Bounded Rationality in Macroeconomic Models: A Continuous-Time Approach

Chandler Lester

November 23, 2020

Preliminary

Link to Latest Version

Abstract

We reconsider optimal decision making in a continuous-time framework using adaptive learning, specifically shadow-price learning, in a real business cycle framework. The agent in our shadow-price learning model takes in information and updates their forecasts of future states and their decisions regarding choice variables. This agent makes decisions at high frequencies, which alters the volatility of the agent's parameter estimates and leads to smoother real-time convergence to the rational expectations equilibrium. We attribute the less volatile convergence to the agent's ability to alter their forecasts more often as new information becomes available. We also investigate alternative sampling methods where the data generating process is a higher frequency than the agent's observations; these sampling methods yield similar results to the case where the agent takes in all data points and are less computationally burdensome.

1 Introduction

Macroeconomic models often assume that both changes in the economy and agent's decisions occur at quarterly intervals, since data are most often available at that frequency. This approximation is bound to generate a loss of precision; since individuals make decisions about their employment, consumption, and investment at higher frequencies—arguably every day—despite less frequent economic data on these measures. In the economy, factors such as productivity and technology also change at a high frequency since computing power and innovations change rapidly. While discrete-time models provide useful insight into the economy, parameters that evolve quarterly and quarterly decision making can produce less accurate measures of volatility in real business cycle models (Aadland, 2001). One way to easily capture these high-frequency changes is continuous-time modeling, which assumes that the economic system is constantly evolving. Thus, building economic models in continuous-time provides an attractive alternative to discrete-time modeling.

We use a continuous-time real business cycle model combined with continuous-time adaptive learning dynamics, which allow our agent to improve their forecasts of key parameters and their optimal choices at high frequencies, to show that volatility of parameter estimates can be improved using high-frequency information. We demonstrate that the continuous-time model has less volatile parameter estimates as the agent's forecasts near rational expectations equilibrium (REE). Additionally, when examining these models near REE, the second moments of the continuous-time model came closer to matching relative moments from economic data than the model's discrete version.

We chose the continuous-time setting not just because of its ability to include high-frequency data and dynamics easily but also because it has a few key advantages over discrete-time and has recently gained popularity in macroeconomics. This class of models had been studied and examined in the past; however, continuous-time models did not gain the same prevalence as discrete-time modeling in economics due to their more complicated solution methods (Merton, 1971; Mirman, 1973; Mirrlees, 1971). With increased computing power

and more interdisciplinary research from applied mathematics and engineering, continuous-time macroeconomic models can now be easily solved even if they are involved. There are several different solution methods for these models ranging from viscosity solutions as in Kaplan et al. (2018), Achdou et al. (2020), and Ahn et al. (2018) to martingale methods as in Brunnermeier and Sannikov (2014).

As this literature enters the mainstream, it is necessary to modify macroeconomic modeling tools standard in discrete-time research. Thus far, continuous-time macroeconomic literature has focused almost exclusively on rational expectations, a modeling assumption wherein the agent knows key model parameters' values and distributions. We aim to extend an alternative to rational expectations, adaptive learning, to continuous-time literature. Adaptive learning models allow the agent to misspecify parameters and then—using data or knowledge that becomes available over time—update their estimates of these parameters. One complication with extending this technique is time-dependency in continuous-time models. For instance, the viscosity solution method and the martingale method both require the system to be either independent of time or if the system is time-dependent, it must be solved working backward from the steady-state (i.e., $t = \infty$). Neither of these methods creates an ideal environment for learning; solving the system from the end of time backward does not facilitate the agent's observation of new data. Additionally, the solution methods for continuous-time systems that do not depend on time lack the necessary feedback mechanisms for learning.

The insufficiency of feedback and observability in these methods necessitates the re-examination of continuous-time macroeconomic problems in a new environment. In this work and previous work, we have examined a linear-quadratic (LQ) framework that though independent of time, allows for the feedback necessary for agent-level adaptive learning. There are extensive studies of discrete LQ environments in economics and other fields, as outlined in Kendrick (2005). One of the LQ setting's key features is that the agent maximizes an objective function with a quadratic form, leading to linear first-order conditions. However,

most economic models are non-linear and do not fit into the traditional LQ format. Several papers, including Benigno and Woodford (2004, 2006, 2012), use discrete-time linearization techniques to recast non-linear models into the LQ setting. Benigno and Woodford (2012) examines various linearization frameworks and how to ensure accurate linearization, the LQ methods implemented in this paper carefully follow the dynamic programming approach outlined Benigno and Woodford (2012) and Hansen and Sargent (2013).

With few exceptions (Hansen and Sargent, 1991), the continuous-time LQ environment has been under-explored in the economic literature, despite its promise for building tractable and complex economic models. The field of computational finance has a considerable number of works on the continuous-time LQ environment, including Forsyth and Labahn (2007), Wang and Forsyth (2010), Huang et al. (2012), and Xie et al. (2008). In these papers, the optimization problems have a finite horizon, making these LQ settings distinct from the one we will outline in this paper. Additionally, some studies implement learning dynamics in linear optimal regulator problems; for instance, Vrabie et al. (2007) and Wang and Zhou (2019) focus on reinforcement learning in an LQ environment.

Recasting non-linear models into the LQ setting has a few key advantages. The LQ framework allows for the inclusion of many economic variables in a compact model, allowing economists to study complex economies with ease. Additionally, solving LQ problems tends to be less computationally intensive than solution methods for complex non-LQ economies. These advantages are particularly relevant in the context of rational expectations equilibrium, solving the REE of the models outlined in the following sections takes mere seconds using the LQ solution methods. This setting's solution method also does not depend on sparse grids or complicated differentiation schemes. The most important advantage of the LQ-setting, concerning adaptive learning, is that LQ methods contain important feedback mechanisms that allow us to understand the decisions an agent makes based on their observations; this is especially important in our shadow-price learning setting.

We aim to not only create a continuous-time setting where an agent learns how to fore-

cast parameter values accurately; we construct a framework in which an agent learns to forecast and make decisions optimally. An adaptive learning technique that accomplishes both of these goals is shadow-price learning. Shadow-price learning, or SP-learning, assumes the agent uses observations of state variables to understand how the states evolve and future shadow prices. Using these estimates, the agent modifies their behavior using updated shadow prices and the state transition dynamics through the LQ framework's built-in feedback mechanism. Since the state variables' evolution depends on the agent's choices, the agent's behavior influences the states they observe. Eventually, after gaining enough information, the agent in our SP-learning environment learns how to make decisions optimally and how to forecast future state values.

SP-learning allows us to examine better our agent's ability to learn to forecast and make decisions in our economy. The agent in our setting does not know the conditional distribution of key variables and faces uncertainty in our stochastic environment. It has been shown that discrete-time SP-learning can converge asymptotically to fully optimal decision-making in Evans and McGough (2018); we demonstrate that those same results hold in the continuous-time version of a real business cycle (RBC) model. We also compare the results of the continuous-time SP-learners to their discrete-time counterparts. Other works have explored various adaptive learning dynamics in RBC models, including Branch and McGough (2011), Eusepi and Preston (2011), and Mitra et al. (2013); this paper builds on this literature by re-examining learning in a continuous-time real-business cycle model.

We also explore data frequency dynamics in the continuous-time version of the model after inspecting the relationship between the discrete and continuous-time versions of the model and learning outcomes in these settings. Though often overlooked in macroeconomic models, data frequency impacts real-world decisions and macroeconomic outcomes. The importance of data frequency in estimating continuous-time financial models via maximum-likelihood methods has previously been studied in Aït-Sahalia (2010), which examines model estimation based on exact discrete-time estimates that take time-interval length into account.

Here we approach this problem using learning algorithms that rely on recursive least squares instead of the maximum likelihood approach.

As part of this exercise, we relax the assumption of continuous updating to better match empirical reality. Our approach assumes that the agent views the time and the economic changes as continuous occurrences and estimates a continuous-time version of our RBC model. Because real-world agents take in information at discrete time intervals and then, in turn, use this information to update their parameter estimates. Some additional considerations have been made regarding data observation. In particular, we examine how observing continuous processes at different frequencies impacts agents' responses and how information asymmetries can influence economic outcomes by comparing outcomes in an RBC model under learning with varying data collection frequencies and examining a version of the model wherein the agent collects data at varying frequencies. This question of how data availability can impact economic agents is of increasing importance since data today is available at increasingly higher frequencies. While quarterly data will likely be the most common frequency in macroeconomic data for some time to come, as macroeconomists move to include more micro-data and even big-data in macroeconomic analysis, we must consider how data frequency can impact our models.

Our work accomplishes several tasks; first, we demonstrate that the continuous-time learning algorithm does converge to rational expectations equilibrium. Then we closely contrast the outcomes of discrete and continuous-time learning models. Our comparison highlights the varying outcomes between these models, particularly the differences between the volatility of estimates and convergence rates in this setting. Additionally, we explore the linearization of simple macroeconomic models in continuous-time. There is sparse literature on this topic; some linearization of continuous-time macroeconomic models has been researched in other settings (Ahn et al., 2018). We also build on the work done in Evans and McGough (2018) and demonstrate that SP-learning can be modified for a continuous-time setting. Lastly, we examine how data collection can impact the agent's decisions in our

model's continuous-time version.

This paper proceeds as follows, section 2 outlines a simple real business cycle model in continuous-time and describes the SP-learning algorithm that the continuous-time agent uses to estimate parameters. A discrete-time version of this model is included in the appendix. After separately examining the discrete and continuous-time algorithms, we compare the rational expectations equilibrium of both settings and the learning outcomes in section 3. In this section, we compare the second moments of the discrete and continuous-models to the data; the continuous-time version of the model slightly outperforms the discrete version when examining standard deviations of key variables relative to the standard deviation of output. The fourth section examines the impact of data frequency on continuous-time models under learning. The final section concludes.

