\lambda(t)$  which functions as a density describing the instantaneous rate at which points accrue at any given time  $t$ .

## 6 Gaussian Processes

The exposition here mainly follows that of Brochu et al. (2010). A Gaussian process is a stochastic process  $f(\mathbf{x})$  such that any finite set of elements  $(f(x_1), f(x_2), \dots, f(x_n)) \sim$

$$\mathbb{N} \left( \begin{pmatrix} \mu(x_1) \\ \mu(x_2) \\ \vdots \\ \mu(x_n) \end{pmatrix}, \begin{pmatrix} k(x_1, x_1) & k(x_1, x_2) & \dots & k(x_1, x_n) \\ k(x_2, x_1) & k(x_2, x_2) & \dots & k(x_2, x_n) \\ \vdots & & \ddots & \vdots \\ k(x_n, x_1) & & \dots & k(x_n, x_n) \end{pmatrix} \right)$$

That is, any finite set of points follows a multivariate Gaussian distribution. To represent this, we can write  $f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), K(\mathbf{x}, \mathbf{x}'))$ .  $K$  is known as the kernel (or covariance) function, and describes the correlation between any set of values  $\{f(\mathbf{x}), f(\mathbf{x}')\}$  for any vectors  $\mathbf{x}, \mathbf{x}'$ . It is a well known property of the Gaussian distribution that the marginal distribution of  $f(x)$  will also be Gaussian. The kernel function we will use in this case is the common squared exponential:

$$k(x, x') = \sigma \cdot \exp\left(\frac{-\|x - x'\|^2}{2l^2}\right) \quad (5)$$

for  $\sigma, l > 0$ , where  $l$  is the automatic relevance determination parameter. This describes how much the distance between  $x$  and  $x'$  affects the covariance of the function evaluated at these points. Sample paths from such processes can be seen in Brochu et al. (2010).

Values  $f(\mathbf{x}_0)$  can be considered points of a function evaluated at  $\{\mathbf{x}_0\}$ . These points are called input points. Conditioning on these, one can generate functions that incorporate these observations by generating random variables

$$f(\mathbf{x}^*) \sim \mathbb{N}(\tilde{\mu}(\mathbf{x}^*), K(\mathbf{x}^*, \mathbf{x}^*) - K(\mathbf{x}^*, \mathbf{x}_0)K^{-1}(\mathbf{x}_0, \mathbf{x}_0)K(\mathbf{x}_0, \mathbf{x}^*)) \quad (6)$$

where  $\tilde{\mu}(\mathbf{x}^*) = \mu(\mathbf{x}^*) + K(\mathbf{x}^*, \mathbf{x}_0)K^{-1}(\mathbf{x}_0, \mathbf{x}_0)(f(\mathbf{x}_0) - \mu(\mathbf{x}_0))$ . Here,  $\mathbf{x}^*$  is referred to as a vector of test points. It is simple to extend this to noisy observations, under the assumption of Gaussian noise by adding variance-covariance matrix  $\Sigma_0$ , to  $K(\mathbf{x}_0, \mathbf{x}_0)$ . Having constructed this kernel, generation of Gaussian processes is conducted by generating a vector of standard normal random variables, left-multiplying by the Cholesky decomposition of the kernel matrix, and adding  $\tilde{\mu}$ .

## 7 Copulas

**Definition 7.1.** A copula is a function  $C(U_1, \dots, U_n)$  such that:

$$C(\{U_1, U_2, \dots, U_n\}) = \mathbb{P}\{u_1 \leq U_1, u_2 \leq U_2, \dots, u_n \leq U_n\} \text{ for } \{u_i\} \sim \text{Unif}[0, 1] \quad (7)$$

That is,  $C$  is the joint distribution of a set of uniform random variables, with some covariance structure. Copulas, as shown by Sklar's theorem (Sklar, 1959), can be used to define covariance/dependence structures between sets of random variables with given marginal distributions as follows. If  $X_i$  is distributed according to some measure with corresponding distribution function  $F$ , then

$$F(X_i) \sim \text{Unif}[0, 1] \quad (8)$$

Hence,

$$G(u_1, u_2, \dots, u_n) = C(F_1(x_1), \dots, F_n(x_n)) \quad (9)$$

Where  $G$  is the joint distribution over  $\{u_i\}$ . Intuitively, the sequence  $\{X_i\}$  can have marginal distributions given by  $\{F_i\}$ , and the joint distribution of them be described by  $G$ .

**Definition 7.2.** A Gaussian copula is a copula  $C(u_1, \dots, u_n)$  such that:

$$C(\{U_1, U_2, \dots, U_n\}) = \Phi_\rho(\Phi^{-1}(u_1), \dots, \Phi^{-1}(u_n)) \quad (10)$$

where  $\Phi_\Sigma$  denotes the cdf of the standard Normal distribution, with covariance matrix  $\Sigma$ .

Now, we can use (8) to write:

$$C(\mathbf{U}) = \Phi_\Sigma(\Phi^{-1}(F(x_1)), \dots, \Phi^{-1}(F(x_n))) \quad (11)$$

Now, the joint distribution of  $\mathbf{x}$  is Gaussian, with covariance matrix  $\Sigma$ , but the marginal distributions of  $\{x_i\}$  are those implied by  $\{F_i\}$ . These tools then can be used to construct a Gaussian Copula process, which I define below. Intuitively, these will allow us to construct covariance structures using the kernel in a Gaussian process, whilst allowing for marginal distributions other than Gaussian. Such processes were first developed by Wilson and Ghahramani (2010), and I write the definition here.

**Definition 7.3.** A Gaussian copula process is a stochastic process  $\{X_t\}$ ,  $t \in \mathbb{R}$ , with marginal distribution functions  $F_t(X_t) = u_t \sim \text{Unif}[0, 1]$ , and for any set of variables  $\{X_{t_i}\}_{i=1}^n$ ,

$$\mathbb{P}\{x_{t_1} \leq A_1, x_{t_2} \leq A_2, \dots, x_{t_n} \leq A_n\} = \Phi_{\Sigma} [\Phi^{-1}(A_1), \dots, \Phi^{-1}(A_n)] \quad (12)$$

The explanation given in Wilson and Ghahramani (2010) as to the intuition around Copula processes is particularly clear and succinct, so I reproduce it here as a quote:

”Imagine choosing a covariance function, and then drawing a sample function at some finite number of points from a Gaussian process. The result is a sample from a collection of Gaussian random variables, with a dependency structure encoded by the specified covariance function. Now, suppose we transform each of these values through a univariate Gaussian cdf, such that we have a sample from a collection of uniform random variables. These uniform random variables also have this underlying Gaussian process dependency structure. One might call the resulting values a draw from a Gaussian-Uniform Process. We could subsequently put these values through an inverse beta cdf, to obtain a draw from what could be called a Gaussian-Beta Process: the values would be a sample from beta random variables, again with an underlying Gaussian process dependency structure.”

This tool will allow me to cleanly adapt the algorithm used in the sequel for the purposes described here. Figure 1 presents an example of a Gaussian Process and its implied Uniform-Gaussian Process. Figure 2 presents the implied marginally Gamma-distributed Gaussian Copula Process parameterised by the hyperparameters displayed in the right-hand image.

## 8 The Model

The aim of this thesis is to construct an environment in which I can examine the effects of having to learn the demand function on firm pricing behaviour. The setup I will use also takes into account the fact that learning about the demand curve implies observing sales as they occur. This means that information about sales is not acquired instantaneously.

I consider a monopolistic firm, facing no marginal cost, whose aim is to maximise lifetime discounted revenue. It does so by choosing  $p \in \mathcal{P} \subset \mathbb{R}_+$ ;

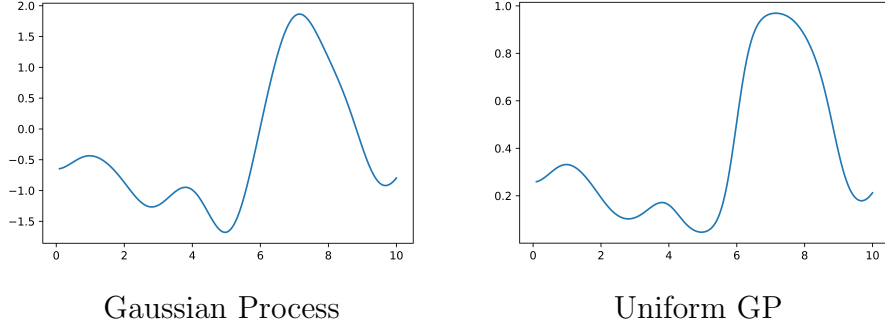


Figure 1: A sample of a Gaussian Process and corresponding Uniform-Gaussian Process.

