CAAM 564 Final Project Report

William Cannon Lewis II

April 30, 2019

For my CAAM 564 class project[1], I studied the optimization problem commonly referred to as *system identification* [Ljung, 1999]. Broadly, this problem consists of fitting a linear model of a dynamical system around a set of experienced state-space transitions. Such a linear model is useful in classical control techniques, which seek to control a linear system around a fixed point or given trajectory. For this project, I restricted my attention to deterministic systems, so as to more closely follow the course material. This project is related to my work on reinforcement learning, which is similar to optimal control but assumes that we wish to control an unknown system subject to an unknown cost function. System identification can be seen as a subtask within the larger problem of reinforcement learning, and in this project I also used the fit linear models for control.

More precisely, we can formulate the optimization problem of interest (in the one-dimensional case) as

$$\begin{aligned} \min_{A,B} \quad & \sum_{t=0}^{T-1} \left[(Ax_t + Bu_t) - x_{t+1}\right]^2 \\ \text{s.t.} \quad & x_{t+1} = F(x_t, u_t) \ \forall t \in \{0, \dots, T-1\} \\ & x_t \in \mathbf{X}, u_t \in \mathbf{U} \ \forall t \in \{0, \dots, T-1\} \end{aligned}$$

Here F is the discrete-time operator of a dynamical system of interest, A and B define our linear approximation, and $\{x_t\}, \{u_t\}$ define a trajectory of experienced states and controls, respectively. This definition can also be generalized to the case where the state-space of the dynamical system is not fully observable, in which case we would fit additional C and D matrices to model an observation function dependent on the state and control. We can also generalize to include multiple trajectories of experience, or even a time-varying model. Once we have fit a model in some variant of the the system identification problem, we can proceed to develop controllers based on this model and observe their behavior in the actual dynamical system.

1 1-1D Polynomial Systems

In my project, I developed Python code to approximately solve the above problem for dynamical systems with one state variable and one input, or control variable. I refer to these systems as "1-1D" systems to distinguish them from higher-order dynamical systems, but the more common name in control theory is "Single-Input, Single-Output", or SISO [Hespanha, 2009]. In this project, I focused on fitting linear models to polynomial 1-1D systems of various orders. In order to create these systems, I wrote code that, for a fixed

[1] All of the code produced for this project, as well as more plots, are available at `https://github.com/cannontwo/caam564_final`

order n, would generate update functions of the form

$$F(x_t, u_t) = \sum_{i=1}^{n} \sum_{j=0}^{i} a_{ij} \cdot x_t^j u_t^{i-j}$$

where the coefficients a_{ij} are sampled uniformly randomly from $[-1, 1]$. Note that these systems have no affine term, and so we expect to be able to fit generated systems where $n = 1$ more or less perfectly. In order to prevent state trajectories going to infinity, I placed a limit on the state space of $s \in [-2, 2]$.

2 Generating Data

Since we are interested in fitting a linear model to an unknown system, we need to generate data about the system somehow. In the experiments that I conducted for this model, we are linearizing the system around the state $s = 1$, so that both the autonomous dynamics (i.e., terms in the dynamics which do not depend on u_t) and the control dynamics will be experienced. We can generate data for system identification by simply applying uniformly random controls in $[-1, 1]$, which should supply us with a more or less random walk through the current dynamical system. This strategy is sufficient to identify a truly linear system, but could be insufficient for more complex systems; see Ljung [1999] or Kumar and Varaiya [1986] for a more thorough treatment. In my experiments, I collect 100 discrete-time transitions of the form (s_t, u_t, s_{t+1}) in order to fit the linear model.

3 Fitting the Linear Model

I used Pyomo [Hart et al., 2011] and IPOPT to fit a linear model based on the gathered discrete transition data. In order to do this, I wrote some matrix algebra utilities so that I could use variables defined by Pyomo to represent the A and B matrices of the fit linear system. I then defined the optimization objective for the model fitting problem as minimizing the function

$$f(A, B) = ||Y - AX + BU||_F \tag{1}$$

where Y is a vector of the experienced "s_{t+1}", X is a vector of the experienced "s_t", and U is a vector of the applied controls "u_t". Note that in this case the Frobenius norm reduces to the regular Euclidean norm, but this formulation is useful if the ideas here are applied to higher-dimensional systems. I had intended to extend this formulation to include constraints in the form of slack variables, but my experiments use this unconstrained optimization formulation because I ran out of time in the semester to implement this extension.