2 A Real Business Cycle Model—An LQ Approach

The framework used throughout this paper is that of a standard real business cycle model. We select this framework because our baseline model's simplicity allows us to add complex dynamics more easily. To efficiently use common SP-learning methods defined in Evans and McGough (2018), we need our RBC model to fit into a linear quadratic format. Accomplishing this involves linearizing our model objective function and recasting it into a quadratic form. The purpose of utilizing the LQ framework is to generate a model that can be solved recursively with clear and well defined connection between our agent's perceptions, or initial prediction for the value function, and the rational expectations equilibrium value. The continuous-time real business cycle model has a few key differences from a familiar discrete model. Our objective function maintains a similar form; it employs an isoelastic utility function that depends on labor and consumption. However, our discount factor is represented by an exponential function. Additionally, the processes that describe the evolution of capital and government spending now follow Brownian motions. Our household maximizes the

following objective function over consumption and labor input,

$$V(k_0, \tilde{z}_0) = \max_{c_t, k_t, h_t} \mathbb{E} \int_{t=0}^{\infty} e^{-\rho t} \left\{ \frac{c_t^{1-\sigma}}{1-\sigma} - \chi \frac{h_t^{1+\varphi}}{1+\varphi} \right\} \tag{1}$$

subject to the following conditions on consumption, productivity, and capital,

$$c_t + i_t = A k_t^{\alpha} (e^{\tilde{z}_t} h_t)^{1-\alpha} \tag{2}$$

$$d\tilde{z}_t = -\theta_{\tilde{z}} \tilde{z}_t dt + \sigma_{\tilde{z}} dZ_t \tag{3}$$

$$dk_t = (-\delta k_t + i_t) dt. \tag{4}$$

In equation (3) $\tilde{z}$ represents the logarithm of productivity and dZ_t is the increment of the Wiener process[1]. Firms in this economy maximize profits, using a Cobb-Douglas production function, $f(k_t, \tilde{z}_t) = k_t^{\alpha} (e^{\tilde{z}_t} h_t)^{1-\alpha}$. Under this production function the equilibrium rental rate on capital is $r_t = \alpha A k_t^{\alpha-1} (e^{\tilde{z}_t} h_t)^{1-\alpha}$ and the equilibrium wage is $w_t = (1-\alpha) A k_t^{\alpha} (e^{\tilde{z}_t} h_t)^{-\alpha} e^{\tilde{z}_t}$.

It is standard to take a dynamic programming approach to find the system's steady-state. Our value function problem takes the form of a Hamilton-Jacobi-Bellman (HJB) equation—the continuous-time analog of a Bellman equation. The HJB for the household's problem takes the following form,

$$\rho V(k_t, \tilde{z}_t) = \max_{c_t} \left\{ \frac{c_t^{1-\sigma}}{1-\sigma} - \chi \frac{h_t^{1+\varphi}}{1+\varphi} \right\} + V_k(-\delta k_t + A k_t^{\alpha} (e^{\tilde{z}_t} h_t)^{1-\alpha} - c_t) - \theta V_{\tilde{z}} \tilde{z}_t + \frac{1}{2} V_{\tilde{z}\tilde{z}} \sigma_{\varepsilon}^2$$

the terms V_k, V_z, and V_{zz} all represent partial derivatives of the value function $V(k, z)$ these terms are functions of k and z. The main difference between the HJB and a Bellman equation is how expectations are handled in continuous-time. Deriving expectations of the future value function requires using Itô's lemma since our state variables' evolution depends on continuous-time stochastic processes. Using the HJB we can find the non-stochastic

[1]One method of approximating dZ_t, is setting $dZ_t = \varepsilon_t \sqrt{dt}$ where $\varepsilon_t \sim N(0, 1)$ (Dixit, 1992). Thus the increments of the Wiener process are independent and Gaussian.

steady state values for our parameters by analyzing this system's first order conditions

$$\begin{aligned}
\rho V_k &= V_k(\alpha A k^{\alpha-1}(e^{\tilde{z}}h)^{1-\alpha} - \delta) \\
c^{-\sigma} &= V_k \\
\chi h^{\varphi} &= V_k(1-\alpha)Ak^{\alpha}(e^{\tilde{z}}h)^{-\alpha}
\end{aligned}$$

in this setting V_k is analogous to the shadow-price of capital as it measures the estimated value of a unit of capital. With our first order conditions defined, a numerical optimizer can be used to find the non-stochastic steady-state for our household's problem. Knowing the non-stochastic steady-state values of key parameters allows us to linearize our model about this point and simplifies the eventual LQ system we build in this work. After finding the system's non-stochastic steady state, we re-examine the planner's problem. First, we eliminate consumption, c_t, from our objective function by re-writing it as a function of capital, labor, investment, and productivity. This allows us to recast our maximization problem so that it only depends on state and control variables,

$$V(x_0) = \max_{x_t, u_t} \mathbb{E} \int_{t=0}^{\infty} e^{-\rho t} r(x_t, u_t).$$

where,

$$r(x_t, u_t) = \frac{1}{1-\sigma}[Ak_t^{\alpha}(e^{\tilde{z}_t}h_t)^{1-\alpha} - i_t]^{1-\sigma} - \chi\frac{h_t^{1+\varphi}}{1+\varphi}$$

The vectors x_t and u_t contain the state and control variables for the system, $x_t = (1, k_t, \tilde{z}_t)'$ and $u_t = (h_t, i_t)'$. Now that the maximization problem is in terms of the state and control vectors, we use a second-order linear approximation of $r(x, u)$ about the non-stochastic steady state to recast the maximization problem into a linear-quadratic format.

The second-order Taylor expansion about the steady-state—where $\bar{x}$ and $\bar{u}$ are the steady-

state values of x and u is standard and the same in continuous and discrete-time,

$$\begin{aligned} r(x,u) &= r(\bar{x},\bar{u}) + (x-\bar{x})' r_x(\bar{x},\bar{u}) + (u-\bar{u})' r_u(\bar{x},\bar{u}) \\ &\quad + \frac{1}{2}(x-\bar{x})' r_{xx}(\bar{x},\bar{u})(x-\bar{x}) + \frac{1}{2}(u-\bar{u})' r_{uu}(\bar{x},\bar{u})(u-\bar{u}) + (x-\bar{x})' r_{xu}(\bar{x},\bar{u})(u-\bar{u}) \end{aligned}$$

automatic differentiation can be used to compute the partial derivatives of $r(x,u)$. Once this is complete the problem is easily reformatting into a linear quadratic problem. This system does not gain any terms from Itô's lemma since the Taylor expansion is about a single point, instead of a stochastic process.

The maximization problem can now be put into a standard LQ representation. Our objective function now depends on several matrices, R is a 3×3 matrix that summarizes how our states impact the optimization problem directly, Q is a 2×2 matrix that describes how choice variables affect the system, and W is a 3×2 matrix that captures indirect effects (terms that involve both x and u). Below is the continuous-time LQ representation of our RBC model,

$$V(x_0) = \max_{u_t} \; -\mathbb{E} \int_{t=0}^{\infty} e^{-\rho t} (\hat{x}_t' R \hat{x}_t + \hat{u}_t' Q \hat{u}_t + 2\hat{x}_t' W \hat{u}_t)$$

where the state variables evolve according to

$$d\hat{x}_t = A\hat{x}_t + B\hat{u}_t + C dZ_t.$$

This problem is linearize about the steady-state, thus $\hat{x}_t = x_t - \bar{x}$ and $\hat{u} = u_t - \bar{u}$. The matrices R, Q, and W are equivalent to the following,

$$\underset{3\times3}{R} = \begin{bmatrix} r(\bar{x},\bar{u}) & \frac{1}{2} r_x(\bar{x},\bar{u}) \\ \frac{1}{2} r_x(\bar{x},\bar{u}) & \frac{1}{2} r_{xx}(\bar{x},\bar{u}) \end{bmatrix} \quad \underset{2\times2}{Q} = \begin{bmatrix} \frac{1}{2} r_{uu}(\bar{x},\bar{u}) \end{bmatrix} \quad \underset{3\times2}{W} = \begin{bmatrix} r_u(\bar{x},\bar{u}) \\ r_{xu}(\bar{x},\bar{u}) \end{bmatrix}$$

The matrices R, Q, and W are the same for both the discrete and continuous-time version of our model. Although the matrices that summarize our objective function remain the same

between these two settings, the matrices that describe the evolution of our state variables are not the exactly alike. In the continuous-time setting our matrix A is noticeable different from what we might expect from a discrete version of the model. This is because in continuous-time our system depends on changes in the state variables not on levels of the state variables at particular moments of time. The matrices that describe the evolution of our states are defined as follows,

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\delta & 0 \\ 0 & 0 & -\theta \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \quad C = \begin{bmatrix} 0 \\ 0 \\ \sigma_{\tilde{z}} \end{bmatrix}$$

this difference occurs because in the discrete version of our model we are measuring the level of $\hat{x}_t$ whereas in the continuous version we are calculating changes over increments of time.

