An Optimal Control Model of Zebra Finch Vocalization

Mike Schachter *
Helen Wills Neuroscience Institute
University of California, Berkeley
Berkeley, CA 94720
mike.schachter@gmail.com

Abstract

In this work, a nonlinear oscillator modeling the syringeal folds of the Zebra Finch is controlled by the state of higher level linear dynamical system. We formulate an optimal control cost function and solution for the learning of bird song.

1 Introduction

First we'll discuss the Zebra Finch vocalization system and it's mathematical formulation, and then the control of that system. All code used to generate figures and results in this paper can be found at:

http://github.com/mschachter/birdy

1.1 The Zebra Finch Vocalization System

Like human speech, Zebra Finch song is generated at it's source by oscillations in airflow generated by vibrating vocal cords. A model for these oscillations is given as:

$$\begin{aligned}\dot{x} &= v \\ \dot{v} &= \gamma^2\alpha + \gamma^2\beta x - \gamma^2 x^3 - \gamma x^2 v + \gamma^2 x^2 - \gamma x v\end{aligned}$$

The control parameters of the model are α and β. γ = 23500 is a constant.

1.2 Control of the Syrinx Model

The goal is to control the parameters α and β in order to produce an observed vocalization. Let $\boldsymbol{\phi}(t) = [\alpha(t)\ \beta(t)]^T$ be the state vector.

Assume that the temporal evolution of $\boldsymbol{\phi}(t)$ is defined by a controlled linear dynamical system:

$$\dot{\boldsymbol{\phi}} = A\boldsymbol{\phi}(t) + \boldsymbol{u}(t)$$

where A is a 2x2 matrix, chosen to make the passive dynamics of the control system decay to some physiologically relevant rest state.

*With a metric ton of help and code from Hédi Soula: hsoula@gmail.com

We discretized time to simplify the analysis. Let $\Delta\tau$ be the time step for simulation of the control system, and define $t_k = k\Delta\tau$. Also define $\boldsymbol{\phi}_k = \boldsymbol{\phi}(t_k)$. Note that the time step for the simulation of the control is significantly larger than that of the oscillator. The discrete time map representing the control system is given by the forward Euler step:

$$\begin{aligned}\boldsymbol{\phi}_{k+1} &= (A\boldsymbol{\phi}_k + \boldsymbol{u}_k)\,\Delta\tau + \boldsymbol{\phi}_k \\ &= (\Delta\tau A + I)\,\boldsymbol{\phi}_k + \Delta\tau \boldsymbol{u}_k\end{aligned}$$

In optimal control theory, a cost function is specified and is to be minimized over time to produce an optimal control law. But what form should the cost function take? We'll make the following assumptions:

1. The control wants to keep the system's instantaneous energy low: $\boldsymbol{\phi}_k^T \boldsymbol{\phi}_k$.
2. The control wants to keep it's own instantaneous energy low: $\boldsymbol{u}^T \boldsymbol{u}$
3. The control wants to produce an instantaneous fundamental frequency that matches that of a stored template.

To elaborate on the last assumption, say we are the given time-varying fundamental frequency of a song syllable that we would like to learn, represented as the function $F(t_k)$. Let $\hat{F} = g(\boldsymbol{\phi}_k)$ be the function that gives the steady state fundamental frequency for a given control $\boldsymbol{\phi}_k$. The function g can be empirically estimated through simulation and approximated through interpolation. To follow this frequency means to keep the quantity $\left(F(t_k) - g(\boldsymbol{\phi}_k)\right)^2$ small.

Given these assumptions, the cost at time t_k of applying control $\boldsymbol{u}$ is:

$$\ell_k\left(\boldsymbol{\phi}_k, \boldsymbol{u}\right) = \boldsymbol{\phi}_k^T \boldsymbol{\phi}_k + \boldsymbol{u}^T \boldsymbol{u} + \left(F(t_{k+1}) - g(\boldsymbol{\phi}_{k+1})\right)^2$$

Let N be the number of time points we want to control, and let $\boldsymbol{\pi}_j = \{\boldsymbol{u}_j(\boldsymbol{\phi}_j), ..., \boldsymbol{u}_N(\boldsymbol{\phi}_N)\}$ be the control law applied from time j to time N. Note that the control law is a sequence of functions! Each control uses feedback information about most recently observed state. The total cost of applying a control law $\boldsymbol{\pi}_0$ given initial state $\boldsymbol{\phi}_0$ is:

$$v\left(\boldsymbol{\phi}_0, \boldsymbol{\pi}_0\right) = \sum_{k=1}^{N} \ell_k\left(\boldsymbol{\phi}_k, \boldsymbol{u}_k\right)$$

1.3 Dynamic Programming Solution to Optimal Control

The optimal cost-to-go is a function that represents the total cost from a time j to time N given that the optimal control function is applied:

$$v_j^*\left(\boldsymbol{\phi}_j\right) = \min_{\boldsymbol{\pi}_j} \sum_{k=j}^{N} \ell_k\left(\boldsymbol{\phi}_k, \boldsymbol{u}_k\right)$$

Dynamic programming is typically used to solve for the optimal cost-to-go. When applying DP, we work backwards in time, creating a recursive algorithm that determines the functional form of the entire optimal control policy $\boldsymbol{\pi}_0^*$. To illustrate this, we are going to take $N = 2$ with a specified initial condition $\boldsymbol{\phi}_0$.

