Parameter Estimation in Dynamical Systems

Erick Ruiz
eruiz@g.harvard.edu

Due: May 12, 2021

We have surveyed various techniques for solving ordinary differential equations throughout the semester. We studied the Brusselator problem extensively as a model problem because of its fast changing dynamics. However, we did not discuss many systems of ordinary differential equations whose solutions can vary according to a set of parameters. Consider the Lotka-Volterra system, a model for studying the dynamics of competing species.

$$\frac{dx}{dt} = \alpha x - \beta xy \tag{1}$$

$$\frac{dy}{dt} = -\gamma y + \delta xy \tag{2}$$

Although we can gain a lot of intuition about the stability of this system via a linearization analysis[1], we will focus on obtaining a numerical solution by building on the techniques we have discussed in the course.

Table 1: Butcher tableau for the Dormand-Prince 5(4) method

0							
$1/5$	$1/5$						
$3/10$	$3/40$	$9/40$					
$4/5$	$44/45$	$-56/15$	$32/9$				
$8/9$	$19372/6561$	$-25360/2187$	$64448/6561$	$-212/729$			
1	$9017/3168$	$-355/33$	$46732/5247$	$49/176$	$-5103/18656$		
1	$35/384$	0	$500/1113$	$125/192$	$-2187/6784$	$11/84$	
	$35/384$	0	$500/1113$	$125/192$	$-2187/6784$	$11/84$	
	$5179/57600$	0	$7571/16695$	$393/640$	$-92097/339200$	$187/2100$	$1/40$

We will implement the Dormand-Prince 5(4) method to solve the Lotka-Volterra problem. The Butcher tableau is given in Table 1. This method is considered a "first, same as last" (FSAL) method because the last intermediate calculation is reused in the following step as the first intermediate calculation. The numbers 5(4) that are part of the name represent the convergence of the solution and the scaling of the error for this method, respectively. In other words, the Dormand-Prince 5(4) method produces a numerical solution that is fifth order accurate and whose error scales as $\mathcal{O}(h^4)$, where h is the step size used in the integration scheme.

[1]Steven Stogatz has written a great book that discusses how to analyze dynamical systems from a graphical perspective.

The `code/` directory contains all of the program files used. The solver is defined and implemented in `dopri54.hh` and `dopri54.cc`, and the Lotka-Volterra problem is defined in `odes.hh`. To test the solver, we use a fixed time step integration scheme to solve Eq. (1) and (2) with the parameters set to $\alpha = 1.5$, $\beta = \delta = 1$, and $\gamma = 3$ and initial conditions $\mathbf{u}_0 = [1, 1]^\top$. The results are shown in Figure 1. Qualitatively, we can see that the time evolution of each species' population depends on the other, which is to be expected since we assume that both species are competing for resources.

This is great, but we can take this analysis one step further by looking at how the parameters affect the solution. What we propose is a *sensitivity analysis* on the Lotka-Volterra equations. If we write Eq. (1) and (2) in vector notation as

$$\mathbf{u}' = \mathbf{f}\left(\mathbf{u}, \mathbf{p}, t\right), \tag{3}$$

where $\mathbf{u} = [x, y]^\top$ and $\mathbf{p} = [\alpha, \beta, \gamma, \delta]$ is a vector containig all of the parameters, then the forward sensitivity equations can be derived directly from the chain rule.

$$\frac{d}{dt}\frac{d\mathbf{u}}{d\mathbf{p}} = \frac{d\mathbf{f}}{d\mathbf{u}}\frac{d\mathbf{u}}{d\mathbf{p}} + \frac{d\mathbf{f}}{d\mathbf{p}} \tag{4}$$

Here, the term $d\mathbf{f}/d\mathbf{u}$ is the Jacobian matrix, $\mathbf{J}$, whose entries are defined as

$$\mathbf{J}_{ij} = \left[\frac{d\mathbf{f}}{d\mathbf{u}}\right] = \frac{\partial f_i}{\partial u_j}. \tag{5}$$

The size of the Jacobian matrix is defined by the number of degrees of freedom of the ODE system. For m degrees of freedom, the Jacobian matrix will be of size $m \times m$. Similarly, the term $d\mathbf{f}/d\mathbf{p}$ is a matrix of partial derivatives with respect to the parameters, whose entries are defined as

$$\left[\frac{d\mathbf{f}}{d\mathbf{p}}\right]_{ij} = \frac{\partial f_i}{\partial p_j}. \tag{6}$$