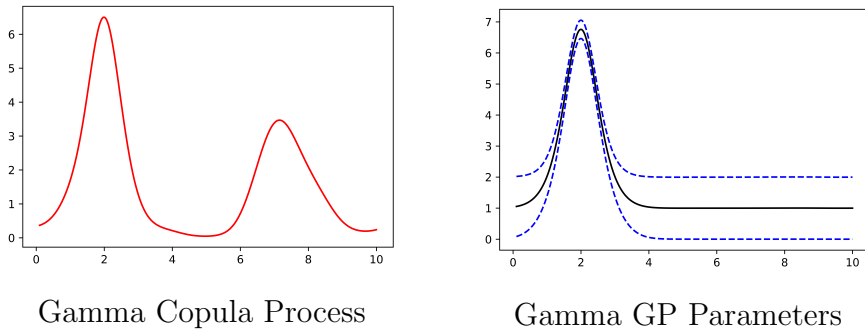


Figure 2: Left: The Gamma Copula Process implied by the Uniform-GP sample from Figure 1, parameterised by the mean and standard deviation shown on the right. Data: a Poisson process, sampled at  $x=2$  for  $t=80$ , with  $\lambda=7$ .

revenue depends on the price it sets via its effect on the rate at which sales occur. I discuss the choice to make  $\mathcal{P}$  a non-discrete set further on. The monopolist is able to choose the posted price that it offers, and incurs no cost to do so, nor to change from one price to another. When setting a price, the firm sets off a homogeneous Poisson process  $N(0, t]$ , which will be parameterised by rate parameter  $\lambda : \mathcal{P} \implies \mathbb{R}^+$ , where  $\lambda(p) \in \mathcal{C}^\infty$ , the set of smoothly differentiable functions, with  $\lambda' < 0$ . For simplicity we will refer to this function as the demand function, even though it is not technically one in the formal sense. We can, for simplicity, assume that the implied revenue function will be strictly concave, and have a unique maximum.

## 9 Full Information Benchmark

The full information benchmark, due to the framework which we have set up, implies a specific closed form for revenue. To construct this, note first that since each inter-arrival time of each sale is exponentially distributed, the

expected discounted value of the first sale is given by:

$$p \cdot \mathbb{E} [e^{-\rho t}] \quad (13)$$

where the expectation over  $t$  is taken with respect to the exponential distribution parameterised by  $\lambda(p)$ , and  $\rho$  is the discount rate. Here, I use  $\lambda$  and  $\lambda(p)$  interchangeably so as to improve readability. This is by definition the Laplace transform of an exponentially distributed random variable, so this expression is equal to

$$p \cdot \frac{\lambda}{\lambda + \rho} \quad (14)$$

The value of the second sale is then similar to that in (13), but now the random variable  $t$  is distributed according to the sum of two independent exponential random variables. This means that the Laplace transform is:

$$\mathbb{E} [e^{-\rho(t+t')}] = \mathbb{E} [e^{-\rho t} e^{-\rho t'}] \quad (15)$$

$$= \mathbb{E} [e^{-\rho t}] \mathbb{E} [e^{-\rho t'}] \quad (16)$$

$$= \left( \frac{\lambda}{\lambda + \rho} \right)^2 \quad (17)$$

This means that the discounted present value of the  $n$ -th exponentially distributed arrival in sequence is  $\left( \frac{\lambda}{\lambda + \rho} \right)^n$ . To evaluate the expected value of a Poisson process for a given amount of time  $t$ , we can then sum over the value of sequences multiplied by their respective probabilities:

$$U(t) = \sum_{n=1}^{\infty} Pr(N_t = n) \sum_{i=1}^n p \left( \frac{\lambda}{\lambda + \rho} \right)^i \quad (18)$$

where  $N_t \sim \text{Poisson}(\lambda t)$ . This is then:

$$= \sum_{n=1}^{\infty} p \left( \frac{1 - u - 1 + u^{n+1}}{1 - u} \right) Pr(N_t = n) \quad (19)$$

$$= \sum_{n=1}^{\infty} pu \frac{1 - u^n}{1 - u} Pr(N_t = n) \quad (20)$$

where  $u \equiv \frac{\lambda}{\lambda + \rho}$ . Taking common factors and summing, we get

$$U(t) = \frac{u}{1-u} \sum_{n=1}^{\infty} pPr(N_t = n)(1-u^n) \quad (21)$$

$$= \frac{u}{1-u} p \left( \sum_{n=1}^{\infty} \frac{(\lambda t)^n e^{-\lambda t}}{n!} - \sum_{n=1}^{\infty} \frac{(\lambda t u)^n e^{-\lambda t}}{n!} \right) \quad (22)$$

$$= p \cdot \frac{e^{-\lambda t} u}{1-u} \left( \sum_{n=1}^{\infty} \frac{(\lambda t)^n}{n!} - \sum_{n=1}^{\infty} \frac{(\lambda t u)^n}{n!} \right) \quad (23)$$

$$= p \cdot \frac{e^{-\lambda t} u}{1-u} ([e^{\lambda t} - 1] - [e^{\lambda t u} - 1]) \quad (24)$$

$$= p \cdot \frac{u}{1-u} (1 - e^{\lambda t u - \lambda t}) \quad (25)$$

$$= p \cdot \frac{\lambda}{\rho} \left( 1 - \exp \left\{ \frac{-\lambda \rho t}{\lambda + \rho} \right\} \right) \quad (26)$$

This means that we can construct the objective function by chaining together a sequence of Poisson-revenue processes, each discounted up to the time they begin:

$$V(\{p_i\}, \{t_i\}) = \sum_{i=0}^{\infty} e^{-\rho t_i} U(t_i) \quad (27)$$

$$= \sum_{i=0}^{\infty} \frac{p_{i+1} \lambda_{i+1}}{\rho} e^{-\rho t_i} \left( 1 - e^{-\frac{\rho \lambda_{i+1} (t_{i+1} - t_i)}{\lambda_{i+1} + \rho}} \right) \quad (28)$$

where  $\lambda_i \equiv \lambda(p_i)$ , and  $t_0 \equiv 0$ . (The notation on  $p$  is so that price  $p_j$  is charged between times  $t_{j-1}$  and  $t_j$ .) The firm's aim is to maximise this function. This can be decomposed into a flow value and a continuation value in order to construct a Bellman equation. Then, the flow value is given by (26), and the discount factor is  $e^{-\rho t_i}$ . This would normally warrant further explanation regarding the existence of a fixed point of an appropriately defined Bellman operator. However, it is clear from the assumptions made on  $\lambda$  that if  $\lambda(p)$  is available to the firm, then it will charge the 'optimal' price at the beginning and stay at that price forever. Under full-information, and due to the assumptions on  $\lambda$ , standard FOCs on the expression for expected revenue will be necessary and sufficient in order to maximise expected lifetime discounted revenue. Evaluating at maximum and taking  $\lim_{t \rightarrow \infty} U(t)$ , we get:

$$V = \frac{p^* \lambda^*}{\rho} \quad (29)$$

where  $\lambda^* = \lambda(p^*)$ .

## 10 Unknown Demand Curve

To model the situation in which the firm does not know its demand curve, we assume that the firm does not initially know any point on  $\lambda(p)$ , but has some belief as to where it is. Since the firm effectively does not know its maximand, simple first-order conditions are not feasible. The firm must again choose a sequence of prices  $\{p_i\}$  and a corresponding sequence of *waiting times*  $\{t_i\}$  so as to maximise the lifetime discounted expected revenue. Time  $t_i$  corresponds to the length of time that price  $p_i$  has been maintained.

To understand the firm's problem, consider the following situation. The firm has set some price and is observing sales arriving according to the Poisson process at that point. The firm can wait long enough to learn the height of the demand at that point arbitrarily well. It may believe that this price earns it a relatively high revenue and be satisfied; we can in such a situation say that the firm is exploiting its knowledge to earn revenues. However, unless this is at the optimal price, revenues will be suboptimal, so there is good reason for the firm to conduct exploration so as to acquire knowledge of other parts of the demand function. It can potentially find more 'profitable' prices to offer. It is this tradeoff that represents the firm's problem. The problem is to find the sequence of prices and waiting times that maximise lifetime revenue given this framework.

As set up, the problem is essentially a multi-armed bandit problem. The 'arms' are the prices, and the returns, for a given period of waiting at any one arm, are a function of a Poisson process. However,  $\mathcal{P}$  being non-discrete means that this setup is not the same as that usually used for multi-armed bandit problems. My choice to pose the problem as one of facing a continuum-armed bandit (as in Agrawal (1995)) is justified as follows. If the set of controls is discrete, and returns from arms are uncorrelated, the problem is the basic multi-armed bandit problem as analysed by Gittins (1979). An optimal solution has been found in the form of the Gittins index. However, learning in such a setting requires that polling an arm informs on that arm and that arm alone. The set of arms (or prices) is then pollable in finite time, but we must then relinquish any kind of correlation between two arms. As a result, further increasing the cardinality of the discrete set of arms does not get around the fact that, regardless of how 'close' prices are to each other, information learnt for given prices is uninformative on other prices; this renders exploitation ex-

tremely expensive, and learning occurs slowly.

To get around this one can consider formally the extent to which the agent believes that information on demand at price  $p$  is relevant for understanding demand at price  $p'$ . It is reasonable to assume some level of smoothness in demand functions; small changes in price should reasonably lead to small changes in demanded quantity, as expressed here by the intensity of the stochastic process. It is thus reasonable to use some mechanism for describing the relationship between arms. Such a mechanism would, in effect, mimic continuity in the parameters as a function of the control. This leads me to consider posing the problem as having a continuous interval of 'arms'.

The problem thus posed is fundamentally more complex than the standard multi-armed bandit problem. Correlation across arms means that Gittins' forward induction program is not optimal, and the state variable for such a decision process does not necessarily have the Markov property. Expressing it as a dynamic programming problem implies that the solution consists of a policy function and a state transition function. The policy function solving such a problem would have to be defined on ever larger sets of possible combinations of observations; this very much complicates the problem. Relatively recent works such as Pandey et al. (2007) have looked at MABs with dependence across arms; these however admit that the optimal solution to the problem is infeasible. There is generally little understanding at the present moment as to even the computational complexity of the bandit problem in general, let alone a solution framework.