Note that this model-fitting algorithm is effectively least-squares estimation of the system matrices, and so a closed-form solution exists to solve this regression. However, I decided to use Pyomo to tackle this problem in order to get some experience working with optimization tools and applying knowledge from class. Additionally, having implemented this optimization problem with Pyomo makes it easy to now add complex constraints, such as positive definiteness of A and B, that would be difficult to incorporate into the closed-form solution.

4 Control

As my research interests are in Reinforcement Learning and Adaptive Control, the final step of my project is to feed the fit linear model into a Linear Quadratic Regulator [Anderson and Moore, 1990] solver, which will generate a feedback controller that seeks to drive the state of the system to 0. In my experiments, I used a Discrete Algebraic Ricatti Equation solver provided by the Python Control package (`https://python-control.readthedocs.io/en/0.8.2/`), which calculates an approximately optimal feedback control gain. I simply use identity matrices to define the quadratic cost function, and so state and action deviance are equally penalized. I then plot a trajectory in the system using the generated controller, which results in the figures below.

As you can see, for low-order systems (e.g., $n = 1$, 2, and 5) the fit linear model is good enough that the resulting feedback controller can more or less stabilize the system at $s = 0$. Note that there is some oscillation in the example trajectories for these low-order systems, but this is largely due to control limits that are not taken into account by the LQR solver. For higher order systems, however, the true dynamics fluctuate quickly enough that our fit linear model is not sufficient to obtain an optimal controller, and so the state trajectories tend toward the state space bounds.

5 Conclusions and Future Work

I have implemented a very simple adaptive control method which allows me to fit a linear model of a 1-1D system and control it via LQR feedback control. This project was largely an exploration of Pyomo for performing experience-based system identification, and I feel that I was successful in demonstrating that this is a feasible way to identify and control low-complexity nonlinear systems. In reality, this project was mostly developed as part of my main PhD research, in which I intend to apply system identification techniques online to control unknown, nonlinear dynamical systems. In that sense, the remaining work to be done on this project will likely take me a few years, but it includes more intelligent data gathering, fitting multiple linear models, more sophisticated control generation, and application to more realistic systems.

References

Lennart Ljung. System identification (2nd ed.): theory for the user. 1999.

João Pedro Hespanha. Linear systems theory. 2009.

Panganamala Ramana Kumar and Pravin Varaiya. Stochastic systems: estimation, identification and adaptive control. 1986.

William E. Hart, Jean-Paul Watson, and David L. Woodruff. Pyomo: modeling and solving mathematical programs in python. *Math. Program. Comput.*, 3:219–260, 2011.

Brian D. O. Anderson and John B. Moore. Optimal control: linear quadratic methods. 1990.