The size of $d\mathbf{f}/d\mathbf{p}$ is defined by the number of degrees of freedom of the ODE system and the number of parameters. If $\mathbf{p} \in \mathbb{R}^\ell$, then $d\mathbf{f}/d\mathbf{p} \in \mathbb{R}^{m\times\ell}$. From this, it follows that the term $d\mathbf{u}/d\mathbf{p} \in \mathbb{R}^{m\times\ell}$ to satisfy the rules of matrix multplication. If we define $\mathbf{S} = d\mathbf{u}/d\mathbf{p}$, then the forward sensitivity equations can be written more compactly as

$$\frac{d}{dt}\mathbf{S} = \mathbf{J}\cdot\mathbf{S} + \frac{d\mathbf{f}}{d\mathbf{p}}. \tag{7}$$

For the Lotka-Volterra equations, the Jacobian matrix

$$\mathbf{J} = \begin{bmatrix} \alpha - \beta y & -\beta x \\ \delta y & -\gamma + \delta x \end{bmatrix} \tag{8}$$

and the $m \times \ell$ matrix of derivatives

$$\frac{d\mathbf{f}}{d\mathbf{p}} = \begin{bmatrix} x & -xy & 0 & 0 \\ 0 & 0 & -y & xy \end{bmatrix}. \tag{9}$$

If we denote the entries of $\mathbf{S}$ as s_{ij}, then the forward sensitivity equations can be expanded as follows.

$$\frac{d}{dt}\begin{bmatrix} s_{11} & s_{12} & s_{13} & s_{14} \\ s_{21} & s_{22} & s_{23} & s_{24} \end{bmatrix} = \begin{bmatrix} \alpha - \beta y & -\beta x \\ \delta y & -\gamma + \delta x \end{bmatrix}\begin{bmatrix} s_{11} & s_{12} & s_{13} & s_{14} \\ s_{21} & s_{22} & s_{23} & s_{24} \end{bmatrix} + \begin{bmatrix} x & -xy & 0 & 0 \\ 0 & 0 & -y & xy \end{bmatrix} \tag{10}$$

The solutions s_{ij} are the derivatives of the i-th state variable with respect to the j-th parameter. The initial conditions for the sensitivity equations, $\mathbf{S}_0$, are defined by $\mathbf{u}_0$, the initial conditions of the original problem. For our Lotka-Volterra problem with $\mathbf{u}_0 = [1, 1]^\top$,

$$\mathbf{S}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{11}$$

The file `odes.hh` contains two small classes, `lv_sensitivity` and `lvs_dop54`, which define the Lotka-Volterra problem with the sensitivity equations built-in. The class `lvs_dop54` is derived from `lv_sensitivty` and `dopri54`. An instance of this class can be used to solve the Lotka-Volterra problem with along with the sensitivity equations. The file `lvs_test.cc` is the front-end program that accomplishes this using the same parameter values and initial conditions as before. We plot the solutions $s_{x\alpha}$ and $s_{y\alpha}$ in Figure 2. Qualitatively, we can see that the solutions are sensitive to changes in the parameter α.

Being able to solve ODEs numerically is a valuable tool. However, in practice we may not know everything about the system we are dealing with. For instance, suppose we are studying a system that can be modeled as a Lotka-Volterra problem, but we do not know the exact parameters of the system. If we can obtain data from the system of interest, we may have a chance at *estimating* the parameters of the system with the help of the sensitivity equations. To do this, let us define a set of perturbed parameters, $\hat{\mathbf{p}} = [\hat{\alpha}, \hat{\beta}, \hat{\gamma}, \hat{\delta}]$. We can generate a numerical solution to the Lotka-Volterra problem with parameters $\hat{\mathbf{p}}$ and compare the result with the reference data. To quantify how accurate our solution is compared to the reference data, let us define the *loss*, or sometimes called the error, between the two solutions.

$$L = \frac{1}{n} \sum_{i=1}^{n} \left(\hat{\mathbf{u}}^{(i)} - \mathbf{u}^{(i)} \right)^2 \tag{12}$$