First the solution must be found for $N = 2$:

$$v_2^*\left(\boldsymbol{\phi}_2\right) = \min_{\boldsymbol{u}_2} \ell_2\left(\boldsymbol{\phi}_2, \boldsymbol{u}_2\right)$$

A straighforward but expensive way to solve would be to perform a search over a grid of values for $\boldsymbol{\phi}_2$. For each value on the grid, we then will use gradient descent to minimize $\ell_2\left(\boldsymbol{\phi}_2, \boldsymbol{u}_2\right)$ with respect to $\boldsymbol{u}_2$. The end result is that we have a lookup table of values, giving us a function $\boldsymbol{u}_2(\boldsymbol{\phi}_2)$, and a function $v_2^*\left(\boldsymbol{\phi}_2\right)$.

The optimal cost-to-go at time $N-1=1$ is

$$v_1^*\left(\boldsymbol{\phi}_1\right)=\min_{\boldsymbol{u}_1}\left[\ell_1\left(\boldsymbol{\phi}_1, \boldsymbol{u}_1\right)+v_2^*\left(\boldsymbol{\phi}_2\right)\right]$$

We know the functional form of $\ell_1\left(\boldsymbol{\phi}_1, \boldsymbol{u}_1\right)$ and the numerical form of $v_2^*\left(\boldsymbol{\phi}_2\right)$. We also know that:

$$\boldsymbol{\phi}_2=\left(A \boldsymbol{\phi}_1+\boldsymbol{u}_1\right) \Delta \tau+\boldsymbol{\phi}_1$$

so we can again use gradient descent for a grid on $\boldsymbol{\phi}_1$ to create a lookup table for $\boldsymbol{u}_1(\boldsymbol{\phi}_1)$ and $v_1^*\left(\boldsymbol{\phi}_1\right)$. We now have a functional form for the optimal control policy: $\boldsymbol{\pi}_0^*=\{\boldsymbol{u}_1(\boldsymbol{\phi}_1), \boldsymbol{u}_2(\boldsymbol{\phi}_2)\}$

Now going foward in time, with the knowledge of a start state $\boldsymbol{\phi}_0$, we can compute $\boldsymbol{\phi}_1$, and then the optimal control $\boldsymbol{u}_1\left(\boldsymbol{\phi}_1\right)$. Given $\boldsymbol{u}_1$, we can then compute $\boldsymbol{\phi}_2$ and then $\boldsymbol{u}_2$. The technique is generally applicable for any N.

1.3.1 A More Efficient Implementation

Solving the dynamic programming problem at a time t_k requires finding a function $\boldsymbol{u}_k(\boldsymbol{\phi}_k)$ that minimizes $v_k(\boldsymbol{\phi}_k)$ for any $\boldsymbol{\phi}_k \in \mathcal{P}$, where $\mathcal{P}$ is a pre-specified set of realistic values. This requires us to uniformly sample from $\mathcal{P}$ and solve an optimization problem for each sample, a very computationally expensive prospect!

To reduce the computational burden, we first model the function $\boldsymbol{u}_k(\boldsymbol{\phi}_k)$ as a linear combination of M radial basis functions $\{\psi_1, ..., \psi_M\}$. Each basis function takes the form:

$$\psi_i(\boldsymbol{\phi})=exp\left(\frac{\|\boldsymbol{\phi}-\boldsymbol{\theta}_i\|^2}{\sigma_i^2}\right)$$

where $\boldsymbol{\theta}_i$ is the center of the basis function, σ_i is the bandwidth of the basis function. Each basis function is multiplied by a coefficient $\boldsymbol{c}_i=\left[c_{i1}\ c_{i2}\right]^T$. Let $\boldsymbol{C}=[\boldsymbol{c}_1, ..., \boldsymbol{c}_M]$ be the $2xM$ matrix of coefficients, the control is computed for a given $\boldsymbol{C}$ as:

$$\boldsymbol{u}_k(\boldsymbol{\phi}_k, \boldsymbol{C})=\sum_{i=1}^M \boldsymbol{c}_i \psi_i(\boldsymbol{\phi}_k)$$

We then formulate a single optimization problem to minimize the sum of cost across all points in $\mathcal{P}$ for a given $\boldsymbol{C}$:

$$\boldsymbol{C}^*=\underset{\boldsymbol{C}}{\operatorname{argmin}} \sum_{\boldsymbol{\phi} \in \mathcal{P}} v_k(\boldsymbol{\phi}, \boldsymbol{C})$$

The optimal control at time t_k is then given as $\boldsymbol{u}_k(\boldsymbol{\phi}_k, \boldsymbol{C}^*)$.

To create an initial guess for the optimization problem, each $\boldsymbol{c}_i$ is initialized independently by finding the minimum cost control at $\boldsymbol{\theta}_i$, the center of basis function i:

$$\boldsymbol{c}_i^0=\underset{\boldsymbol{c}_i}{\operatorname{argmin}}\, v_k(\boldsymbol{\theta}_i, \boldsymbol{c}_i)$$

Acknowledgments

Thank you to Hédi Soula, who created a C++/Python implementation of this model in our lab and has provided ideas, expertise, and feedback on this project.

References