To solve the value function problem we utilize a "guess-and-verify" approach by positing that the value function takes the form $V(x_t) = -x_t' P x_t - \xi$, where P is a positive semi-definite matrix. We then solve for P by substituting our supposed value function into the HJB equation

$$\rho x' P x + \rho \xi = \max_u \{x' R x + u' Q u + 2x' W u + 2x' P(Ax + Bu) + P(CC')\}. \tag{5}$$

As previously mentioned, one of the advantages of the LQ setting is its neat recursive solution methods. To implement this method we need to eliminate x and u from equation (5), this can be achieved by finding the system's policy function (a function that defines choices u based on states and model parameters). Using this system's first order conditions with respect to u we can define this system's policy function,

$$u = -(Q')^{-1}(W + PB)' x = -\tilde{F} x. \tag{6}$$

Combining the policy function in equation (6) and the system in equation (5) allows us to

eliminate both u and x from the system. With the state-independent version of our value function problem we formulate a recursive algorithm that solves for the value function matrix P (Anderson and Moore, 2007; Vrabie et al., 2007),

$$P_i = -(2\tilde{A}_i')^{-1}(\tilde{F}_i' Q^{-1} \tilde{F}_i + R - 2W\tilde{F}_i) \tag{7}$$

$$\xi_i = \rho^{-1}\text{trace}(P_{i-1}CC') \tag{8}$$

where $\tilde{A}_i = (A - B\tilde{F}_i - .5\rho)$, $\tilde{F}_i = (Q')^{-1}(W + P_{i-1}B)'$, i represents iterations of the recursive algorithm, and P_0 is set exogenously. Additionally, note that this system is formulated under the assumption that A is symmetric.

Several conditions must be met to ensure solutions to the algorithm are asymptotically stable and exist (Lewis, 1986; Anderson and Moore, 2007; Evans and McGough, 2018).

LQ.1 The matrix R is symmetric positive semi-definite and can be decomposed in $R = DD'$ by rank-decomposition, and the matrix Q is symmetric positive definite.

LQ.2 The matrix pair (A,B) is *stabilizable*—there exists a matrix $\tilde{F}$ such that $A - BF$ is stable, meaning the eigenvalues of $A - B\tilde{F}$ have modulus less than one.

LQ.3 The pair (A,D) is *detectable*—if y is a non-zero eigenvector of A associated with eigenvector μ then $D'y = 0$ only if $|\mu| < 0$. Detectability implies that the feedback control will plausibly stabilize any unstable trajectories.

The continuous-time recursive algorithm will have a unique solution provided that the conditions in LQ.1-LQ.3 hold true for this system's R, Q, A, and B matrices and the continuous-time policy function $\tilde{F}$.[2] Conveniently, the conditions outlined in LQ.1-LQ.3 also apply to the discrete-time version of this system; the only difference being that the discrete problem has a different policy function F. Now we turn to adding adaptive learning dynamics to our linearized RBC model.

[2]For a proof of this result, see Lewis (1986).

2.1 Shadow-Price Learning in the Continuous-Time RBC Model

The recursive solution method for our linearized model has a clear linkage between perceptions and actuality, which can be used to establish learning dynamics in this setting (Evans and McGough, 2018). Focusing on equation (7), we see a relationship between our agent's initial perception, P_{i-1}, and updated calculations of their value function matrix P_i. In this setting we define the agent's perceived value function as $V^P(x) = -x'T(P)x$ where $T(P)$ is our T-map, the formal link between perceptions and actuality in learning models.

The T-map, $T(P)$ is matrix function that maps an initial perception of shadow-prices, P, to the updated shadow-prices generated by our agent's choices. Our T-map's fixed point, $T(P^*) - P^* = 0$, is our learning model's equilibrium point, given that certain stability conditions hold. As shown by our derivation of the recursive algorithm in (7) and (8), the stochasticity of our system does not impact the solution to our value function problem. The solution for P is not impacted by the stochastic term C. Knowing this, we begin our explanation of the learning algorithm by focusing on our problem's non-stochastic version. The agent's perceived value function in the continuous-time non-stochastic setting is,

$$\rho V^P(x) = \max_u \{-x'Rx - u'Qu - 2x'Wu - 2x'P(Ax + Bu)\}. \tag{9}$$

The unique optimal control decision for perceptions P is given by,

$$u = -\tilde{F}(P)x = -(Q')^{-1}(W + PB)'x.$$

Our policy function is then substituted into equation (9) to find the T-map for our system,

$$T(P) = (2\tilde{A}')^{-1}(\tilde{F}'Q^{-1}\tilde{F} + R - 2W\tilde{F}) \tag{10}$$

here $\tilde{A} = A - \frac{1}{2}I\rho - B\tilde{F}$ and we again assume that $\tilde{A}$ is symmetric. The T-map above describes the mapping between perceptions and reality in a model without stochasticity.

The unique fixed point, P^* of this mapping, is the solution to our value function problem. This result has been proved in discrete-time and has been analytically demonstrated to hold for continuous-time models (Evans and McGough, 2018; Lester, 2020). As in the discrete-time case, the non-stochastic case mapping will yield the same fixed point as the T-map for the stochastic version of the system.

Thus far, the continuous-time learning dynamics assume that our agent knows information about the value function's quadratic nature and the values of the state transition dynamics. These assumptions are strict, it is unlikely an average person would understand the form of their utility function let alone assume that it was quadratic in nature. Instead it is more likely they estimate the system's shadow-prices using a simple linear forecasting rule. Equation (11) represents this simple linear forecasting model, where the agent predicts shadow prices μ_t using state values,

$$\mu_t = Hx_t + \varepsilon_t^{\mu}. \tag{11}$$

The matrix H is the shadow-price parameter matrix as we soon show it is directly related to our value function matrix P, in fact $H = -2P$ at rational expectations equilibrium. This forecasting rule can then be used to estimate the shadow-price parameters for our state variables, x.

$$\mathbb{E}[V_x(x)] = \mu^e = Hx$$

where μ^e is the updated estimate of μ. We use this forecasting rule to estimate the future expected utility in our HJB equation,

$$\rho V(x) = \max_u \{-x'Rx - u'Qu - 2x'Wu + (Hx)'(Ax + Bu) + \frac{1}{2}(H'CC')\}.$$

Our modified HJB equation provides insight into how our agent selects optimal choice variables under their forecast of shadow-price parameters. Again we use the policy function to

eliminate x and u from our system, to create a compact recursive solution method. We find our learning agent's policy function using the first-order conditions of the HJB,

$$u = -\frac{1}{2}(Q^{-1})'(2W - H'B)x = -\tilde{F}^{SP}(H, B)x.$$

Then to get the mapping from the PLM to the actual law of motion (ALM) we use the envelope condition,

$$\rho\mathbb{E}[V_x(x)] = \rho\mu^e = -2x'R - 2u'W + 2x'A'H + u'B'H. \tag{12}$$

we can rewrite (12) as,

$$\begin{aligned}\mu^e &= \rho^{-1}\{-2x'R - 2u'W + 2x'A'H + u'B'H\} \\ &= \rho^{-1}\big(-2R + 2H'A - (H'B - 2W)\tilde{F}^{SP}(H, B)\big)x \\ &= T^{SP}(H, A, B)x.\end{aligned} \tag{13}$$

The T-map in equation (13) will define the mapping between the agent's PLM in equation (11) and the actual law of motion (ALM) of the system. Our T-map allows us to model the boundedly rational behavior of an agent in this model, using a continuous-time analog to recursive least squares (RLS) that has been derived using the parallels between RLS and the Kalman filter (Lewis et al., 2007; Ljung and Söderström, 1983; Huarng and Yeh, 1992).

A brief discussion of recursive least squares methods is necessary before we define our SP-learning algorithm. To create a functional SP-learning algorithm, we need to define how the agent updates forecasts in the continuous-time setting. In discrete-time, this forecasting updating rule takes the form of RLS, an adaptive algorithm that allows an agent to update their parameter estimates as they acquire additional information. We begin in a discrete

setting with a simple linear regression model,

$$y_t = \theta' x_t + \varepsilon_t.$$

For this example y_t is a vector that contains our dependent variable, x_t is a matrix of independent variables (the information that agents' receive), θ is a vector of coefficients, and ε_t our error term, which is assumed to be a normally distributed white-noise process. The recursive least squares algorithm's objective is to update parameter estimates as new data points are observed by minimizing a weighted function of the summed of squared errors. In discrete-time this objective function takes a familiar form,

$$\phi_N(\theta) = \frac{1}{N} \sum_{t=1}^{N} \alpha_t [y_t - \theta' x_t]^2.$$

since this is a weighted least squares problem, α_t is a vector of weights set by the modeler. This vector of weights is related to the gain parameter present in most adaptive learning algorithms. Using this estimator we arrive at a simple recursive algorithm for estimates of the vector of parameters θ_t and the second moment of the data x_t,

$$\begin{aligned} \hat{\theta}_t &= \hat{\theta}_{t-1} + \gamma_t \mathcal{R}_t^{-1} x_{t-1} [y_t - \hat{\theta}'_{t-1} x_t], \\ \mathcal{R}_t &= \mathcal{R}_{t-1} + \gamma_t [x_t x'_t - \mathcal{R}_{t-1}], \end{aligned} \tag{14}$$

the parameter γ_t is the aforementioned gain parameter. The RLS algorithm allows for the agent to use an initial estimate of the coefficient matrix and second moment matrix, $\mathcal{R}_t$, and then update their estimates as they acquire additional information.