Experiments

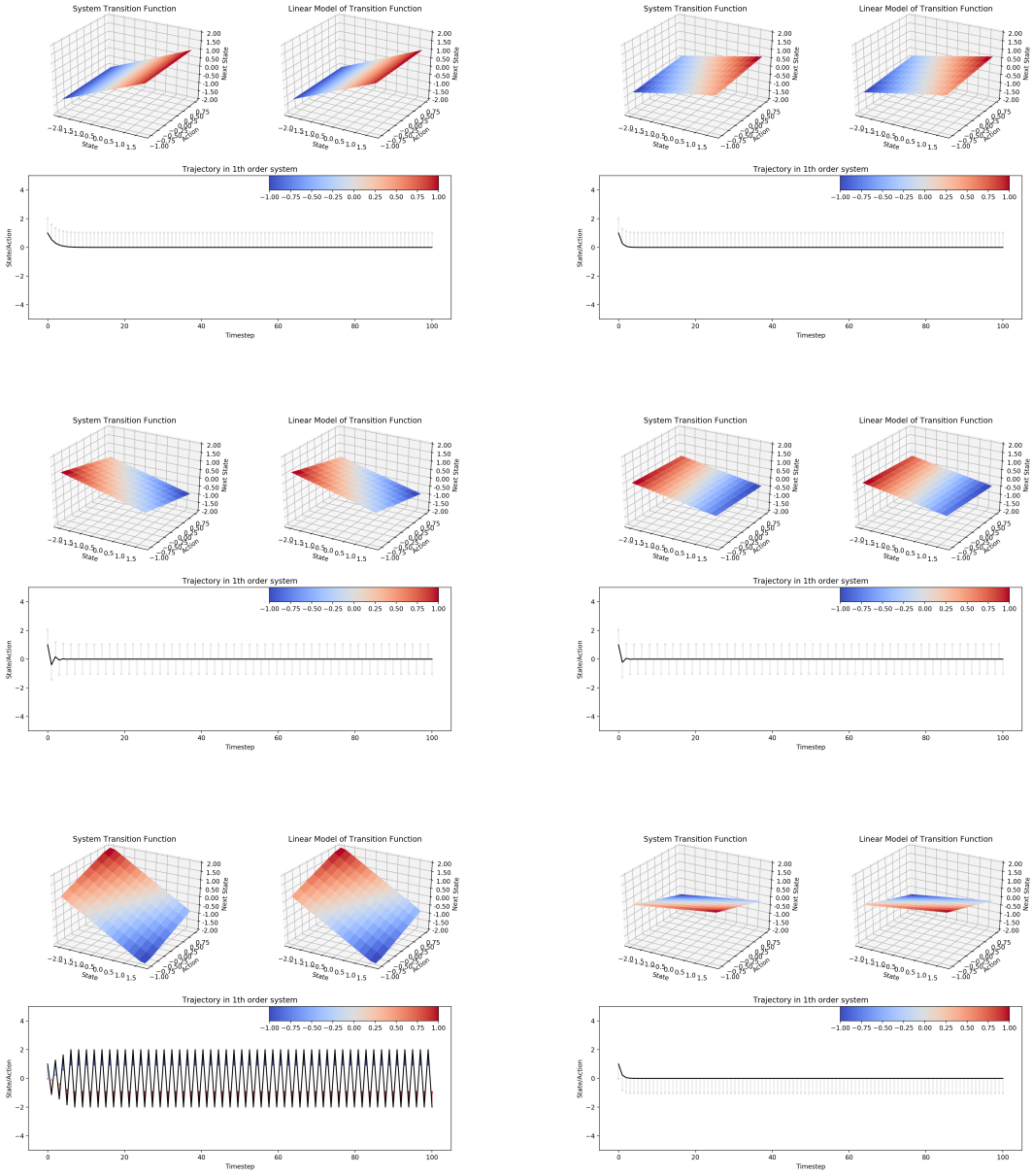

Figure 1: Fitting linear systems. Note basically perfect fit of linear system.