Since agents cannot simply be modelled as solving the problem, I model the firm as approximating 'optimal' behaviour by using algorithms from machine learning that acquire relatively good results. Specifically, I will implement algorithms in the class used to conduct 'Bayesian Optimisation' - a set of techniques originally developed mainly by J. Mockus. (Mockus, 1989). I consider this as analogous to using the calculus of variations to model the optimisation problem of agents in dynamic models, or construction of Lagrangians in static microeconomic models. This approach is justified by some recent experimental psychology literature, comparing the performance of algorithms used in machine learning with that of humans solving similar tasks (Wu et al., 2017). The work appears to show that there are good similarities between the two, and hence that use of the one as a proxy for behaviour of the other is a viable approach to research.

To replicate such a situation, I present the following model. Assume that the firm has prior belief over  $\lambda(p) \forall p \in \mathcal{P}$ . Since this is a theoretical model,

one can freely assume a conjugate prior without loss of flexibility. Because the interarrival times of the Poisson process with rate parameter  $\lambda(p)$  are distributed according to an exponential distribution, each sample of sales at a given price is effectively a sample from that distribution. The conjugate prior for this model is the Gamma distribution, parameterised by  $(\alpha(p), \beta(p))$  and with pdf:

$$\frac{\beta(p')^{\alpha(p')}}{\Gamma(\alpha)} \lambda^{\alpha(p')-1} \exp\{-\lambda\beta(p')\} \quad (30)$$

That is, prior belief is expressed at any point  $p'$  with a pair  $(\alpha(p'), \beta(p'))$ . The use of conjugate priors means that, when updating from prior to posterior, only the parameters need to be updated; for the exponential distribution rate parameter  $\lambda(p')$ , with a sample  $\{t_i\}$  the conjugate posterior parameters would become

$$\left( \alpha(p') + n, \beta(p') + \sum_{i=1}^n t_i \right) \quad (31)$$

It is clear that, sampling for infinite time, belief at the point  $p'$  will converge since

$$\frac{\alpha + n}{\beta + \sum_{i=1}^n t_i} = \frac{\frac{\alpha}{n} + 1}{\frac{\beta}{n} + \frac{1}{n} \sum_{i=1}^n t_i} \xrightarrow{p} \lambda \quad (32)$$

by the law of large numbers and the assumption of independent arrivals. (The reader is reminded that the exponential distribution, when parameterised to have pdf  $\lambda e^{-\lambda t}$ , has mean  $\frac{1}{\lambda}$ ).

This approach is a departure from the basic Bayesian optimisation framework. That setup implies that the function can be observed directly, or with some i.i.d Gaussian noise. In that case, the function is assumed to follow a Gaussian process, a ridge is added to the kernel/covariance matrix, and the noise is handled very simply (Brochu et al., 2010). My framework allows me to generalise the approach to handle the case where the maximand is a function of a parameter which in this case is the intensity of the Poisson process; observations of the parameter cannot be made directly, and it may be that the noise is not Gaussian.

We could thus endow the firm with a pre-chosen pair of functions,  $(\alpha(p), \beta(p))$ , and use these to model updating belief over the returns of the arms. The problem implied by this fact is that the agent now learns only on sets of zero

measure; the prior functions  $(\alpha(p), \beta(p))$  will, as they are updated at observed points  $\{p_i\}$  and become posteriors, acquire removable discontinuities if updated at those points naively. Consequently, for any price  $p'$ , belief is not updated for any price  $p + \varepsilon$  where  $\varepsilon$  is any real number.

I thus augment the framework to handle this situation, modelling the prior functions as warped Gaussian processes. The conjugate nature of the prior belief means we can update these hyperparameter functions as though they are being observed directly and without noise. These themselves parameterise a Gaussian-Gamma copula process, so that points of the unknown function  $\lambda(p)$  are marginally Gamma-distributed, but their correlation structure is described by a simple Gaussian covariance matrix. For this matrix we will use the squared exponential kernel which allows for relatively high flexibility. The 'automatic relevance determination' parameter (Brochu et al., 2010), denoted  $l$  above, can then be shifted so as to regulate the extent to which updating belief on  $p$  informs on price  $p'$ . Hence, prior belief can be described using hyperparameters on the Gaussian processes for the parameters. Since for the Gamma distribution, both parameters must be positive, I warp their respective Gaussian processes  $\{\mathcal{GP}_\alpha, \mathcal{GP}_\beta\}$  into  $\mathbb{R}^+$  (see Snelson et al., 2004). Sampling at price  $p$  updates the parameters at that point as in (31), and I generate Gaussian processes for  $\alpha(p)$  and  $\beta(p)$  conditioned on these observations. This then circumvents the difficulties of observing only a statistic converging to  $\lambda(p)$ , and keeps the entire model within a Bayesian framework. We then avoid the necessity of choosing an estimator with which to evaluate a given sample. It also effortlessly handles the noise present due to sampling, without imposing Gaussian noise for all sample sizes. Having built this framework, I can now begin to detail the price selection algorithm.

## 11 Two-Stage Thompson Sampling

Suppose an agent's returns depend on some stochastic process  $X_t$ , whose distribution's parameters depend on some control variable  $u$ . The agent wants to optimise w.r.t this variable, choosing the parameter that provides some measure of 'best' returns as a result of the stochastic process. Suppose some parameter

$$\theta : \mathcal{U} \rightarrow \Theta \text{ where } \theta(u) \in \mathcal{C}^\infty \tag{33}$$

and that  $\theta$  parameterises some distribution in the exponential family. The aim is to maximise  $h(u) \equiv g(\theta(u))$ , where  $g \in \mathcal{C}^\infty$ . One cannot sample  $h(u)$  in its entirety; only at individual points. I refer to this property as "pointwise observability". Assume the form of  $g$  is known. When observing data about  $h(u)$ , one has prior belief over the form of  $\theta(u)$ . We will denote this prior distribution p.d.f  $\gamma$ , and its c.d.f.  $\Gamma$ . Since  $\theta$  parameterises some distribution with p.d.f  $f$  and c.d.f  $F$  it can never be learnt exactly for any  $u$ . However, we can use Bayesian conjugate priors to describe a mechanism for optimisation of  $h$ . We describe belief about  $\theta$  using the hyperparameters (the parameters of the prior distribution with cdf  $\Gamma$ ). Thus we have a function

$$\psi(u) : \mathcal{U} \rightarrow \Psi \tag{34}$$

where  $\Psi$  is the parameter space for the conjugate prior. For example, if  $\theta$  is the mean of a normal distribution, then the conjugate prior is also normal, and  $\Psi = \mathbb{R} \times \mathbb{R}_+$ .

Thompson sampling (Thompson, 1933) is a method for approaching the maxima of unknown but pointwise observable functions, while handling the exploration-exploitation tradeoff. The method is as follows. An agent has prior belief over the maximand, and for continuous functions it is customary to describe this belief using a Gaussian process with a smoothing kernel. The agent then generates a sample function  $\nu(u)$  from the prior belief defined on the set of controls, and maximises this. Data is updated with the maximand evaluated at  $\operatorname{argmax} \{\nu\}$ . The process begins again, with the data now incorporating the new observation. The sequence of maximisers chosen converges to the maximiser of the true function, and converges at an exponential rate (Basu and Ghosh, 2017). The simplicity of the formulation and the global guarantees that exist for this algorithm have meant that much recent work has been done on its properties and it finds use in many industrial applications.

In our case, however, we have assumed we can take samples at points in  $\mathcal{U}$ , and we never observe the true maximand  $h$  directly. Thompson sampling in particular and Bayesian optimisation in general can handle noisy observations of maximands, but such methods focus on Gaussian noise. To handle functions with domains other than  $\mathbb{R}$ , we can use warped Gaussian processes (see Snelson et al., 2004), but then it is not always clear what the distribution of the noisy observation will be. In our case the firm takes samples rather than observing the function directly. Furthermore, the variance of the sample will depend on the underlying distribution of the process  $X_t$  so one cannot always handle this

noise as an observable or something whose distribution can be assumed. Only if the sample is very large can one approximate this noise by taking advantage of the CLT.

The proposed solution is as follows. I refer to this as two-stage Thompson sampling due to the use of two separate sets of Gaussian processes. First, one generates a Gaussian process  $\mathcal{GP}_d$ , where  $d = \dim \{\Psi\}$ . With an appropriate warping function  $W : R^d \rightarrow \Psi$ , strictly increasing in all arguments. Using this, construct surrogate hyperparameters  $\hat{\psi} = W(\mathcal{GP}_d)$ , so that  $\hat{\psi}(u) \in \Psi$ . We then use this to parameterise the cdf  $\Gamma$ .

Now we generate a second Gaussian process  $\mathcal{GP}_h$ , with mean zero and the chosen kernel. Construct

$$\hat{\theta}(u) = \Gamma_{\hat{\psi}}^{-1}(\Phi(\mathcal{GP}_h(u))) \quad (35)$$

$\hat{\theta}(u)$  is now a Gaussian copula process where the auxiliary distribution is the marginal distribution  $\Gamma$ . That is, we generate a sample function such that  $\hat{\theta}(u) \sim \gamma(\bullet|\hat{\psi}(u))$ . Evaluate  $\hat{h}(u) = g(\hat{\theta}(u))$ , and calculate

$$u_{t+1} = \underset{u}{\operatorname{argmax}} \hat{h}(u) \quad (36)$$

With a given sampling budget of  $n$  observations, we take a sample from  $F$  parameterised by true  $\theta(u)$  at  $u_{t+1}$  and update the data  $\mathcal{D}$ . Repeat the cycle using  $\hat{\psi}(u|\mathcal{D})$ .