RLS takes a similar form in the continuous-time setting; however, our algorithm becomes a system of stochastic differential equations. We begin with a stochastic differential equation

instead of the linear regression model,

$$dy_t = \theta' x_t dt + dZ_t$$

the term dZ_t represent the increment of the Wiener process as we've described previously. The RLS estimator now takes the form of

$$\phi_N(\theta) = \frac{1}{N} \int_{\tau=1}^{N} \alpha_\tau [dy_\tau - \theta' x_\tau d\tau]^2.$$

The continuous-time version of RLS is then found using parallels between recursive least squares and other filtering methods (Sastry and Bodson, 1989). We use a constant gain algorithm in this work, thus below is a version of RLS where γ_t is set as a constant. Implementing this version of RLS means that individuals put equal weight on all observations and expect some noise in their parameter estimates,

$$\begin{aligned} d\hat{\theta}_t &= \frac{1}{1-\gamma_t} \mathcal{P}_t x_t [dy_t - \hat{\theta}'_{t-1} x_t dt], \\ d\mathcal{P}_t &= \frac{1}{1-\gamma_t} [\gamma_t \mathcal{P}_t - \mathcal{P}_t x_t x'_t \mathcal{P}_t] dt. \end{aligned} \tag{15}$$

It is most common in continuous-time literature to use the matrix $\mathcal{P}_t$, the covariance matrix, to avoid matrix inversion. The "recursive least squares filter" as it often called in engineering literature, is strikingly similar to the system in (14). By observation one can see that (15) is essential the derivative of the system in (15) with respect to time. For a full derivation of the continuous-time RLS system, please see the appendix or Goodwin and Mayne (1987).

With an established background in Shadow-Price learning dynamics and continuous-time recursive least squares, we can now outline an algorithm that models an agent's bounded rationality in our framework. In this system, the agent's policy function $\tilde{F}$ impacts the choices they make, and the future states they observe. Thus, our learning algorithm includes updates to the state variable impacted by the agent's choices and subsequent updates to

agent's choice and forecasts based on the current state of the economy.

$$
\begin{aligned}
dx_t &= Ax_t dt + Bu_t dt + CdZ_t \\
d\mathcal{P}_t &= \frac{1}{1-\gamma_t}(\gamma_t \mathcal{P}_t - \mathcal{P}_t x_t x_t' \mathcal{P}_t)dt \\
dH_t' &= \frac{1}{1-\gamma_t}\mathcal{P}_t x_t(\lambda_t - H_t x_t)' dt \\
dA_t' &= \frac{1}{1-\gamma_t}\mathcal{P}_t x_t(dx_t - Bu_t dt - A_t x_t dt)' \\
u_t &= -F^{SP}(H_t, B)x_t = -\frac{1}{2}(Q')^{-1}(2W - H_t B)' x_t \\
\lambda_t &= T^{SP}(H_t, A_t, B)x_t \\
\gamma_t &= \kappa(t+N)^{-\nu}.
\end{aligned}
\tag{16}
$$

In this algorithm $\mathcal{P}_t$ is the covariance matrix, unlike the discrete algorithm that uses $\mathcal{R}_t$, an approximation for the second moment, $\mathcal{P}_t$ can tend toward zero, something we need to be careful of in our setting (Sastry and Bodson, 1989). The use of $\mathcal{P}_t$ reduces the computational burden of taking the matrix inverse and is more in-line with the continuous-time Kalman filter notation. The gain parameter, γ_t, will again be assume to be constant with $\kappa = 0.01$ and $\nu = 0$.

2.1.1 Continuous-Time Learning Results

Now that we have defined an agent's bounded rationality in this setting, we can examine our learning algorithm's convergence. Before we can examine the dynamics of the learning model, the model parameters must be set. The continuous-time model was selected to align with parameters from discrete-time literature. To select appropriate values for some of the parameters, such as the discount factor, we consulted Kaplan et al. (2018). For the continuous-time SP-learning algorithm, it is necessary to approximate the time-step dt. We selected 1/100. Since the discrete-time version of the model is calibrated based on quarterly

data, $dt = 1/100$ indicates that our agent updates parameters at least once a day.[3] The final parameters, $\sigma_{\tilde{z}}$ and $\theta_{\tilde{z}}$ were set in accordance with discrete time literature. The process for $\tilde{z}_t$ defined in equation (3) is the continuous-time analog to an AR(1) process, thus there exist many comparisons of the two. Basing our estimates off a discrete model with an auto-regressive term of 0.895 and white-noise term with a standard deviation of 0.01, the parameters of the continuous-time model are set to $\theta_{\tilde{z}} = 0.105$ and $\sigma_{\tilde{z}} = 0.01$.[4] Table 1 summarizes the parameter values for the continuous-time model.

Table 1: Continuous-Time Parameter Values

Description		**Parameter Value**
A	Total Factor Productivity	1.0
ρ	Discount factor	0.01
σ	Intertemporal elasticity of subst.	1.0 (log utility)
φ	Frisch elasticity of labor supply	-1.0 (log utility)
χ	Disutility of labor	1.75
α	Capital share	1/3
δ	Depreciation rate	0.025
$\theta_{\tilde{z}}$	Drift parameter for tech.	0.105
dt	Approximation of time-step	1/100
$\sigma_{\tilde{z}}$	Standard deviation for tech.	0.01

After finalizing key parameter values, we focused on the initial values for the learning algorithm. The misspecification used in this setting varied from the discrete-time version. Here A and H were set to small negative constants times identity matrices. Initial values for x_0 and u_0 were, again, set near steady-state values. The second-moment matrix $\mathcal{P}$ was

[3] Approximately 1.09 times a day. Assuming 91 days in a quarter.

[4] With our naive estimation approach, the limiting distributions of the discrete and continuous-time models have approximately the same variance (Posch et al., 2011). For the discrete-time case $\text{Var}(x) = \frac{\sigma_z^2}{1-\theta_z^2} = \frac{(0.01)^2}{1-0.895^2} \approx 0.0005$. While in the continuous-time setting $\text{Var}(x) = \frac{\sigma_{\tilde{z}}^2}{2\theta_{\tilde{z}}} = \frac{(0.01)^2}{2 \cdot 0.105} \approx 0.0005$.

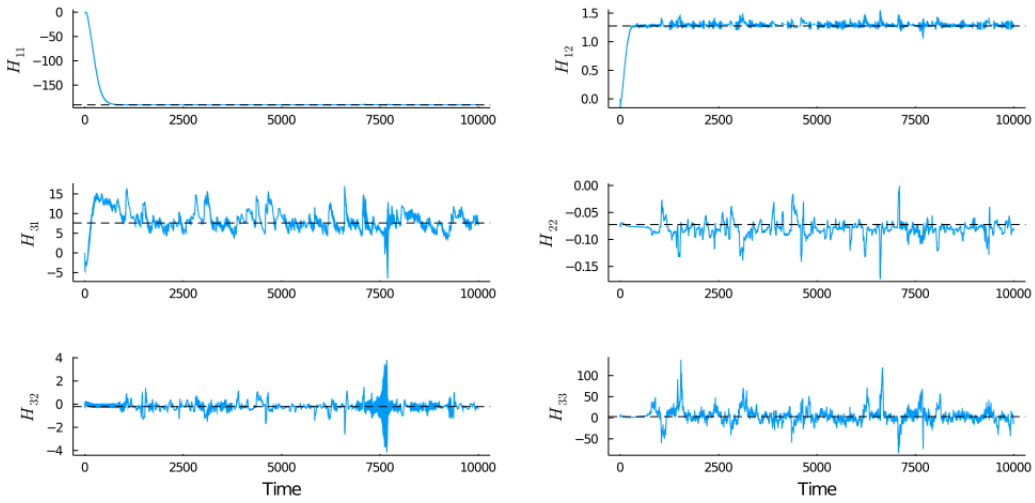

Figure 1: Convergence of Shadow-Price Parameters

initialized based on initial values of x_0. Misspecification in the continuous-time was set to ensure stability with the continuous-time T-map and policy function. While the SP-learning algorithm's initialization does not need to be near the REE, it is best if the initial policy is stable, meaning the T-map's derivative has eigenvalues within the unit root. Additionally, the agent in this setting understands the basic structure of the transition matrix A and does not use the constant in estimating parameters; instead, they estimate the technology and capital processes' parameters separately.

Simulations of the model were run for the equivalent $10,000$ discrete-time periods so the agents were able to update their forecasts over $(100 \times 10,000)$ iterations, since $dt = 1/100$. Examining figure 1 we see that in the continuous-time model the agent's estimates converge quickly and fluctuate around their REE values. At the end of $10,000$ periods the agent in continuous-time model forecasts a shadow-price parameter matrix that is a distance of 11.25 from REE according to the matrix norm measurement. The agent also updated their estimates of the state transition matrix A over these 50,000 periods. The distance between the agent's estimate of A and the true transition matrix, measured using matrix norms, is 0.012 after only 10,000 periods.

Thus far, we have demonstrated that the continuous-time real business cycle model con-

verges to REE under our SP-learning algorithm. Next, we compare these models' learning outcomes to understand the differences between bounded rationality in these settings. The following section of this paper examines parameter values and their distances from REE values, the volatility of these models' estimations, and the second moments of key variables, as is common in real business cycle literature.

3 Comparing Discrete and Continuous-Time Systems

We now compare the REE values to the learning models' outcomes, with the models initialized "far-away" from the rational expectations equilibrium. As measured by state and choice variable values, the economic outcomes of the discrete and continuous-time models closely match REE values after 50,000 periods; however, the continuous-time model comes closer to reaching the REE for shadow-price parameters. Additionally, the continuous-time model's shadow-price parameter estimates display less volatility than the estimates for the discrete-time model, implying that the continuous-time learning estimates exhibit more stability than their discrete counterparts.

For some reference, the rational expectations equilibrium values of the discrete and continuous-time shadow-price parameter matrices are given in equations (17) and (18). Solutions for the steady-state values of key variables and the value function matrix H are similar for the discrete and continuous-time versions of the real-business cycle model.

$$H^*_{\text{Discrete}} = \begin{bmatrix} -190.764 & 1.29026 & 7.67191 \\ 1.29026 & -0.0753887 & -0.210606 \\ 7.67191 & -0.210606 & 2.73081 \end{bmatrix} \tag{17}$$

and

$$H^*_{\text{Continuous}} = \begin{bmatrix} -190.642 & 1.2759 & 7.6087 \\ 1.2759 & -0.0724069 & -0.212827 \\ 7.6087 & -0.212827 & 2.64364 \end{bmatrix} \tag{18}$$

The fact that the continuous-time matrix is so close to the discrete-time version (the matrix norm of the discrete solution minus the continuous one is 0.386) solidifies that these matrices are the equivalent solutions to their respective problems.

