2.086 Final Project: Parameter Fitting and Sensitivity Analysis for Differential Systems of Motion

Landon Carter

December 10, 2014

Abstract

Many problems in science and technology can be modelled more easily via differential equations than directly modelled, and many of these differential equations are insoluble symbolically. This leaves numerical simulation as the only route towards modelling the problem. Here we utilize `MATLAB` to set up a generally applicable method for modelling these problems and fitting starting parameters to data. Using a falling ball and a swinging pendulum as two example problems, we analyze the sensitivity of the example models to these starting parameters as well as providing an error analysis of the results.

1 Problem Statement

1.1 Motivation and Objective

Many problems in science and technology are modelled more easily via differential equations than by direct models, so being able to solve differential equations is an important tool in any scientist's toolkit. Unfortunately, many scientifically interesting differential systems are insoluble and require numeric methods to approximate solutions. Many of these differential equations include fitting parameters - constants that must be fit to individual datasets.

In this work, we model two example systems, implementing a fast and robust method to numerically solve the differential equations given fitting parameters, implementing a method to fit the fitting parameters given starting values, and finally a method to test many starting values to ensure a global minimum fit is reached.

1.2 Mathematical Formulation

In general, we are given an ordinary differential equation of X as function of t. This differential equation may be of any order, linear or nonlinear. In this work, we

consider two second-order nonlinear equations - one modelling the motion of a falling ball with air resistance, and one modelling the motion of a swinging pendulum with air resistance.

Next, we define β - the set of fitting parameters required to numerically solve the differential equations.

In order to fit β, we adopt an objective function which should be minimized over β.

$$J(\beta) \equiv \sum_{i=1}^{m} (X_i^{\text{meas}} - X^{\text{model}}(t_i; \beta))^2$$

Clearly $J(\beta)$ is minimized when $X_i^{\text{meas}} = X^{\text{model}}(t_i; \beta)$. The differential equations we will be examining are described below.

1.2.1 Falling Ball

In this case, we will be examining the motion of a falling ball, modelled by

$$\begin{cases} \dfrac{d^2Z}{dt^2} + \alpha \dfrac{dZ}{dt}\left|\dfrac{dZ}{dt}\right| = -g, & 0 < t \leq 0.6\text{s} \\ Z(0) = 1.997\text{m}, & \dfrac{dZ}{dt}(0) = \dot{Z}_0 \end{cases},$$

Where Z is the height of the ball in meters, g is the magnitude of the acceleration of gravity, 9.81m/s^2, and α is given by

$$\alpha \equiv \frac{\frac{1}{2} C_D \rho_{\text{air}} A_{\text{frontal}}}{m_{\text{ball}}}.$$

Here C_D is the drag coefficient, ρ_{air} is the density of air, A_{frontal} is the projected area of the ball in the direction of motion, and $m_{\text{ball}} = 0.014\text{kg}$ is the mass of the ball. Note our ball is roughly spherical, hence $A_{\text{frontal}} = \pi r_{\text{ball}}^2$, for $r_{\text{ball}} = 0.0508\text{m}$.

Based on this formulation, $\beta = (\dot{Z}_0 \ \alpha)^{\text{T}}$.

1.2.2 Swinging Pendulum

In this case, we will be examining the motion of a swinging pendulum, modelled by

$$\begin{cases} \dfrac{d^2\theta}{dt^2} + \dfrac{g}{L_{\text{eff}}} \sin(\theta) + \alpha \dfrac{d\theta}{dt}\left|\dfrac{d\theta}{dt}\right| = 0, & 0 < t \leq t_{\text{final}} \\ \theta(0) = \theta_0, & \dfrac{d\theta}{dt}(0) = \dot{theta}_0 \end{cases}.$$

Here $\theta(t)$ is the angular position of the pendulum bob in radians, g is the magnitude of acceleration of gravity, 9.81m/s^2, L_{eff} is the effective length associated with the (in

actuality) compound pendulum of interest, α is a parameter related to the aerodynamic drag on the pendulum bob, and θ_0 and $\dot{\theta_0}$ are the initial angular displacement and angular velocity of the pendulum, respectively.