Two-Stage Thompson Sampling

**Data:**

Warping function  $W : \mathbb{R} \rightarrow \Psi$ ;

Hyperprior parameters:  $m(x), \sigma, k(x, x')$ ;

**for**  $t = 1, 2, \dots$  **do**

Generate $\hat{\psi} = W(\mathcal{GP}(m(x), \sigma k(x, x') \mathcal{D}))$ ;
Generate $\hat{\theta}(u) = \mathcal{GCP}(\Gamma^{-1}(\bullet \hat{\psi}), 0, k(u, u'))$ , a Gaussian copula process with marginal measure the hyperprior c.d.f.;
$u_{t+1} = \operatorname{argmax} [g(\hat{\theta}(u))]$ ;
Take a sample at this point and update the data;

**end**

The mechanism described thus elegantly handles sampling error in observation of the maximand, allows for much more general distributions for  $X_t$ ,

and nests standard noisy Bayesian optimisation. To see why, assume that:

$$W(x) = x \tag{37}$$

$$\Psi = \mathbb{R} \times \mathbb{R}_+ \tag{38}$$

$$\psi(u) = (0, k(u, u)) \forall u \tag{39}$$

$$h(u) \sim \mathbb{N}(m(u), \sigma(u)) \tag{40}$$

$$\Gamma = \Phi \tag{41}$$

In this case, when generating the hyperparameter functions, we have a standard Gaussian process with mean zero and variance described by  $k(u, u')$ , the chosen kernel. For smoothness I assume the squared exponential kernel as described above. Using this, we generate a Thompson sample function from a  $\mathcal{GP}$  prior using the generated hyperparameters, and take its maximum. In the simplest case, this is just a composition of two normally distributed random variables, and is equivalent to standard Thompson sampling with Gaussian noise.

## 12 Application

I can now describe application of this to the model. The firm is endowed with a distribution over hyperparameter functions  $(\alpha(p), \beta(p))$ . These functions play the role of  $\psi$  above, and represent, on a stage by stage basis, firm belief over the Poisson demand intensity function. Here,  $\hat{\psi} = (\hat{\alpha}(p), \hat{\beta}(p))$ . These then parameterise a Gamma distribution cdf (i.e.  $\Gamma$  in the exegesis above), and one can generate the sample functions from the posterior over the maximand. These will be realisations of a Gamma-Gaussian copula process, and will be samples of  $\lambda(p)$ . These then allow one to build a sample revenue functions, multiplying the generated sample functions  $\lambda(p)$  by  $p$  (i.e. constructing  $h$  above). and hence to conduct Thompson sampling.

In terms of what this means for the firm’s behaviour, this represents the process of the firm “considering” its demand function, and picking the maximiser of the corresponding revenue function. Since the firm in any case is considered to optimise the revenue function it *believes* it faces, this is reasonable as a paradigm for modelling firm behaviour. The system here thus formalises the updating process in an internally consistent way.

The Bayesian model that is implied by this setup is:

$$f(X_t | \lambda(p)) \cdot g(\lambda(p) | \alpha(p), \beta(p)) \cdot h(\alpha(p), \beta(p)) \quad (42)$$

$$\forall p \in \mathcal{P}, \text{ and where } \alpha, \beta \sim W(\mathcal{GP}(0, K)) \quad (43)$$

where  $K$  is a kernel.

In this application, the warp function  $W(\cdot)$  I use is the exponential function. This is useful since this implies that marginally,  $\alpha(p) = e^{x(p)}$ , where  $x(p) \sim \text{Normal}(\mu, \sigma)$  and  $\beta(p) = e^{y(p)}$ , where  $y(p) \sim \text{Normal}(\mu', \sigma')$ . hence,  $\frac{\alpha(p)}{\beta(p)} = e^{x(p)-y(p)}$ . That is,  $\frac{\alpha}{\beta}$  is log-normally distributed.

It is worth discussing why one does not just use a prior such as  $\ln \lambda \sim \mathcal{GP}(0, K)$ . Generation of sample Gaussian processes implies inversion of matrices of dimension equal to the number of tested points, and Cholesky decomposition of matrices of size  $\overline{\mathcal{P}}$  (ie, the control/input space), when  $\mathcal{P}$  has been discretised. This two-stage Thompson algorithm requires three such GPs to be generated for every sample function. However, when the number of tested points is low, this is not especially complex. This means that sample generation is simplified since at a small scale, generation of GPs is relatively fast. I can then use the transformations described above. In the case where one uses  $\ln \lambda$  following a GP as the prior, there is a non-conjugacy and thus a relatively complex posterior. This means that generation of samples no longer takes advantage of the inverse-CDF method of simulating random variables that the copula process is able to use; one must resort to Monte Carlo Markov Chain (MCMC) algorithms for the generation of posterior sample functions, since the posterior is no longer a member of the same family. However, approximations to the posterior, using for example Bayesian variational techniques, could be used to generate approximate sample functions relatively easy, and this is likely the technique that would be used if one were to do online dynamic pricing in real-time. Restricting ourselves to a theoretical model, however, means that this is not a problem.

Furthermore, since as noted above the mean of the Gamma distribution can be written as a log-normal random variable due to our prior formulation, it is possible to take advantage of the simple data-updating rules for Gaussian processes. Then, using Copula processes one can get exact marginal distributions for the parameter  $\lambda$ . I conjecture that this will always be possible within the exponential family with an appropriately designed linking conjugacy. If this is true, any function of a parameter of a distribution in the exponential family can be simply optimised using a Gaussian copula process, Thompson sampling,

and an appropriately designed prior. This appears to be a completely novel approach.

Much of what is written in the sequel suppresses notation displaying the fact that the sequence of generated functions  $\{\alpha_i(p_i), \beta_i(p_i)\}$  are conditioned on to get the flow value in each stage. Conditioning on these and referring them only to  $\{\alpha, \beta\}$  simplifies notation and improves readability.

### 13 The Value Function

The firm now operates as follows. It chooses a price  $p_1$ , waits time until time  $t_1$ , and updates the data. Then, it sets price  $p_2$ , waits until time  $t_2$ , etc. This implies a stochastic optimisation problem, where the control variable is the time to maintain each price 'given' to the firm by the operation of the Thompson sampling algorithm. Construct the value function:

$$V(\mathcal{D}) = \max_t \{ \Psi(t; p) + e^{-\rho t} \mathbb{E}[V(\mathcal{D}')] \} \quad (44)$$

where  $\Psi(t; p)$  is a flow value to be determined, and  $\mathcal{D}$  the state variable. The expectation is taken with respect to the measure describing the probability  $\mathbb{P}\{\mathcal{D}_{N+1} = \mathcal{D}' | \mathcal{D}_N = \mathcal{D}\}$ . This variable consists of a sequence of triplets:

$$\mathcal{D}_N = \left\{ \left( p_j, \alpha(p_j) + n_{p_j}, \beta(p_j) + \sum_{i=1}^{n_{p_j}} t_i \right) \right\}_{j=1}^N \quad (45)$$

where  $n_p$  the number of arrivals at price  $p$ ,  $\sum_{i=1}^{n_{p_j}} t_i$  the time spent at price  $p$ , and  $N$  the number of different prices tested. This contains all the information required for the operation of the Thompson sampling process, beside the prior functions  $(\alpha, \beta)$ . The decision problem then is to solve the above for  $V$ . Define the operator:

$$\mathcal{T}(z)(\mathcal{D}) = \max_t \{ \Psi(t; p) + e^{-\rho t} \mathbb{E}[z] \} \quad (46)$$

Solution of the problem constitutes finding fixed points of the operator  $\mathcal{T}$ , should they exist. They will do if  $t_i > 0$ , as then this operator fulfill's Blackwell's conditions (Blackwell, 1965).

## 14 The Flow Value

In order to construct this value function, we need to take into account the fact that the firm has uncertainty over the parameter for a given  $\lambda$ . The price-choosing heuristic induces the Poisson process that the firm observes, and the firm chooses the time to wait. Therefore, the value of a period of waiting has to be evaluated. This value depends on the unknown parameter  $\lambda$ , over which the firm has belief described as above by a pair  $(\alpha, \beta)$ . That is, the interarrival times are exponentially distributed according to  $\lambda$ , but for the firm,  $\lambda$  itself is a random variable. Hence, to acquire the flow value relevant for the dynamic programming problem, the value of a Poisson process for a given time  $t$  where its parameter is Gamma-distributed must be acquired. In the sequel I will call this the Gamma-Poisson process for reference. I begin in similar fashion to the full information benchmark case examined above.