There are some minor computational gains when solving for the rational expectations equilibrium in continuous-time. The discrete version of the recursive LQ algorithm presented which is presented in the appendix converges in 0.009413 seconds and 1,333 iterations for the model with government spending. While the continuous version, from section 2, converges in 0.000316 seconds and 11 iterations.[5] Additionally, the discrete-time algorithm used 10.764 MiB of memory, while the continuous-time version only required 0.2849 MiB. The continuous-time LQ algorithm's speedier convergence is not observable by the programmer in this instance but could have serious impacts on a more complex economy with more than four state variables.

3.1 Comparing Learning Outcomes

Comparing learning outcomes between the discrete and continuous-time models is difficult since there are many factors to consider, such as the distance between REE and the initial specifications and how the initial covariance/second-moment matrix is set. Since the learning algorithms both implement constant gain, the most accurate method of comparing learning outcomes in both models is to examine the learning parameters over the last $1,000$ periods of the learning iterations.

For a better comparison between the discrete and continuous-time cases, we have only included points from the continuous models that occurred at the end of each discrete period,

[5]Run-times were calculated using the instructions in Julia documentation. This requires compiling functions beforehand for accurate measurements.

so the continuous-time mean values and standard deviations are calculated using the same observation size as the discrete-case. Without this sampling scheme, the continuous-time standard deviations would still be almost the same any change in these values occurred at the third (or higher) decimal place. Standard deviations of state and choice variables are included in the table, in parentheses underestimated parameter values.

In our shadow-price learning algorithm, the agent forecasts two key objects, the state transitions matrix, and their shadow prices; these state-transition dynamics and shadow-prices impact the system's evolution via the choices our agent makes regarding investment and hours worked. We first examine the impact of learning on key parameters' values, such as investment and capital, before more closely examining shadow-price parameters. Table 2 lists the REE equilibrium values of economic variables for the continuous-time and discrete-time models as well as the averaged learning outcomes over the last 1,000 periods.

Table 2: Steady State Values and Learning Outcomes

	Discrete		**Continuous**	
Variable	**REE Value**	**Learning**	**REE Value**	**Learning**
Labor	0.333	0.333	0.333	0.333
Investment	0.244	0.243	0.245	0.245
Capital	9.749	9.758	9.797	9.805
Consumption	0.783	0.783	0.784	0.785
Wages	2.054	2.055	2.057	2.059
Rental Rate on Capital	0.035	0.035	0.035	0.035

Although the differences between the discrete-time and continuous-time steady state values are similar they highlight a few key differences between the systems. In the continuous-time system, steady-state wages are slightly higher, as is an investment. This is likely necessary to help offset continuous-time discounting. Learning outcomes between these two models are similar; however, the discrete version of our learning model appears to underes-

timate the level of capital. This likely comes from the shadow-price parameter estimates as these impact the agent's investment choices, which in turn impact capital accumulation.

Next, we examine the shadow-price parameters. The matrix norm between the agent's forecast of H and the REE was 2.35. in the continuous case and 2.42 in the discrete case. In the discrete case, the matrix norm between the initial guess H_0 and the true value was 192, and in the continuous version, that same measure was 191.

To analyze the difference between shadow-price forecasts in the continuous and discrete model, we again examine the last 1,000 periods of both learning algorithms. Table 3 contains the average learning outcome over the last 1,000 periods, the standard deviation of the parameter over the last 1,000 periods, and the rational expectations equilibrium values.

Table 3: Shadow-Price Parameter Outcomes

	Learning Outcome		**REE Value**	
Variable	**Discrete**	**Continuous**	**Discrete**	**Continuous**
Constant	-189.909	-190.564	-190.764	-190.642
	(0.026)	(0.0018)		
Capital	-0.077	-0.075	-0.075	-0.072
	(0.0002)	(0.0000)		
Productivity	2.544	2.548	2.731	2.644
	(4.64)	(0.004)		

Overall the continuous-time version of the model has more accurate measures of shadow-price parameter values and lower standard deviations for these parameter estimates.

3.2 Comparing the Accuracy of the Models' Second Moments

After examining the parameter estimates under SP-learning dynamics, a few questions arise about the impact of continuous-time on the model's second moments. In real business cycle literature, it is common to examine the theoretical model's second moments and compare

them to economic data (Plosser, 1989; Hansen and Wright, 1992; Romer, 1996). In this exercise, we compare the outcomes of the discrete and continuous-time learning models to second moments from data that have been detrended using the HP-filter.

Economic data from 1960-2019 on GDP, consumption, investment, wages, and hours worked was collected using the FRED database. Then using the HP-filter and logarithmic transformation, we detrended the data. We simulated the same model used in the previous sections to compare the second moments between the data, discrete, and continuous-time systems. The calibration of our model was changed the stochastic process for technology in this version has an auto-correlation term of 0.99 in the discrete case and 0.01 for the continuous-time case. In both instances, the standard deviation of the white-noise process was also set to 0.01. In the continuous-time setting time intervals, dt are approximated as $1/100$. This approximation of dt means that the agent updates their estimates about once a day since the discrete model is calibrated using quarterly data.

Each model's economy was simulated for 240 periods (the same number of periods present in the data). We ran these simulations one thousand times for the discrete and continuous-time models with learning dynamics and applied the logarithmic transformation and HP-filter to these 1,000 series. We report standard deviations and correlations averaged over all 1,000 simulations in table 4. Since the variables we measured are primarily flow variables, the continuous-time model's points were aggregated by integrating information to compare with the discrete model. Table 4 displays the standard deviations of values from the data and the theoretical models, along with the correlations between key variables and output.

Table 4: Second Moments and Autocorrelations of Key Economic Variables

	Standard Deviation*			Correlation with Output		
Variable	**Data**	**Discrete**	**Continuous**	**Data**	**Discrete**	**Continuous**
Output	1.43%	1.02%	1.02%	1.00	1.00	1.00
Consumption	0.510	0.485	0.487	0.748	0.971	0.773
Investment	2.880	2.757	2.877	0.799	0.989	0.972
Hours	0.646	0.365	0.456	0.650	0.982	0.854
Wage	0.660	0.650	0.624	0.172	0.994	0.925

*standard deviations for variables other than output are measured relative to output

The continuous-time version of the model matches the relative second moments from the data slightly better for consumption, investment, and hours worked. While the continuous version of the model still overestimates economic variables' procyclicality, it does so by less than the discrete version.

4 Learning and Data Frequency

Now that we have outlined methods and results for continuous-time learners, we examine outcomes when an agent takes in information over larger intervals. In this section, the economy that the agent participates in is continuous, and state variables update continuously as well; however, the agent is only capable of taking in observations at lower frequencies. This setting parallels the real-world where we may believe that economic factors like productivity or even GDP are continually updating. However, due to our limited ability to take in information and data availability restrictions, we cannot constantly update our estimates of these parameters. Our state variables evolve according to a continuous-time process we approximate as updating daily—$dt \approx 1/100$. We examine three different agents in this setting. The first updates information weekly, the second bi-weekly, and the third every day. An essential aspect of our agent's forecasts is that they understand that they are

approximating a higher frequency process, i.e., the weekly updater understands that they are using weekly data and includes that information in their estimations.

4.1 Learning under Varying Data Frequencies

As previously mentioned, the agents in this section exist in an economy where variables are continuously evolving. They maximize their utility subject to the continuous-time RBC model in section 2. However, the agents in this setting do not continuously update their parameters. Instead, they only observe data at specific time intervals, and they know they are approximating a continuous-time system using this discrete data. Considering this, they use Δ—the time step of their discrete observations—in their forecasts to approximate dt. The learning algorithm implemented by these agents is similar to the continuous-time algorithm, with a few key changes.

$$
\begin{aligned}
dx_t &= Ax_t dt + Bu_t dt + CdZ_t \\
\Delta\mathcal{P}_t &= \frac{1}{1-\gamma_t}(\gamma_t\mathcal{P}_t - \mathcal{P}_t x_t x_t'\mathcal{P}_t)\Delta \\
\Delta H_t' &= \frac{1}{1-\gamma_t}\mathcal{P}_t x_t(\lambda_t - H_t x_t)'\Delta \\
\Delta A_t' &= \frac{1}{1-\gamma_t}\mathcal{P}_t x_t(x_t - Bu_t\Delta - A_t x_t\Delta - x_{t-\Delta})' \\
u_t &= -F^{SP}(H_t, B)x_t = -\frac{1}{2}(Q')^{-1}(2W - HB)'x_t \\
\lambda_t &= T^{SP}(H_t, A_t, B)x_t \\
\gamma_t &= \kappa(t+N)^{-\nu}.
\end{aligned}
\tag{19}
$$

The state variables for this system still evolve continuously but now they are observed at distinct periods of time. Meaning the agent will observe, $x_1, x_{1+\Delta}, x_{1+2\Delta} + \ldots x_\tau$ where τ represents the end period of the model.

We examine three different agents that observe data at the three varying frequencies in this system. For ease, we assume that our state variables evolve almost daily and ap-

proximate $dt = 1/100$. This is consistent for all agents in this section and is the same approximation of dt used in the previous sections. For these versions of the model, the same parameterization from 2 is recycled; however, the three models explored in this section have an additional parameter Δ. The new parameter Δ represents the intervals at which agents take in additional information, whereas dt is the actual time interval for the data generating process.