Based on this formulation, $\beta = (\theta_0\ \dot{\theta_0}\ \alpha\ L_{\text{eff}})^{\text{T}}$. We are further given that $0 \leq \alpha \leq 0.02$, and that the theoretical value of $L_{\text{eff}} = 0.7857$m. We may also estimate θ_0 and $\dot{\theta_0}$ based on the experimental data provided.

1.2.3 Goals

Mathematically, we would like to develop methodology to rapidly set up solutions to any ODE with fitting parameters, as well as methodology to quickly set up method to find fitting parameters even given a range of potential starting values. We also provide commentary on the efficiency of parallelization on the fitting of parameters given a range of starting values.

In order to accomplish this, we apply this to our two model systems, as well as summarizing the general procedure in the conclusions. In conjunction with this, we will analyze the sensitivity of $J(\beta)$ to small fluctuations in β for the swinging pendulum.

2 Computational Methodology

Our basic computational methodology is described in the flowchart below: [h] First,

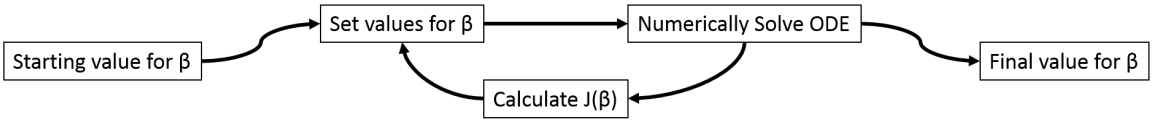

Figure 1: Flowchart of basic computational methodology

we choose initial values for β. In the falling ball portion, we choose these as reasonable values, and manually check a few other values to ensure they all converge to the same optimal values. In the swinging pendulum portion, we fix two of the values, L_{eff} and θ_0, based on the provided theoretical value for L_{eff} and the provided first data point for $\theta(t)$. We vary the other two parameters from $-.005 < \alpha_{\text{est}} < .025$ and $2 < \dot{\theta_0} < 5$, based on an initial estimate from a finite-difference approximation on the first two data points of $\dot{\theta_0} = 3.78$. The modified procedure when considering a range of starting values for β is described later.

After setting initial values for β, we run a function minimizer, `fminsearch`, to minimize $J(\beta)$. The function minimizer numerically solved the differential equation using `MATLAB`'s built in `ODE45`. One caveat in this process is that `ODE45` can only solve

first-order systems of ODE's. Thus, to adapt it to our higher-order ODE, we redefined it as a system of multiple first-order ODE's, where different variables represented $\frac{dX}{dt}$ and X.

This process of `fminsearch` optimizing $J(\beta)$ based on `ODE45` is summarized by the loop. Once `fminsearch` has converged, it produces a final, optimized value for β, as well as a value for $J(\beta_{\text{opt}})$.

The modified procedure, where we consider a range of starting values for β, is summarized in the flowchart below: The primary modifications are setting the range

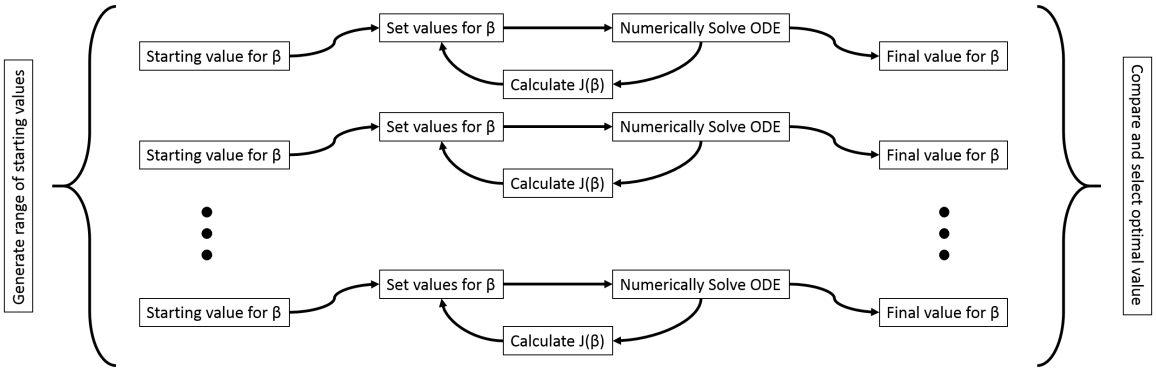

Figure 2: Flowchart of modified computational methodology

of initial β values, then running the same procedure given each starting point. After each point has converged, the values for $J(\beta_{\text{opt}})$ are compared and the overall best value for β is selected.