First we can compute the value of one arrival after a period of time distributed exponentially, where the rate parameter itself is  $\lambda \sim \text{Gamma}(\alpha, \beta)$ . It is well known that this distribution, which is the predictive posterior distribution of an exponential random variable with conjugate Gamma prior, is the Lomax distribution. This has pdf:

$$f(t) = \frac{\alpha\beta^\alpha}{(\beta + t)^{\alpha+1}} \quad (47)$$

Hence, the discounted value of an arrival paying  $p$  after such a period of time is given by the Laplace transformation of this distribution, which I derive here. The analysis follows that of Nadarajah and Kotz (2006).

$$p \cdot \mathbb{E} [e^{-\rho t}] \text{ where } t \sim \text{Lomax}(\alpha, \beta) \quad (48)$$

$$= p \cdot \int_0^{+\infty} e^{-\rho t} \frac{\alpha\beta^\alpha}{(\beta + t)^{\alpha+1}} dt \quad (49)$$

$$= p \cdot \alpha\beta^\alpha \int_0^{+\infty} \frac{1}{(\beta + t)^{\alpha+1}} e^{-\rho t} dt \quad (50)$$

$$(51)$$

Substituting  $u = (\beta + t)$ , the integral becomes:

$$= \alpha \beta^\alpha \int_{\beta}^{+\infty} \frac{1}{u^{\alpha+1}} e^{-\rho(u-\beta)} du \quad (52)$$

$$= \alpha \beta^\alpha e^{\rho\beta} \int_{\beta}^{+\infty} \frac{1}{u^{\alpha+1}} e^{-\rho u} du \quad (53)$$

$$(54)$$

now, substituting again  $y = \rho u$ , we can find:

$$= \alpha \beta^\alpha e^{\rho\beta} \rho^{\alpha+1} \int_{\beta}^{+\infty} \frac{1}{(\rho u)^{\alpha+1}} e^{-\rho u} du \quad (55)$$

$$= \alpha \beta^\alpha e^{\rho\beta} \rho^{\alpha+1} \int_{\rho\beta}^{+\infty} y^{-\alpha-1} e^{-y} \frac{1}{\rho} dy \quad (56)$$

$$= \alpha (\beta\rho)^\alpha e^{\rho\beta} \int_{\rho\beta}^{+\infty} y^{-\alpha-1} e^{-y} dy \quad (57)$$

$$= \alpha (\beta\rho)^\alpha e^{\rho\beta} \Gamma(-\alpha, \rho\beta) \quad (58)$$

where  $\Gamma(x, y)$  is the upper incomplete Gamma function. The value is then:

$$= p \cdot \alpha e^{\rho\beta} (\beta\rho)^\alpha \Gamma(-\alpha, \rho\beta) \equiv p \cdot \gamma \quad (59)$$

For what follows it is necessary that this expression  $\gamma \leq 1$ . This is true if:

$$\alpha^{-1} e^{-\rho\beta} (\rho\beta)^{-\alpha} \geq \Gamma(-\alpha, \rho\beta) \quad (60)$$

$$\Leftrightarrow \alpha^{-1} e^{-\rho\beta} (\rho\beta)^{-\alpha} \geq \int_0^{+\infty} (x + \rho\beta)^{-(\alpha+1)} e^{-(x+\rho\beta)} dx \quad (61)$$

$$\Leftrightarrow \alpha^{-1} e^{-\rho\beta} (\rho\beta)^{-\alpha} \geq e^{-\rho\beta} \int_0^{+\infty} (x + \rho\beta)^{-(\alpha+1)} e^{-x} dx \quad (62)$$

$$\Leftrightarrow \alpha^{-1} (\rho\beta)^{-\alpha} \geq \int_0^{+\infty} (x + \rho\beta)^{-(\alpha+1)} e^{-x} dx \quad (63)$$

Integrating by parts, we have:

$$\alpha^{-1} (\rho\beta)^{-\alpha} \geq \alpha^{-1} e^{-x} (x + \rho\beta)^{-\alpha} \Big|_0^{+\infty} - \int_0^{+\infty} \frac{(x + \rho\beta)^{-\alpha}}{\alpha} e^{-x} dx \quad (64)$$

$$\Rightarrow \alpha^{-1} (\rho\beta)^{-\alpha} \geq \alpha^{-1} (\rho\beta)^{-\alpha} - \int_0^{+\infty} \frac{(x + \rho\beta)^{-\alpha}}{\alpha} e^{-x} dx \quad (65)$$

$$\Rightarrow (\rho\beta)^{-\alpha} \geq (\rho\beta)^{-\alpha} - \int_0^{+\infty} (x + \rho\beta)^{-\alpha} e^{-x} dx \quad (66)$$

$$\Rightarrow (\rho\beta)^{-\alpha} \geq (\rho\beta)^{-\alpha} - \Gamma(-\alpha + 1, \rho\beta) \quad (67)$$

$$(68)$$

which is clearly true since the latter term is positive. To get the value of later sales, say the  $n$ -th one, we can again take the  $n$ -th power of  $\gamma$ . This is because we are evaluating the moment generating function for the  $n$ -th arrival at  $-\rho$ . That is, the value of the  $n$ -th sale accruing according to the Gamma-Poisson process will be

$$p \cdot (\alpha (\beta\rho)^\alpha e^{\rho\beta} \Gamma(-\alpha, \rho\beta))^n \quad (69)$$

We now have to acquire the probability of  $n$  sales in the interval  $(0, t]$ . For a given time  $t$ , this is the probability  $\mathbb{P}(N(0, t] = k)$ , where  $N(0, t] \sim Po(\lambda t)$  and  $\lambda \sim Gamma(\alpha, \beta)$ . This is equivalent to the distribution of the counting function associated with a renewal process with interarrival times according to the Lomax distribution described above.

$$\mathbb{P}(N(0, t] = k) = \int_0^\infty \mathbb{P}(N(0, t] = k \mid \lambda) f(\lambda) d\lambda \quad (70)$$

$$= \int_0^\infty \frac{(\lambda t)^k}{k!} e^{-\lambda t} \cdot \lambda^{\alpha-1} e^{-\lambda\beta} \frac{\beta^\alpha}{\Gamma(\alpha)} d\lambda \quad (71)$$

$$= \frac{\beta^\alpha}{\Gamma(\alpha)} \frac{t^k}{k!} \int_0^\infty \lambda^{k+\alpha-1} e^{-\lambda(\beta+t)} d\lambda \quad (72)$$

$$= \frac{\beta^\alpha}{\Gamma(\alpha)} \frac{t^k}{k!} \left( \frac{t + \beta}{t + \beta} \right)^{k+\alpha-1} \int_0^\infty \lambda^{k+\alpha-1} e^{-\lambda(\beta+t)} d\lambda \quad (73)$$

$$= \frac{\beta^\alpha}{\Gamma(\alpha)} \frac{t^k}{k!} \left( \frac{1}{(t + \beta)^{k+\alpha-1}} \right) \int_0^\infty (\lambda(t + \beta))^{k+\alpha-1} e^{-\lambda(\beta+t)} d\lambda \quad (74)$$

$$(75)$$

and substituting  $u = \lambda(t + \beta)$ , we get

$$= \frac{1}{\Gamma(\alpha)} \left( \frac{\beta^{\alpha} t^k}{(t + \beta)^{k+\alpha-1}} \right) \frac{1}{k!(t + \beta)} \int_0^{\infty} u^{k+\alpha-1} e^{-u} du \quad (76)$$

$$\mathbb{P}(N(0, t] = k) = \left( \frac{\beta^{\alpha} t^k}{(t + \beta)^{k+\alpha}} \right) \frac{\Gamma(k + \alpha)}{k! \Gamma(\alpha)} \quad (77)$$

$$(78)$$

Using this probability, the expected value of access to a Gamma-Poisson process for a length of time  $t$  as described above is:

$$\Psi(t; p) \equiv p \cdot \sum_{n=1}^{\infty} \underbrace{\left( \frac{\beta^{\alpha} t^n}{(t + \beta)^{n+\alpha}} \right) \frac{\Gamma(n + \alpha)}{n! \Gamma(\alpha)}}_{\mathbb{P}(N(0, t] = n)} \underbrace{\sum_{i=1}^n (\alpha(\rho\beta)^{\alpha} e^{\rho\beta} \Gamma(-\alpha, \rho\beta))^i}_{\text{Value of n sales}} \quad (79)$$

$$= p \cdot \sum_{n=1}^{\infty} \left( \frac{\beta^{\alpha} t^n}{(t + \beta)^{n+\alpha}} \right) \frac{\Gamma(\alpha + n)}{n! \Gamma(\alpha)} \sum_{i=1}^n \gamma^i \quad (80)$$

where

$$\gamma \equiv \alpha(\rho\beta)^{\alpha} e^{\rho\beta} \Gamma(-\alpha, \rho\beta), \gamma < 1 \quad (81)$$

for a given  $t$ . This then functions as the flow value for the firm at price  $p$ . Similarly to the full information case, we can simplify the value of  $n$  sales:

$$\sum_{i=1}^n \gamma^i \quad (82)$$

$$= \frac{1 - \gamma^{n+1}}{1 - \gamma} - 1 \quad (83)$$

$$= \frac{1 - \gamma^{n+1}}{1 - \gamma} - \frac{1 - \gamma}{1 - \gamma} \quad (84)$$

$$= \frac{\gamma - \gamma^{n+1}}{1 - \gamma} \quad (85)$$

$$= \gamma \frac{1 - \gamma^n}{1 - \gamma} \quad (86)$$

$$(87)$$

Substituting back in, we have:

$$= p \cdot \frac{\gamma}{1 - \gamma} \frac{\beta^{\alpha}}{\Gamma(\alpha)(t + \beta)^{\alpha}} \sum_{n=1}^{\infty} \frac{\Gamma(\alpha + n)}{n!} \left( \left( \frac{t}{t + \beta} \right)^n - \left( \frac{t\gamma}{t + \beta} \right)^n \right) \quad (88)$$