We use three specifications, one where $\Delta = 1/25$ for individuals that update their estimates every four days, or about twice a week, another with $\Delta = 1/50$ to represent weekly up-daters, and the last has $\Delta = 1/100$ meaning that the agent observes every point in the true data generating process. In all three of these cases, agents updated thier shadow-price forecasts over 10,000 discrete-time periods (in this case, over 10,000 quarters). Initially, this exercise examines the differences in learning dynamics over varying time intervals. However, learning outcomes are nearly identical in all three cases– likely because we did not constrain the number of learning iterations and gave each type of agent 10,000 periods of data. The only major difference between these specifications was run-time. Measurements for matrix norms and the standard deviation of matrix norms were measured using the mean matrix norms and standard deviation of the matrix norm over the last $1,000$ discrete-time periods.

Table 5: Continuous-Time Learning Results under Varying Data Frequencies

dt	**Specification**	**Matrix Norm**	**Norm Std.**	**Computational Time**
$dt = 1/364$	$\Delta = 1/364$	12.86	15.37	187
	$\Delta = 1/91$	13.26	15.56	58
	$\Delta = 1/52$	13.34	15.49	19
	$\Delta = 1/26$	13.07	15.12	12
dt=1/100	$\Delta = 1/100$	13.28	16.96	36
	$\Delta = 1/50$	13.25	16.88	12
	$\Delta = 1/25$	13.46	17.13	7

Table 5 demonstrates that one the short comings of the continuous-time learning algorithm, long run-times, can be minimized by implementing different sampling frequencies. This table also includes extra specifications using $dt = 1/364$ to provide additional evidence on how sampling frequencies and smaller approximations for dt can reduce computational time.

5 Conclusion

Rational expectations is a powerful modeling tool that allows economists to compute equilibrium outcomes efficiently. However, as we look to micro-foundations, that assumption of rational expectations is far too strict. It is unlikely that individuals understand the evolution and distribution of productivity or the capital stock. It is also unlikely that they understand how to fully optimize when making decisions.

The adaptive learning literature has relaxed both of these assumptions; in this paper, we relax a third assumption: that agents make decisions intermittently at fixed time intervals. Previous literature has approached optimization and forecasting as a discrete problem. We introduce a continuous-time shadow-price learning algorithm that converges to the rational expectations equilibrium without imposing unrealistic assumptions.

Not only does this result match the point estimates in the continuous-time rational expectations model, it improves the estimates volatility when compared against discrete-time models and economic data. This result supports the outcomes of continuous-time rational expectations models while demonstrating that convergence in the continuous-time setting is not the same as convergence in the discrete-time case. Our continuous-time model displays less volatile shadow-price parameter estimates and smoother convergence (measured using matrix norms) of the shadow-price parameter matrix to REE values. This decreased volatility demonstrates that when agents gain more information more rapidly and have the ability to update their forecasts more frequently, they will make smaller, less reactionary

updates to their predictions and choices.

Furthermore, we demonstrate that the continuous-time version of the model can provide improvements when matching the data's second moments. Since the continuous-time model more closely matches the data and displays less volatile convergence to the REE values of shadow-price parameters, we can conclude that our continuous-time model captures important dynamics that the discrete version of our model does not.

Though helpful in demonstrating the continuous-time framework, continually updating is unlikely for agents and computationally burdensome for modelers. In a refinement exercise, we introduce an alternative sampling method that allows the continuous-time agent to sample data observed at high frequency and update their forecasts less frequently. This alternative sampling method results in faster computational time and similar parameter estimates.

Since the continuous-time shadow-price learning algorithms presented in this work converge to REE, it would be simple to conclude that we should model agents as fully rational and fully optimizing or as discrete decision-makers. However, in reality, agents are not infinitely lived, and they may experience structural changes that will cause them to re-evaluate their decisions. Additionally, there are key differences between convergence in continuous and discrete settings. These dissimilarities show that the agent more gradually converges to REE in continuous-time and makes less volatile choices as they near convergence. It seems that continuous-time agents make more stable decisions and smaller updates to their choices. The proposed framework improves on two desirability properties of agent optimization models: predictive precision and assumption parsimony.

Continuous-time adaptive learning literature is limited, and there is much work to be done on this topic. We intend to explore extensions to this work, including further improvements to the shadow-price learning algorithm. Improvements to our basic shadow-price learning algorithms likely exist; unlike the discrete version of the algorithm, our problem is a system of differential equations with no matrix inverse necessary. Therefore, we could attempt to simplify our problem using matrix algebra. We also would like to apply this method

to a wide range of macroeconomic and financial models. For instance, many portfolio selection problems are already in the LQ format; thus, our shadow-price learning framework could be easily extended to these models. Additionally, we would like to find applications for continuous-time adaptive learning algorithms to economic models outside of the linear-quadratic format.

A The Discrete-Time Model

We examine decision making under bounded rationality using a real business cycle model with taxation on wages and capital. In this RBC model households maximize their utility according to the following function of consumption and labor,

$$V(k_0, z_0) = \max_{c_t, k_{t+1}, h_t} \mathbb{E}_t \sum_{t=0}^{\infty} \beta^t \left\{ \frac{c_t^{1-\sigma}}{1-\sigma} - \chi \frac{h_t^{1+\varphi}}{1+\varphi} \right\}. \tag{20}$$

This maximization problem is subject to the following constraints on consumption and capital accumulation

$$c_t + k_{t+1} = A k_t^{\alpha} (z_t h_t)^{1-\alpha} - \delta k_t \tag{21}$$

$$k_{t+1} = (1-\delta) k_t + i_t. \tag{22}$$

Firms in this economy seek to maximize profits according to their costs and production capabilities with a Cobb-Douglas production function, $f(k_t, z_t h_t) = A k_t^{\alpha} (z_t h_t)^{1-\alpha}$. Productivity, z_t, evolves according to

$$\log(z_t) = \theta_z \log(z_{t-1}) + \varepsilon_t^z. \tag{23}$$

and $\varepsilon_t^z \sim N(0, \sigma_z^2)$. The LQ format necessary for implementing SP-learning in our social planner's problem must be linearized about the steady state, thus we must first find the non-stochastic steady state of the system. We use these steady state values to build the LQ version of the model by recasting the objective function to depend solely on state and choice variables than then re-writing this new objective function as a second degree Taylor expansion about the system's steady state (Ljungqvist and Sargent, 2012).

To use the LQ framework we want need the RBC model in the following form

$$V(x_0) = \max_{x_t, u_t} \mathbb{E}_t \sum_{t}^{\infty} \beta^t r(x_t, u_t).$$

where u_t is a vector of the agent's choice variables and x_t is a vector of state variables. These state variables evolve according to the following process,

$$x_{t+1} = Ax_t + Bu_t + \varepsilon_t$$

Reformatting the problem is accomplished using the modified equation for consumption. Using this, the objective function depends solely on capital, labor, investment, and technology

$$r(x_t, u_t) = \frac{1}{1-\sigma}[Ak_t^{\alpha}(z_t h_t)^{1-\alpha} - i_t - g_t]^{1-\sigma} - \chi\frac{h_t^{1+\varphi}}{1+\varphi}.$$

The vectors x_t and u_t contain the state and control variables for the system respectively—$x_t = (1, k_t, \log(z_t), g_t)'$ and $u_t = (h_t, i_t)'$. Now that our maximization problem is rewritten to depend on x_t and u_t, we use a second order linear approximation of $r(x_t, u_t)$ about the non-stochastic steady state to reformat the maximization problem.

The second-order Taylor expansion about the steady-state where $\bar{x}$ and $\bar{u}$ are the steady-state values of x and u, can be found using automatic differentiation to compute the partial derivatives of $r(x, u)$. Once this is complete the problem is easily reformatted into a linear-quadratic optimization problem,

$$V(x_0) = \max_{u_t} \; -\mathbb{E}_t \sum_{t=0}^{\infty} \beta^t (\hat{x}_t' R \hat{x}_t + \hat{u}_t' Q \hat{u}_t + 2\hat{x}_t' W \hat{u}_t)$$

where the state variables evolve according to

$$\hat{x}_{t+1} = A\hat{x}_t + B\hat{u}_t + C\varepsilon_t$$

here $\hat{x}_t = x_t - \bar{x}$ and $\hat{u} = u_t - \bar{u}$.

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & (1-\delta) & 0 \\ 0 & 0 & \theta_z \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \quad C = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

The matrices that define the objective function—R, Q and W– will be the same as before. These matrices combined with the matrices that define the state variables' evolution—A, B, and C—can solve the value function problem for the system above

$$V(\hat{x}_t, \hat{u}_t) = -\hat{x}_t' R x_t - \hat{u}_t' Q \hat{u}_t - 2\hat{x}_t' W \hat{u}_t + \beta \mathbb{E}_t V(x_{t+1}, u_{t+1}).$$

To get a closed-form solution to this problem we posit that the value function takes the form $V(x_t) = -\hat{x}_t P \hat{x}_t - \xi$, where P is a positive semi-definite matrix that summarizes the evolution of value function Hansen and Sargent (2013). Thus, we can rewrite the problem above as

$$-\hat{x}_t P \hat{x}_t - \xi = -\hat{x}_t' R \hat{x}_t - \hat{u}_t' Q \hat{u}_t - 2\hat{x}_t' W \hat{u}_t - \beta(A\hat{x}_t + B\hat{u}_t)' P (A\hat{x}_t + B\hat{u}_t) - \beta \text{trace}(PCC') - \beta\xi.$$

To simplify this system we eliminate $\hat{u}$ by taking the first-order condition with respect to $\hat{u}$, this yields our policy function

$$\hat{u}_t = -(Q + \beta B' P B)^{-1}(\beta B' P B + W')\hat{x}_t = -F\hat{x}_t$$

Next, using a well-established algorithm we can use the matrices above to calculate the matrix P that summarizes the evolution of the value function. In this stochastic discrete-time setting this algorithm will take the form,

$$P_{j+1} = R + \beta A' P_j A - (\beta A' P_j B + W')(Q + \beta B' P_j B)^{-1}(\beta B' P_j A + W) \tag{24}$$

$$\xi_{j+1} = \beta(1-\beta)^{-1} \text{trace}(P_{j+1} CC') \tag{25}$$

the subscript j represents iterations of the recursive solution method and P_0 is set exogenously.