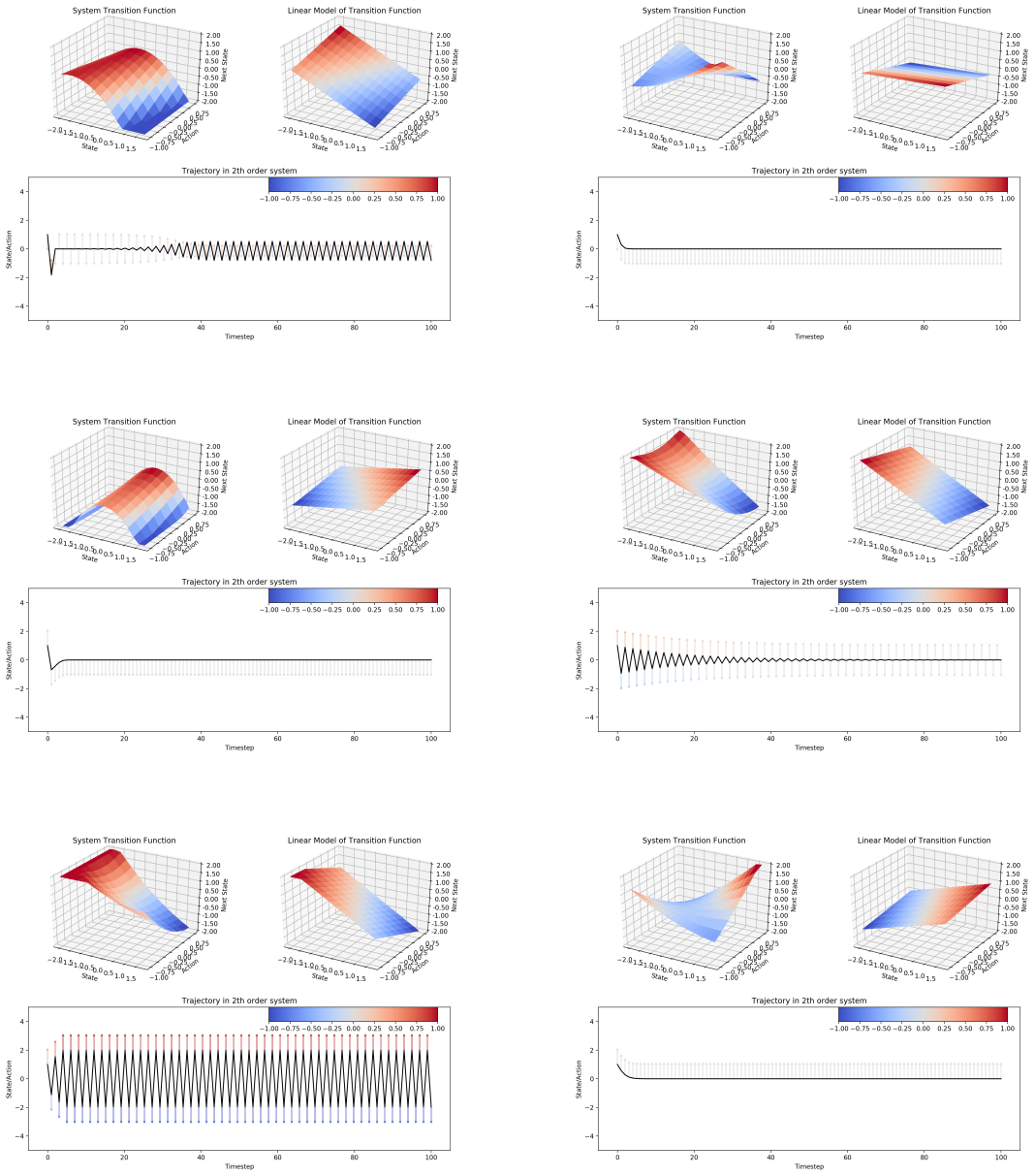

Figure 2: Fitting quadratic systems. Linear model corresponds roughly to slope at $s = 1$.

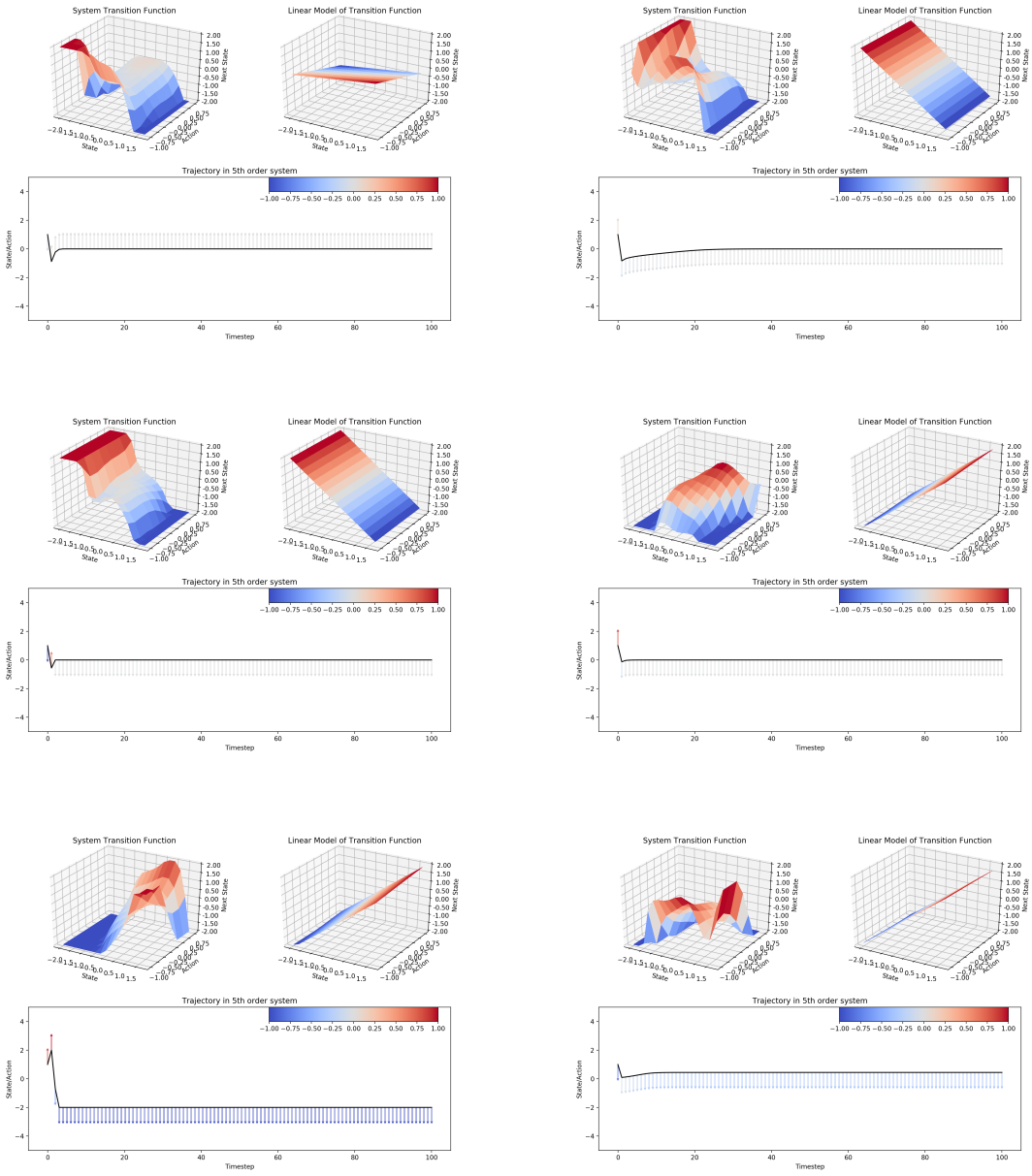

Figure 3: Fitting quintic systems. Linear model corresponds roughly to slope at $s = 1$.