Here, $\hat{\mathbf{u}}^{(i)}$ and $\mathbf{u}^{(i)}$ are the solutions to the Lotka-Volterra problem at the i-th time step. Our goal is to obtain a set of parameters that are close enough to the real parameters. This can be framed as an optimization problem. We wish to minimize the loss, or the error, between the reference data and our numerical solution. Our strategy will be to generate a numerical solution using the pertubed parameters, $\hat{\mathbf{p}}$, and update the parameters using gradient descent as follows.

$$\hat{\mathbf{p}} := \hat{\mathbf{p}} - \eta \frac{\partial L}{\partial \hat{\mathbf{p}}} \tag{13}$$

The notation ":=" does not represent equality. Instead, it can be read as "assign $\hat{\mathbf{p}}$ to be the old values of $\hat{\mathbf{p}}$ minus some change that is proportional to the derivative of the loss with respect to the parameters." The term $dL/d\hat{\mathbf{p}}$ can be obtained from the chain rule.

$$\frac{dL}{d\hat{\mathbf{p}}} = \frac{dL}{d\hat{\mathbf{u}}} \frac{d\hat{\mathbf{u}}}{d\hat{\mathbf{p}}} \tag{14}$$

Notice that the term $d\hat{\mathbf{u}}/d\hat{\mathbf{p}}$ is exactly the solution to the sensitivity equations that we derived earlier. Hence, we can use the numerical solution to the sensitivity equations to update the parameters. The front-end program `param_est.cc` estimates the parameters of the Lotka-Volterra system from Figure 1 starting with the perturbed parameters $\hat{\mathbf{p}} = [1.2, 0.8, 2.8, 0.8]$. The class `lvs_dop54`, which is defined in `odes.hh`, contains a special `solve` method that optimizes the parameters using the gradient descent algorithm. Figure 3 plots the loss with respect to the *epoch* number, and Table 2 lists the estimated parameter values.[2] Notice that the results agree with our original parameters!

[2] In the machine learning literature, an *epoch* is one complete cycle through a set of training data. Even though this parameter estimation problem is not framed as a machine learning problem, we will borrow some of the terminology.

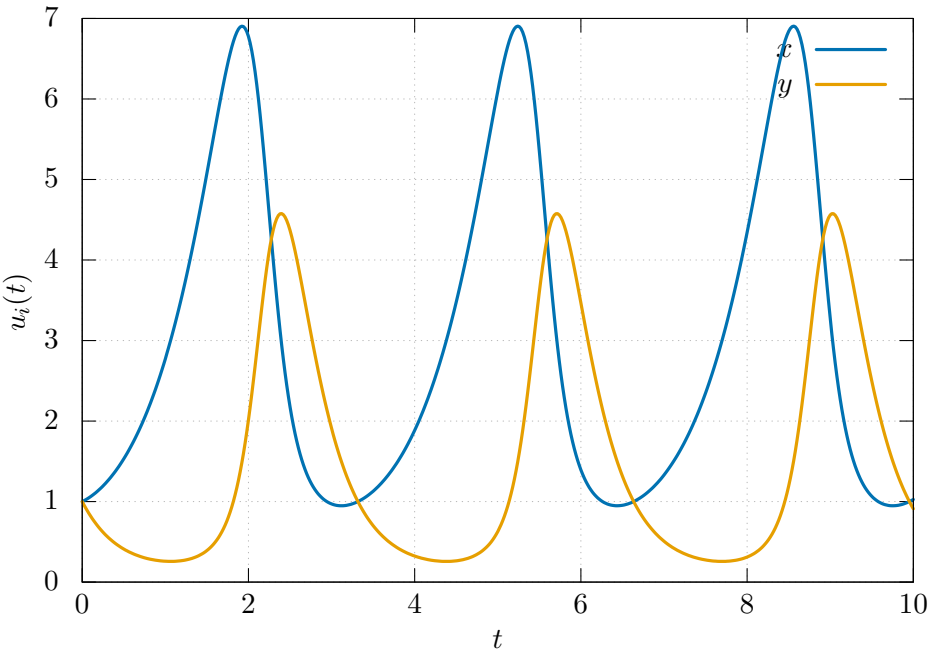

Figure 1: Solution to the Lotka-Volterra equations with $\alpha = 1.5$, $\beta = \delta = 1$, $\gamma = 3$ and initial conditions $x_0 = y_0 = 1$. The Dormand-Prince method produces a smooth solution with $n = 500$ time steps.