A.1 Shadow-Price Learning in the Discrete Model

The iterative solution method outlined in the previous section, provides more information about the system than simply the solution. In the recursive algorithm outlined in (24) and (25) an initial guess or perception of the equilibrium in these equations maps to an updated perception of the value function matrix.

We can describe this mapping between perceptions and actuality using an adaptive learning tool called the T-map. The T-map is constructed by examining the link between agents' perceptions and the updated value function that results from these perceptions. The T-map is derived by examining the induced value functions for perception, $V^P(x) = -xT(P)x$. For the discrete non-stochastic case ($C = 0$) the value function induced by a perceived matrix P is

$$V^P(x) = \max_u - (x'Rx + u'Qu + 2x'Wu) - \beta(Ax + Bu)'P(Ax + Bu).$$

Once we characterize agent's control decision we can then describe the T-map, $T(P)$. In the discrete setting the control decision will take the following form,

$$F(P) = (Q + \beta B'PB)^{-1}(\beta B'PA + W')$$

using this we can rewrite the induced value function for perceptions as

$$T(P) = R + \beta A'PA - (\beta A'PB + W)'(Q + \beta B'PB)^{-1}(\beta B'PA + W').$$

This is the mapping between agent's perceptions and actuality in this model. The fixed point of the T-map is the unique steady-state solution for our system (Evans and McGough, 2018).

We, as in the continuous-time case, impose a linear forecasting rule for μ_t since long the optimal path $\mu_t^* = -2P^* x_t$. For additional simplification we assume that the agent forecasts a matrix H instead of $-2P^*$; thus

$$\mu_t = Hx_t + \varepsilon_t^{\mu} \tag{26}$$

This forecasting rule is what our agent believes at time t, the rule acts as a perceived law of motion (PLM). Our agent wants to develop a forecast of future prices using this linear relationship and their beliefs about transition matrix for the state variables A,

$$\mathbb{E}_{t+1}\mu_{t+1} = H\mathbb{E}_{t+1}(x_{t+1}) = H(\tilde{A}x_t + Bu_t)$$

in this forecast $\tilde{A}$ represents the agent's estimation of A. When the agent uses this estimate in their decision making they will estimate the following policy rule and shadow-price parameters

$$u = (2Q - \beta B'HB)^{-1}(\beta B'H\tilde{A} - 2W') = F^{SPD}(H, \tilde{A}, B)x \tag{27}$$

and

$$\begin{aligned}\mu &= \Big(-2R - 2WF^{SPD}(H, \tilde{A}, B) + \beta\tilde{A}'H\big(\tilde{A} + BF^{SPD}(H, \tilde{A}, B)\big)\Big)x \\ &= T^{SPD}(H, \tilde{A}, B)\end{aligned} \tag{28}$$

Equation (28) defines the T-map for our learning rule, this maps the agent's perceived law of motion to the actual law of motion for the system. In our models the agent takes in more information over time using new data. The basic forecasting model the agent implement is,

$$\begin{aligned}x_{t+1} &= A_t x_t + Bu_t + \epsilon_t^x \\ \mu_t &= H_t x_t + \epsilon_t^{\mu}\end{aligned}$$

where ϵ_t^{μ} and ϵ_t^{μ} are error terms. The agent updates their estimates of A_t and H_t using this

new information. Below is a dynamic system describing how the agent estimates A_t and H_t, and how this estimations evolve over time under bounded rationality (Evans and McGough, 2018).

$$
\begin{aligned}
x_t &= Ax_{t-1} + Bu_{t-1} + C\varepsilon_t \\
\mathcal{R}_t &= \mathcal{R}_{t-1} + \gamma_t(x_t x_t' - \mathcal{R}_{t-1}) \\
H_t' &= H_{t-1} + \gamma_t \mathcal{R}_{t-1}^{-1} x_{t-1}(\lambda_{t-1} - H_{t-1}x_{t-1})' \\
A_t' &= A_{t-1} + \gamma_t \mathcal{R}_{t-1}^{-1} x_{t-1}(x_t - Bu_{t-1} - A_{t-1}x_{t-1})' \\
u_t &= -F^{SPD}(H_t, A_t, B)x_t \\
&= (2Q - \beta B'HB)^{-1}(\beta B'HA_t - 2W')x_t \\
\mu_t &= T^{SPD}(H_t, A_t, B)x_t \\
&= \Big(-2R - 2WF^{SPD}(H_t, A_t, B) + \beta A_t' H\big(A_t + BF^{SPD}(H_t, A_t, B)\big)\Big)x_t \\
\gamma_t &= \kappa(t+N)^{-\nu}.
\end{aligned}
\tag{29}
$$

Here $\mathcal{R}_t$ is a measurement for the second moment of the state variable observations x_t and γ_t is a standard gain sequence. For our purposes we will use a constant gain thus $\kappa = 0.01$ and $\nu = 0$.

A.1.1 Learning Results

The algorithm in (29) was applied to a misspecified version of the RBC model outlined in the beginning of this section and a simplified version of the RBC model without government spending or taxation. For both misspecifications, the initial H and A matrices were set as identity matrices, and R was set to fifty times an identity matrix. The initial x and u observations were set near their steady-state values, despite being in deviation from steady-state form. In the discrete-time RBC model we used typical parameter values for the many of the model parameters, the parameter χ was set so that the portion of hours worked in

the non-stochastic steady state was 33% (Hansen, 1985). Below is a table summarizing our parameter values.

Table 6: Discrete-Time Parameter Values

Description		**Value**
A	Total Factor Productivity	1.0
β	Discount factor	0.99
σ	Intertemporal elasticity of subst.	1.0 (log utility)
φ	Frisch elasticity of labor supply	-1.0 (log utility)
χ	Disutility of labor	1.75
α	Capital share	1/3
δ	Depreciation rate	0.025
θ	Drift parameter for tech.	0.895
σ_z	Standard Deviation for tech.	0.01

The agent in this setting understands the basic structure of the transition matrix A and does not use the constant in estimating parameters, instead they estimate coefficients for the processes governing technology and capital using only relevant data. Similar results can be achieved when the agent uses the full set of regressors. Since we use constant gain, the agent's forecast of these parameters oscillates around their rational expectations equilibrium (REE) value, since the agent places equal weight on the information gained from all observations.

The simple model without government spending was run for $50,000$ discrete time periods, at the end of which subtracting the shadow-price parameter matrix from its REE counterpart results in matrix with a norm of 2.42.

B Deriving Continuous-time Recursive Least Squares

First, we define the recursive least squares algorithm. Then, we rewrite our model to depend on time steps Δ to derive a continuous-time version of recursive least squares. Similar approaches for deriving continuous-time recursive algorithms have been outlined in Ljung (1977), Lewis et al. (2007), Sargent (1999). The recursive least squares approach begins with the following difference equation model,

$$y_t = \theta' x_t + e_t$$

where $e_t \sim \mathrm{N}(0, 1)$. Here we can estimate the model by choosing an estimate the minimizes the errors of the model. We select a least-squares method,

$$V_N(\theta) = \frac{1}{N} \sum_{t=1}^{N} \alpha_t [y_t - \theta' x_t]^2 \tag{30}$$

where N is the number observations in the data and α_t is a weighting vector that depends on time. The weighting vector α_t is indirectly related to the gain sequence in adaptive learning literature, for the derivation of RLS it is often set to 1. Implementing this least-squares method we can derive RLS,

$$\hat{\theta}_t = \hat{\theta}_{t-1} + \frac{1}{t} \mathcal{R}_t^{-1} x_t \alpha_t [y_t - \hat{\theta}'_{t-1} x_t],$$
$$\mathcal{R}_t = \mathcal{R}_{t-1} + \frac{1}{t} [\alpha_t x_t x'_t - \mathcal{R}_{t-1}]$$

Note that for illustrative purposes we use a decreasing gain version of our algorithm, the agent puts a decreased amount of weight on observations observed at a later date. Our reason for examining a decreasing gain version of our system is because the derivation of this version of RLS is more intuitively derive from a least squares algorithm. This recursive algorithm estimates coefficients based on observations and an estimate of the second moment

$\mathcal{R}_t$. The avoid the matrix inversion in the system above we can instead use $\mathcal{P}_t = (t \cdot R_t)^{-1}$.

$$\begin{aligned}\mathcal{P}_t &= [\mathcal{P}_{t-1}^{-1} + x_t \alpha_t x_t']^{-1} \\ &= \mathcal{P}_{t-1} - \frac{\mathcal{P}_{t-1} x_t x_t' \mathcal{P}_{t-1}}{1/\alpha_t + x_t' \mathcal{P}_{t-1} x_t}.\end{aligned}$$