Note that

$$\begin{aligned}\Gamma(\alpha + n) &= \Gamma(\alpha)(\alpha)(\alpha + 1)(\alpha + 2)(\alpha + 3)(\alpha + 4)\dots(\alpha + n - 1) \\ &\equiv \Gamma(\alpha)\alpha^{(n)}\end{aligned}$$

where  $\alpha^{(n)}$  is the Pochhammer factorial of  $\alpha$ . It is clear that the latter term in (88) is the difference between two exponential generating functions of the Pochhammer factorial. Hence, we can use Petojević (2008) to show that this will be:

$$\begin{aligned}p \cdot \frac{\gamma}{1 - \gamma} \frac{\beta^\alpha}{\Gamma(\alpha)(t + \beta)^\alpha} \Gamma(\alpha) &\left[ \frac{1}{\left(1 - \frac{t}{t + \beta}\right)^\alpha} - \frac{1}{\left(1 - \frac{\gamma t}{t + \beta}\right)^\alpha} \right] \\ &= p \cdot \frac{\gamma}{1 - \gamma} \beta^\alpha \left[ \frac{1}{\beta^\alpha} - \frac{1}{((1 - \gamma)t + \beta)^\alpha} \right] \\ \Rightarrow \Psi &= p \cdot \frac{\gamma}{1 - \gamma} \left[ 1 - \frac{\beta^\alpha}{((1 - \gamma)t + \beta)^\alpha} \right]\end{aligned}\tag{89}$$

We now have the flow value for the firm.

## 15 Solution

Various methods for approaching solution of this model exist. Q-learning is a key tool in both model solution like this and reinforcement learning problems. This is similar in principle to the Parameterised Expectations Algorithm for policy iteration (den Haan and Marcet, 1990). Methods like this construct a parameterised function of the state variable, and usually use least squares regression to choose parameters for that function which best fit the model. A more modern approach, Deep-Q learning, uses neural networks to approximate this function. A feed-forward neural network for regression (as opposed to classification) is essentially a series of connected regression models; the output of one regression model becomes an input for the following set. The backpropagation algorithm is a method for estimating all regression coefficients (i.e, weights) in the set simultaneously. Varied combinations of weights and biases on the inputs at each node can be used to approximate any function (Hornik, 1991). This universality property has lead to widespread use of such networks in modern machine learning applications. As a result, since the state space is

extremely large, I can approximate the value function (or some component of it) using such a neural network. In a similar vein to the aforementioned PEA algorithm, I use the first order conditions of the problem to acquire an implicit policy function.

I can now solve the model using something akin to Howard’s Improvement algorithm (Ljungqvist and Sargent, 2004).

Substituting in the derivative w.r.t  $t$  of the flow value found above, the first order condition from the value function is written:

$$\begin{aligned} \Psi'(t) &= e^{-\rho t} [\rho V(\mathcal{D}') - V'(\mathcal{D}')] \\ \Leftrightarrow p \cdot \frac{\alpha \beta^{\alpha \gamma}}{((1 - \gamma)t + \beta)^{\alpha + 1}} &= e^{-\rho t} g(t, \mathcal{D}') \end{aligned} \quad (90)$$

where  $g(t, \mathcal{D}')$  will be an approximation to  $[\rho V(\mathcal{D}') - V'(\mathcal{D}')]'$ . Here,  $\hat{\mathcal{D}}$  will be substituted in for  $\mathcal{D}$ , where  $\hat{\mathcal{D}}$  will be some set of basis functions of features of the data, such as number of prices polled, average time, etc. Call the function that the network actually forms  $\hat{g}$ . First we generate random weights on the network. Using this, we generate a sequence of times  $t_i$  sequentially, by solving (90). Each element of the sequence implies data used to update the parameters for the next element. Using this sequence of times, we then train the network, by using the set of generated points in the state space as inputs and  $e^{\rho t} \Psi'(t; p)$  as the output. Having trained the network on this data, I re-solve equation (90), repeating the process. After enough repetitions, the function  $\hat{g}$  will be an approximation of  $g$ .

I built the approximating neural network using the TensorFlow Keras module. I used MSE as the loss function, and the ReLU activation function for the hidden layers. Three hidden layers of 32 nodes were used, and one output layer using a linear activation function. The inputs used were the mean and standard deviation of the prices sampled, the mean and standard deviation of the times spent at each price so far, the mean and standard deviation of the number of arrivals observed, and finally the number of prices sampled (i.e, how many different prices the firm has sampled so far). I considered these statistics appropriate as a summary of the information contained in each data point. The use of such statistics implies an equivalency class of data points with the same or similar features with respect to these metrics. It is possible that other propensity-scoring techniques can be used to generate equivalencies that are more appropriate for the form of the ‘data’ here, but work on this is outside the scope of this thesis. Furthermore, it is perhaps reasonable to

assume that the sufficient statistics of the distribution of the maximiser of the Thompson sampling algorithm are not dissimilar to those I use here.

The complexity of the state variable in this problem means that whilst a solution can be found using the method described above, there is no simple form to represent the results (such as easily interpretable parameters on the imputed function  $g$ ). However, I present simulated paths of the optimal time-to-wait, having run the training of the simulation such that a relatively low MSE on the weights of the network has been achieved.

## 16 Transition Distribution

I now turn to the expectation operator in (44). this is taken with respect to the transition distribution, given data at the current stage. At stage  $i$  (i.e, at the  $i$ -th sampled price), the state  $\mathcal{D}_i$  determines the transition distribution over  $\mathcal{D}_{i+1}$ . The Thompson sampling algorithm chooses the next price sampled and therefore the position of the state in the future, which is also affected by current chosen time  $t$ . It is well known that this distribution is impossible to evaluate analytically (see Bijl et al., 2016). Since its density does not exist, it is not generally possible to use standard Monte Carlo techniques such as importance sampling to acquire approximations.

One could approximate this distribution by simulating the relevant phenomenon directly. This would involve simulating large numbers of Thompson samples and keeping the maxima, for a large set of possible values of the state variable  $\mathcal{D}$ . These would allow for kernel density estimation and a good approximation of this latent distribution. However, such a technique would be extremely slow. Assuming an interpolation on the control space of 1000 points, generating 10,000 samples from the distribution for one point requires finding the Cholesky decomposition of 10,000 covariance matrices of size 1000. The Cholesky Decomposition is of complexity  $O(n^3)$ , so this can be slow even for a sample of one, and infeasible for any good level of approximation of the state space since this sample must be taken for every point in it. To tackle this problem, Bijl et al. (2016) have developed an algorithm they refer to as Monte Carlo Maximum Distribution approximation. This simplifies the problem by approximating the distribution of interest using a particle filter. Particle filters are a method for constructing samples from distributions by weighting particles according to their relevance, and resampling from the discrete distribution on the particles implied by the weights. Generating a sample of 10,000 can then be done by taking the Cholesky decomposition of 10,000 matrices of

much smaller size. One can then construct the empirical cdf as an approximation to the true cdf. For the large state space in this problem, this is still infeasible using regular computation. However, recent advances in computing have led to algorithms for batch-decomposition of very large numbers of small matrices. These involve parallelising computation by utilising the architecture of Graphics Processing Units.

GPUs are used for solving problems that require, for example, millions of simple arithmetic calculations to be conducted simultaneously. Standard Central Processing Units (CPUs) are specialised for fast sequential calculations. CPUs are able to achieve some level of parallelisation by having 4 or 8 cores for the purpose of simultaneous computation. Modern GPUs available to consumers can have up to 5120. Utilisation of such processors can allow for massively parallel batched calculation of matrix decompositions, allowing for very fast generation of approximate samples.

In order to acquire this transition distribution, I therefore adapt the MCMD algorithm for the Thompson Sampling mechanism to the Two-Stage Thompson Sampling approach, and implement it in the Numba CUDA GPU interface for Python. To achieve approximation of the transition distribution for given points in the state space, I wrote batch Cholesky decomposition routines for Numba CUDA, by hand, following Lemaitre and Lacassagne (2016). These allow me to generate approximate samples from transition distributions for any point in the chosen state space very quickly. Using CUDA I was able to achieve a great increase in speed; standard for-loop implementation of the technique on the CPU took around 20 minutes, but on the GPU this was cut down to around 40 milliseconds.

I had attempted to build a system to conduct value function iteration. This requires acquisition of the transition distribution for every point in the state space (ie, for every point in an approximating discretisation). However, despite cutting down the state space to areas that will be visited only with high probability, and utilising a symmetric objective function in order to halve the number of calculations, the state space still grows very quickly. I estimated that even using the high-speed parallelisation techniques I have developed, it would take around 2,000 minutes to calculate the distributions for when two prices have been observed; then approximately 1000 times this for three prices. It is clear that this is infeasible to run. As a result, I use the aforementioned PEA algorithm. This allows me to utilise a projection that cuts down the extent of the space over which I have to solve the model; the statistics based on the firm's observed information, that I use in my function approximation,

imply equivalence classes between different points in the true state space in a consistent manner, allowing for solution of the model in a reasonable time-frame.