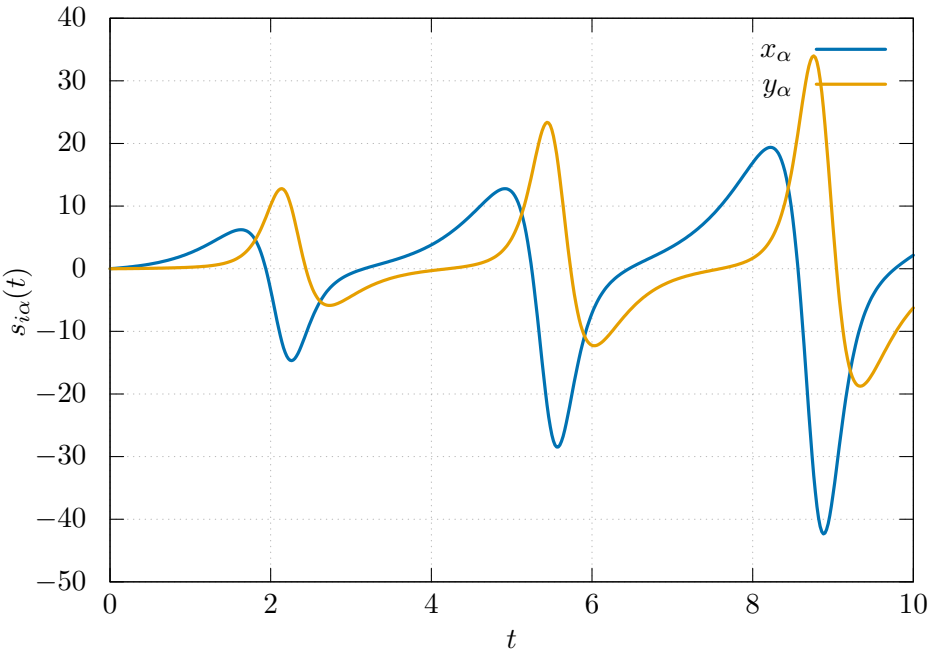

Figure 2: Changes in the solutions, x and y, on the parameter α as a function of time

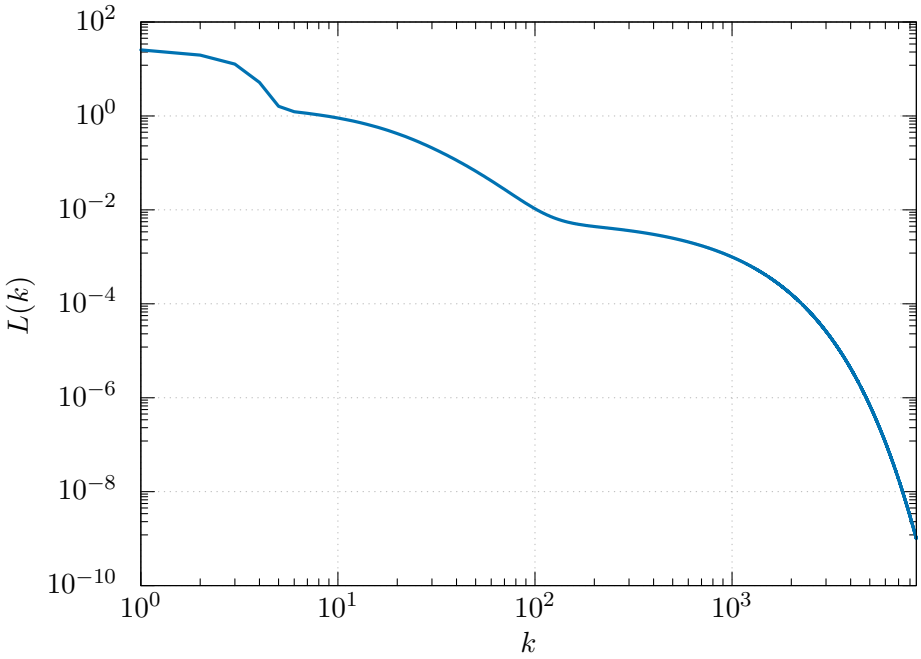

Figure 3: A plot of the loss function as a function of epoch number, k

Table 2: Estimated parameter values using the forward sensitivity equations

$$\hat{\alpha} \approx 1.50001 \quad \hat{\beta} \approx 1.00001 \quad \hat{\gamma} \approx 2.99996 \quad \hat{\delta} \approx 0.99999$$