Thus our system will become,

$$\hat{\theta}_t = \hat{\theta}_{t-1} + L_t[y_t - \hat{\theta}_t' x_t], \tag{31}$$

$$L_t = \frac{\mathcal{P}_{t-1} x_t}{1/\alpha_t + x_t' \mathcal{P}_{t-1} x_t}, \tag{32}$$

$$\mathcal{P}_t = \mathcal{P}_{t-1} - \frac{\mathcal{P}_{t-1} x_t x_t' \mathcal{P}_{t-1}}{1/\alpha_t + x_t' \mathcal{P}_{t-1} x_t}. \tag{33}$$

We can discretize this model and derive a continuous-time version of recursive least squares. The discretized version of our model with an undetermined time step Δ is,

$$y_t = \theta_t' x_t + e_t$$

Where, the covariance matrix for $e_t \sim N(0, \frac{1}{\alpha_t \Delta})$ as in Lewis et al. (2007). First, we can examine the gain term in (32). Rewriting (32) for this model yields,

$$\begin{aligned}L_t &= \mathcal{P}_{t-\Delta} x_t [(1/\alpha_t \Delta) + x_t \mathcal{P}_{t-\Delta} x_t']^{-1} \\ &= \mathcal{P}_{t-\Delta} x_t \Delta [1/\alpha_t + x_t \mathcal{P}_{t-\Delta} x_t' \Delta]^{-1}.\end{aligned}$$

Dividing through by Δ and then taking the limit as $\Delta \to 0$,

$$K = \lim_{\Delta \to 0} \frac{1}{\Delta} L_t = \alpha_t \mathcal{P}_t x_t$$

Next, if we look at (33) we can rewrite this equation as,

$$\begin{aligned}\mathcal{P}_t - \mathcal{P}_{t-\Delta} &= -\mathcal{P}_{t-\Delta} x_t x_t' \mathcal{P}_{t-\Delta}[(1/\alpha_t\Delta) + x_t \mathcal{P}_{t-\Delta} x_t']^{-1} \\ &= -\mathcal{P}_{t-\Delta} x_t x_t' \mathcal{P}_{t-\Delta}\Delta[1/\alpha_t + x_t \mathcal{P}_{t-\Delta} x_t'\Delta]^{-1}.\end{aligned}$$

Dividing through by Δ and taking the limit as $\Delta \to 0$,

$$\frac{d\mathcal{P}_t}{dt} = -\mathcal{P}_t x_t x_t' \mathcal{P}_t = -K x_t' \mathcal{P}_t.$$

Last, we can derive the continuous-time estimate updating equation (31). Rewriting this equation and diving through by Δ yields,

$$\frac{1}{\Delta}\big(\hat{\theta}_t - \hat{\theta}_{t-\Delta}\big) = \frac{1}{\Delta} L_t[y_t - \hat{\theta}_{t-\Delta}' x_t].$$

Limiting this as $\Delta \to 0$ we get,

$$\frac{d\hat{\theta}_t}{dt} = K[y_t - \hat{\theta}_t' x_t].$$

Assuming $\Delta \approx 0$ our continuous-time RLS algorithm is thus,

$$\begin{aligned}\frac{d\mathcal{P}_t}{dt} &= K[y_t - \hat{\theta}_t' x_t] \\ \frac{d\mathcal{P}_t}{dt} &= -K x_t' \mathcal{P}_t \\ K &= \alpha_t \mathcal{P}_t x_t.\end{aligned} \tag{34}$$

The version of the algorithm depicted in (34) uses time derivatives unlike our maintain specification; this does not change the algorithm's outcomes.

References

Aadland, D. (2001). High frequency real business cycles. *Journal of Monetary Economics*, 48(2):271 – 292.

Achdou, Y., Han, J., Lasry, J., Lions, P., and Moll, B. (2020). Income and Wealth Distribution in Macroeconomics: A Continuous-Time Approach. *Review of Economic Studies*.

Ahn, S., Kaplan, G., Moll, B., Winberry, T., and Wolf, C. (2018). When Inequality Matters for Macro and Macro Matters for Inequality. *NBER Macroeconomics Annual*, 32(1).

Aït-Sahalia, Y. (2010). Estimating Continuous-Time models with discretely sampled data. *Advances in Economics and Econometrics, Theory and Applications, Ninth World Congress*.

Anderson, B. D. and Moore, J. B. (2007). *Optimal Control Linear Quadratic Methods*. Dover.

Benigno, P. and Woodford, M. (2004). Optimal Monetary and Fiscal Policy: A Linear-Quadratic Approach. In *NBER Macroeconomics Annual 2003, Volume 18*, NBER Chapters, pages 271–364. National Bureau of Economic Research, Inc.

Benigno, P. and Woodford, M. (2006). Optimal Taxation in an RBC Model: A Linear-Quadratic Approach. *Journal of Economic Dynamics and Control*, 30(9-10):1445–1489.

Benigno, P. and Woodford, M. (2012). Linear-quadratic Approximation of Optimal Policy Problems. *Journal of Economic Theory*, 147(1):1–42.

Branch, W. and McGough, B. (2011). Business cycle amplification with heterogeneous expectations. *Economic Theory*, 47(2):395–421.

Brunnermeier, M. K. and Sannikov, Y. (2014). A Macroeconomic Model with a Financial Sector. *American Economic Review*, 104(2):379–421.

Dixit, A. (1992). The Art of Smooth Pasting. STICERD - Theoretical Economics Paper Series, Suntory and Toyota International Centres for Economics and Related Disciplines, LSE.

Eusepi, S. and Preston, B. (2011). Expectations, learning, and business cycle fluctuations. *American Economic Review*, 101(6):2844–72.

Evans, G. and McGough, B. (2018). Learning to Optimize. *Draft*.

Forsyth, P. and Labahn, G. (2007). Numerical Methods of Controlled Hamilton-Jacobi-Bellman PDEs in Finance. *Journal of Computational Finance*, 11(2):1–44.

Goodwin, G. and Mayne, D. (1987). A parameter estimation perspective of continuous time model reference adaptive control. *Automatica*, 23(1):57–70.

Hansen, G. and Wright, R. (1992). The labor market in real business cycle theory. *Quarterly Review*, 16(Spr):2–12.

Hansen, G. D. (1985). Indivisible labor and the business cycle. *Journal of Monetary Economics*, 16(3):309–327.

Hansen, L. and Sargent, T. (1991). *Rational Expectations Econometrics*. Underground classics in economics. Westview Press. Includes bibliographical references (pages 283-293).

Hansen, L. and Sargent, T. (2013). *Recursive Models of Dynamic Linear Economies*. Princeton University Press.

Huang, M., Caines, P. E., and Malhame, R. P. (2012). Social optima in mean field lqg control: Centralized and decentralized strategies. *IEEE Transactions on Automatic Control*, 57(7):1736–1751.

Huarng, K. . and Yeh, C. . (1992). Continuous-time recursive least-squares algorithms. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 39(10):741–745.

Kaplan, G., Moll, B., and Violante, G. (2018). Monetary Policy According to HANK. *American Economic Review*, 108(3):697–743.

Kendrick, D. (2005). Stochastic Control for Economic Models: Past, Present and the Paths Ahead. *Journal of Economic Dynamics and Control*, 29(1):3 – 30. Computing in economics and finance.

Lester, C. (2020). Boundedly rational decision making in continuous-time. *Working*.

Lewis, F., Xie, L., and Popa, D. (2007). *Optimal and Robust Estimation: With an Introduction to Stochastic Control Theory*. CRC Press, Second edition.

Lewis, F. L. (1986). *Optimal Control*. Wiley-Interscience.

Ljung, L. (1977). Analysis of recursive stochastic algorithms. *IEEE transactions on automatic control*, 22(4):551–575.

Ljung, L. and Söderström, T. (1983). *Theory and Practice of Recursive Identification*. MIT Press.

Ljungqvist, L. and Sargent, T. J. (2012). *Recursive Macroeconomic Theory*. Mit Press.

Merton, R. (1971). Optimum Consumption and Portfolio Rules in a Continuous-Time Model. *Journal of Economic Theory*, 3(4):373–413.

Mirman, L. (1973). Steady State Behavior of a Class of One Sector Growth Models with Uncertain Technology. *Journal of Economic Theory*, 6(3):219–242.

Mirrlees, J. (1971). *Optimum Accumulation Under Uncertainty: the Case of Stationary Returns to Investment*, pages 36–50. Palgrave Macmillan UK, London.

Mitra, K., Evans, G., and Honkapohja, S. (2013). Policy change and learning in the rbc model. *Journal of Economic Dynamics and Control*, 37(10):1947–1971.

Plosser, C. I. (1989). Understanding real business cycles. *Journal of Economic Perspectives*, 3(3):51–77.

Posch, O., Rubio-Ramírez, J. F., and Fernández-Villaverde, J. (2011). Solving the new Keynesian model in continuous time. Technical report.

Romer, D. (1996). *Advanced Macroeconomics*. Advanced Series in Economics. McGraw-Hill Companies.

Sargent, T. J. (1999). *The Conquest of American Inflation*. Princeton University Press.

Sastry, S. and Bodson, M. (1989). *Adaptive Control: Stability, Convergence, and Robustness*. Prentice-Hall, Inc., USA.

Vrabie, D., Abu-Khalaf, M., Lewis, F. L., and Wang, Y. (2007). Continuous-time adp for linear systems with partially unknown dynamics. In *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 247–253.

Wang, H. and Zhou, X. Y. (2019). Continuous-time mean-variance portfolio selection: A reinforcement learning framework.

Wang, J. and Forsyth, P. (2010). Numerical solution of the hamilton–jacobi–bellman formulation for continuous time mean variance asset allocation. *Journal of Economic Dynamics and Control*, 34(2):207 – 230.

Xie, S., Li, Z., and Wang, S. (2008). Continuous-time portfolio selection with liability: Mean–variance model and stochastic lq approach. *Insurance: Mathematics and Economics*, 42(3):943 – 953.