## 17 Two-Stage Thompson Sampling: Monte Carlo Maximum Distribution

To acquire the next price, I adapted the aforementioned MCMD algorithm to suit the Two-Stage Thompson Sampling method outlined above. This TSTSM-CMD algorithm operates as follows.

One generates a set of uniformly distributed ‘particles’ on the interval over which the distribution is to be approximated. These particles will initially be uniformly distributed but the dynamics of the algorithm allow one to construct a sample, using these particles, of the distribution of interest. Having generated the initial particles, one then generates for each particle  $x_i$  a set of challenger particles  $\{x_{ic}\}_j^{N_c}$  as described by Bijl et al. (2016). These will then undergo a selection process and one of them will be chosen as a replacement for  $x_i$ . The process of generating the challenger particles is as follows: For particle  $i$ , draw each of  $N_c$  new particles from a low-variance normal distribution centred at  $x_i$ , with probability  $\delta$ , and uniformly on the interval of controls (i.e. prices) with probability  $1 - \delta$ . These  $N_c$  particles will be the input points for Gaussian processes  $(\ln(\alpha), \ln(\beta))$ . Having generated the set of challengers, generate  $\{\alpha_j, \beta_j\}_j^{N_c}$  at these points, with their covariance and mean using previously observed data as in (31). Generate a GP with mean 0, and smooth exponential kernel at these input challenger points, and use the inverse-cdf method to construct a Gamma-Gaussian Copula Process using these generated function values  $\{\alpha_j, \beta_j\}_j^{N_c}$ . This is a sample demand function, evaluated at the challengers. Multiplying this by the vector of challengers gives a sample revenue function, since the challengers act like values of the control (i.e. prices). One then acquires the maximiser of this sample revenue function and replaces the particle  $x_i$  with this maximiser.

Secondly, one attaches to each particle the weight  $\frac{u(x_i)}{\delta\varphi(x_i)+(1-\delta)u(x_i)}$ . These weights form unnormalised probabilities for each particle. Normalising these, one resamples the particles and adds a small amount of jitter to each particle to avoid particle attrition. Repetition of this process causes the particles to approximate a sample of the distribution of the phenomenon. The approximation improves arbitrarily as  $N_c \rightarrow +\infty$ , since in the limit one is maximising a

sample function itself.

This adaptation to the original MCMD method would have been extremely slow, but I was able to improve speed by conducting the operations on each particle in parallel on a GPU. This involved constructing a set of CUDA kernels that generates the appropriate Gaussian Processes, conducts Cholesky decomposition of their covariance matrices, and picks the maximiser of each sample revenue function.

In doing this I am relying on the validity of the method produced by Bijl et al.. I have not yet managed to prove convergence of this method, rather than relying on empirical experimentation, and future work would need to investigate the theoretical properties of the method. However it would necessarily involve investigation of the balance equations for a given particle, and their associated stationary distribution. It is clear from my empirical experimentation, however, that the distribution does indeed converge. My code is available upon request.

## 18 Results

The equipment I have available means that only a small sample ( $N = 120$ ) can be generated for a given sample path. The memory requirements on the GPU for larger samples are great since I must store matrices of dimension  $N, N - 1, N - 2, \dots$  etc. simultaneously. My GPU has only 4GB of graphics memory, and storing both the simulation data and fulfilling the requirements of running the computer puts a strain on these resources. With access to state-of-the-art equipment I expect to be able to provide much better empirical evidence of the operation of the algorithm.

Figure 3 presents sample paths of prices and times. It is clear that the expected result is shown here; ie, the time to wait responds to the exploitation and exploration parts of the learning process, and can explain stickiness in prices, as well as potentially explaining the short-term shifts in price to and from different regions of the control space. It is worth noting stage 38, where the price goes to a new region (around 15), and time spent there decreases accordingly to offset the potential losses from this exploratory phase. A similar event happens at stage 82. Admittedly, the sojourns into exploratory phases here do not necessarily appear to be quantitatively similar to those seen in the reference price literature (ie, times of around 2 weeks, corresponding to approximately  $t=20$  here) but we clearly see the qualitative features of shorter-length phases at prices away from the reference price.

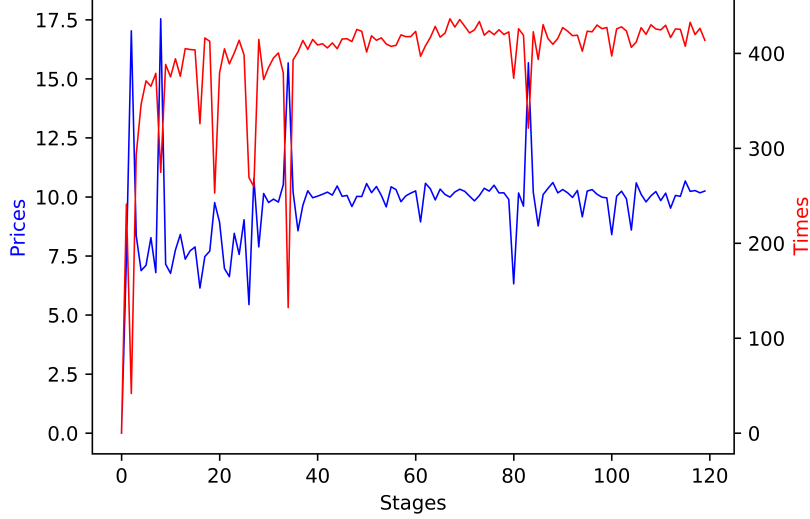


Figure 3: Sample Path of Prices and Times

## 19 Regret

Figure 4 present average regret for the operation of this policy. Regret is defined as the difference in outcomes, on a stage-by-stage basis, between operation of the truly optimal policy (ie, the full-information base case) and the policy actually followed by the agent. I contrast this to a policy of constant time, where I set  $t_i = 365 \forall i$ . It is clear that optimal timing policy still leads to sticky prices, while the timing component of the policy allows the agent to regulate the cost of exploratory phases. For the calculation of these average cumulative regret graphs, I take the prices and times and chosen under each policy, and place them into the full-information value function given by equation (28). To get an individual cumulative regret function, this value is then subtracted from the value acquired by spending the total time spent under optimal policy for 120 stages at the optimal price, which is given by equation (26) These functions are then averaged over the 100 repetitions I was able to run, and presented here. The dashed purple represents average cumulative regret from operating a constant time policy, whereas the green solid line represents average regret from following the optimal time policy. The distance between the two functions differs based on the value of the discount parameter  $\rho$ , since this affects the extent to which later rewards are discounted; when pursuing a constant time policy, later sojourns at better prices are far less valuable than earlier ones, so convergence to the optimal price is unable to compensate for the naivete of a constant time policy. Hence, I use a conservative value of  $\rho = 0.001$  so as not to overstate the improvement over a constant time policy

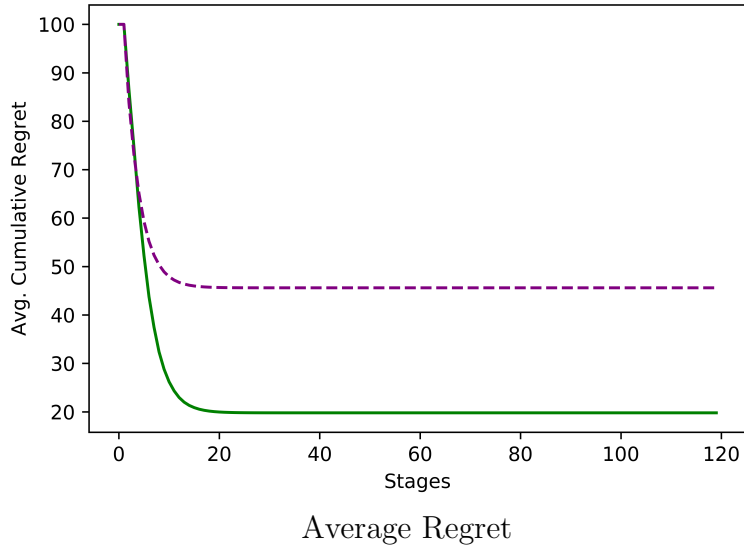


Figure 4: Purple - Constant Time Policy; Green - Optimal Time Policy

that is acquired by the optimal policy I describe.