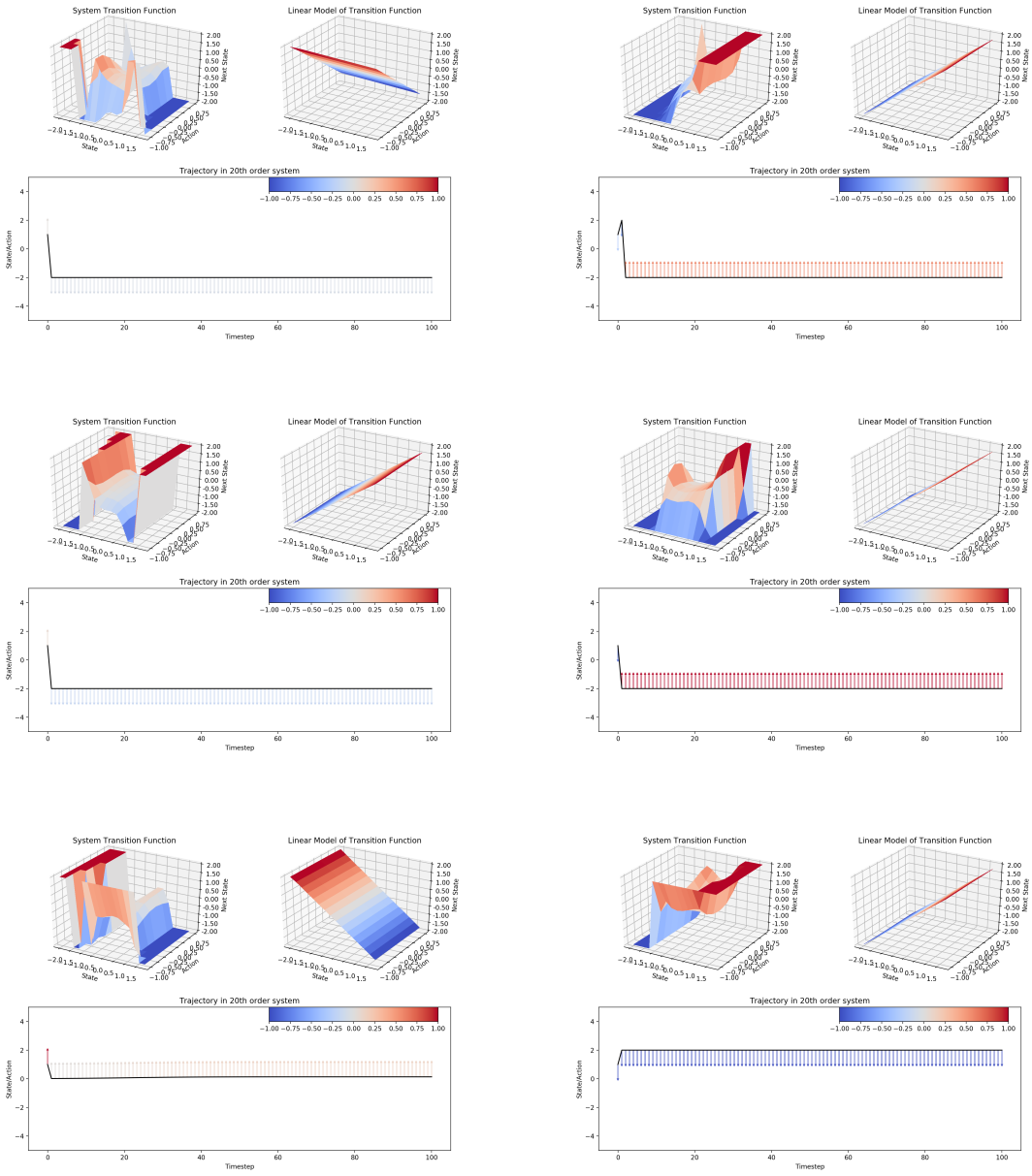

Figure 4: Fitting order 20 systems. Linear model is not sufficient for LQR stabilization.