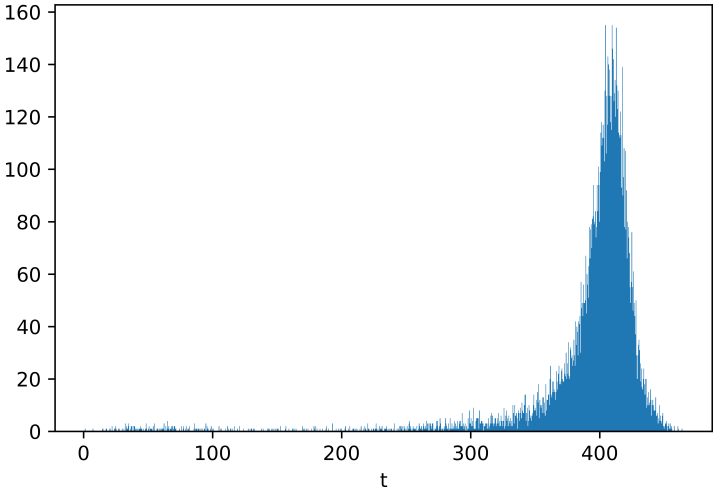
## 20 Time-To-Change Distribution

As for the times-to-change, the calibration of this model, with demand intensity given by:

$$\lambda = 20 - P$$

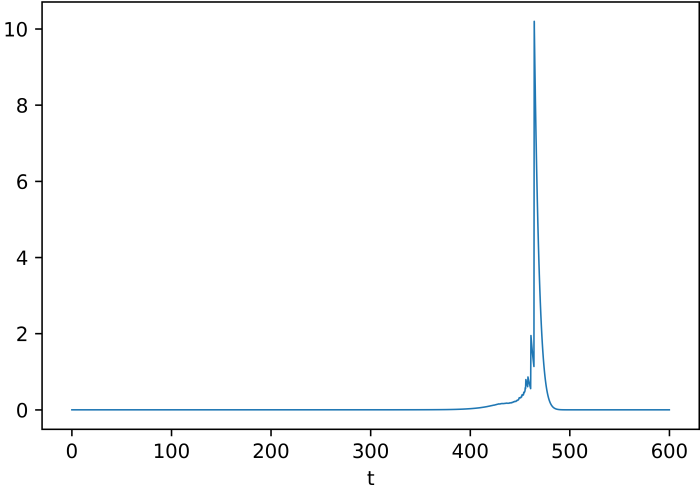
and  $\rho = 0.05$  imply a time-to-change distribution, and corresponding hazard function, as shown in Figures 5 and 6 respectively. We see that this simple learning mechanism (with an entirely stationary environment) is able to replicate the key features of the hazard function found by Nakamura and Steinsson (2008) for Services. In the model, optimal pricing (ie,  $P = 10$ ) implies an intensity of 10 in one unit of time; if one unit of time is a day, this is ten sales a day. If one considers  $t = 400$  to correspond to around 11 months, then this implies around 12 sales per day as the implied intensity. Further work would do well to look at the reasonable rate of sales for any given product at the level of the store-pricing manager, so as to ascertain how much this differs from real-world sales rates. The extent to which real-world daily sales correspond to sales intensities like those seen in the model is potentially a good baseline for judging the relative importance of the exploration/exploitation tradeoff in price-timing decisions, since that is the only mechanism operating in this model. I conjecture that reasonable extensions into non-stationarity

will allow the model to capture more closely the hazard functions found in the data for products.



Price Change Distribution

Figure 5



Price Change Hazard Function

Figure 6

## 21 Conclusion

I developed a new way of modelling firm behaviour in an unknown environment. Then, I used this to look at a heretofore unconsidered question; that of

optimal *time-to-wait* when posting prices and learning about demand. Solution of the model required adaptation and development of modern techniques in Machine Learning, as well as usage of parallelisation techniques and niche tools and resources. I built the programs I used to solve the model and acquire the distributions implied by the objects in the model by hand.

As a result, doing this I showed that prices are sticky to varying degrees due to the learning process. This may be able to explain to price stickiness in the real world and has implications for policy that does not take into account this component of the repricing process by firms.

The approach I have developed is novel and has opened up many avenues for further work. On the economics side, further work would do well to see what calibrations can lead to similar *quantitative* results regarding length of time-to-change, based on the level of the demand process intensity. Furthermore, work must be done to see how firms interact with each other in such a learning environment, and whether this could potentially be a way of describing a mechanism for convergence on Nash equilibria in oligopoly markets. Extensions could also include modelling of the environment when it is non-stationary. This leads to a much more complex problem as one must also take into account the increasing irrelevance of past information. As for the novel Machine Learning aspects, future research could include development of the theory around Copula processes, and their use in Thompson sampling and Bayesian optimisation in general; theory and guarantees on the MCMD algorithm for acquiring the distribution of the Thompson sample maximiser, and extension to handling different types of point process outside just the Poisson processes looked at here.

## References

- Aghion, P., Bolton, P., Harris, C., and Jullien, B. (1991). Optimal Learning by Experimentation. *The Review of Economic Studies*, 58(4):621.
- Agrawal, R. (1995). The Continuum-Armed Bandit Problem. *SIAM Journal on Control and Optimization*, 33(6):1926–1951.
- Araman, V. F. and Caldentey, R. (2006). Dynamic Pricing for Non-Perishable Products with Demand Learning. *Ssrn*, 57(5):1169–1188.
- Aviv, Y. and Pazgal, A. (2002). Pricing of short life-cycle products through active learning. *Under revision for Management Science*, pages 1–32.

- Barro, R. J. (1972). A Theory of Monopolistic Price Adjustment. *The Review of Economic Studies*, 39(1):17.
- Basu, K. and Ghosh, S. (2017). Analysis of Thompson Sampling for Gaussian Process Optimization in the Bandit Setting. *Available on the ArXiv* - <https://arxiv.org/pdf/1705.06808.pdf>.
- Bijl, H., Schön, T. B., van Wingerden, J.-W., and Verhaegen, M. (2016). A sequential Monte Carlo approach to Thompson sampling for Bayesian optimization. *Available on the ArXiv:1604.00169v3 [stat.ML]* - <https://arxiv.org/abs/1604.00169>.
- Bils, M. and Klenow, P. (2004). Some Evidence on the Importance of Sticky Prices. *Journal of Political Economy*, 112(5):947–985.
- Blackwell, D. (1965). Discounted Dynamic Programming. *Annals of the Institute of Statistical Mathematics*, 36:226—235.
- Brochu, E., M. Cora, V., and De Freitas, N. (2010). A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. *CoRR*, abs/1012.2.
- Caballero, R. J. and Engel, E. M. (2007). Price stickiness in Ss models: New interpretations of old results. *Journal of Monetary Economics*, 54(SUPPL.):100–121.
- Calvo, G. A. (1983). Staggered prices in a utility-maximizing framework. *Journal of Monetary Economics*, 12(3):383–398.
- Caplin, A. and Spulber, D. (1987). Menu Costs and the Neutrality of Money. *Quarterly Journal of Economics*, 102(4):703–726.
- den Haan, W. J. and Marcet, A. (1990). Solving the Stochastic Growth Model by Parameterizing Expectations. *Journal of Business & Economic Statistics*, 8(1):31.
- Farias, V. F. and Van Roy, B. (2010). Dynamic Pricing with a Prior on Market Response. *Operations Research*, 58(1):16–29.
- Gallego, G. and van Ryzin, G. (1994). Optimal Dynamic Pricing of Inventories with Stochastic Demand over Finite Horizons. *Management Science*, 40(8):999–1020.

- Gittins, J. C. (1979). Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, 41(2):148–177.
- Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257.
- Kehoe, P. and Midrigan, V. (2015). Prices are sticky after all. *Journal of Monetary Economics*, 75:35–53.
- Kingman, J. (1992). *Poisson Processes*. Clarendon Press.
- Lemaitre, F. and Lacassagne, L. (2016). Batched Cholesky Factorization for tiny matrices. *2016 Conference on Design and Architectures for Signal and Image Processing (DASIP)*, pages 130–137.
- Li, L. (1988). A stochastic theory of the firm. *Mathematics of operations research*, 13(3):447–466.
- Ljungqvist, L. and Sargent, T. J. (2004). *Recursive macroeconomic theory*. MIT Press.
- Lombardo, G. and Vestin, D. (2008). Welfare Implications of Calvo Vs. Rotemberg Pricing Assumptions. *Economics Letters*, 100(2):275–279.
- Maćkowiak, B. and Wiederholt, M. (2009). Optimal sticky prices under rational inattention. *American Economic Review*, 99(3):769–803.
- Mockus, J. (1989). *Bayesian Approach to Global Optimization*, volume 37. Springer Netherlands.
- Nadarajah, S. and Kotz, S. (2006). On the Laplace Transform of the Pareto Distribution. *IEEE Communications Letters*, 10(9):682–682.
- Nakamura, E. and Steinsson, J. (2008). Five facts about prices: A reevaluation of menu cost models. *Quarterly Journal of Economics*, 123(4):1415–1464.
- Pandey, S., Chakrabarti, D., and Agarwal, D. (2007). Multi-armed bandit problems with dependent arms. In *Proceedings of the 24th international conference on Machine learning - ICML '07*, pages 721–728.
- Petojević, A. (2008). A note about the Pochhammer symbol. *Mathematica Moravica*, (12-1):37–42.
- Rotemberg, J. (1982). Sticky Prices in the United States. *The Journal of Political Economy*, 90(6):1187–1211.

- Sims, C. A. (2003). Implications of rational inattention. *Journal of Monetary Economics*, 50(3):665–690.
- Sklar, A. (1959). Fonctions de répartition à n dimensions et leurs marges. *Publications de l'Institut de Statistique de l'Université de Paris*, pages pp 229–231.
- Snelson, E., Ghahramani, Z., and Rasmussen, C. E. (2004). Warped Gaussian Processes. In Thrun, S., Saul, L. K., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*, pages 337–344. MIT Press.
- Stokey, N. L. (2009). *The Economics of Inaction : Stochastic Control Models with Fixed Costs*. Princeton University Press.
- Thompson, W. R. (1933). on the Likelihood That One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. *Biometrika*, 25(3-4):285–294.
- Wilson, A. G. and Ghahramani, Z. (2010). Copula Processes. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 2 (NIPS'10)*. Curran Associates Inc., USA, 2460-2468.
- Wu, C. M., Schulz, E., Speekenbrink, M., Nelson, J. D., and Meder, B. (2017). Mapping the unknown : The spatially correlated multi-armed bandit. *bioRxiv*, pages 2–7.