Though not shown in the flowchart, it was additionally required to handle exceptions where a particular `fminsearch` would fail to converge. In this case, β_{opt} was set to $(1\ 1\ 1\ 1)^{\text{T}}$ and $J(\beta)$ was set to 5 (properly converged values of $J(\beta)$ were in the 10^{-2} range). Because the results were then filtered to select the best convergence, all unconverged values were excluded from the final result.

In this modified procedure, it was additionally possible to parallelize the processing of different initial β values, using `MATLAB`'s built-in `parfor` loop instead of the single-threaded `for` built-in. This allowed for a dramatic increase in speed on my quad-core computer. Those with access to more computational resources, especially a `MATLAB` compute pool, can experience nearly linear scale-ups in speed though parallelization.

3 Results

In the general case, we have now described a methodology and developed a framework for rapidly setting up and solving ODE's with fitting parameters, even given a range of starting values for the fitting parameters.

Additionally, we have applied these methodologies to our two model systems, achieving a well-converged value for β in each case, as well as achieving confidence that our value is well-converged and accurate.

First, we present results for the falling ball, illustrating application of the method described above to a simple system that does not require convergence testing. Next, we present results for the swinging pendulum, illustrating application of the method described above to a system that requires multiple starting values for β optimization in order to find a global minimum for $J(\beta)$.

After going through the simple results of our models, we move to the sensitivity analysis, presenting a map of convergence for the swinging pendulum as well as a map of sensitivity of $J(\beta)$ to fluctuations in β.

Finally, we provide commentary on the effectivness of parallelization of the multiple β starting values variation.

3.1 Model Results

3.1.1 Falling Ball

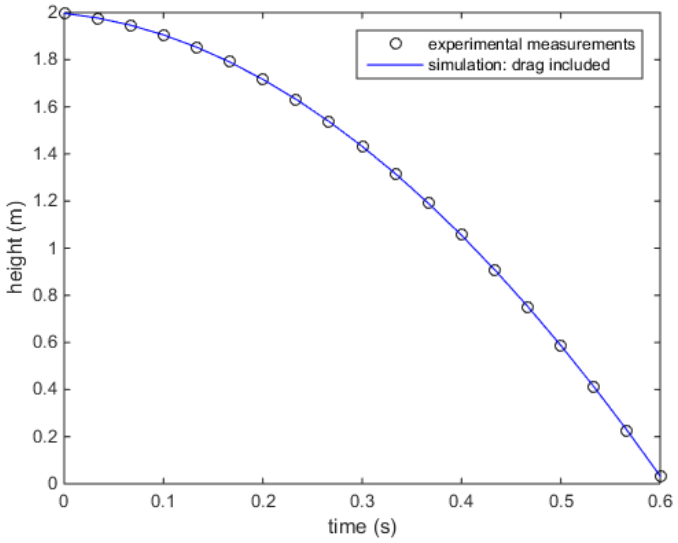

Figure 3: Falling Ball: simulation vs experiment

As can be seen in figure 3, the simulation results for the falling ball closely match the experimental results.

The final values for β, given an initial value for `fminsearch` of $\beta_{\text{initial}} = (-1\ .02)^{\text{T}}$, were as follows:

$$\beta = (\dot{Z}_0\ \alpha)^{\text{T}} = (-0.4283\ 0.0444)^{\text{T}}$$

This led to a value of $J(\beta) = 2.9310\text{e} - 5$. This very small value of $J(\beta)$ gives us confidence in our fitting parameters. Furthermore, repeating the minimization with $\beta_{\text{initial}} = (-.5\ .05)^{\text{T}}$, and $\beta_{\text{initial}} = (-1.5\ 0)^{\text{T}}$ also yielded the same converged result to within a predefined tolerance of 10^{-4}.

3.1.2 Swinging Pendulum

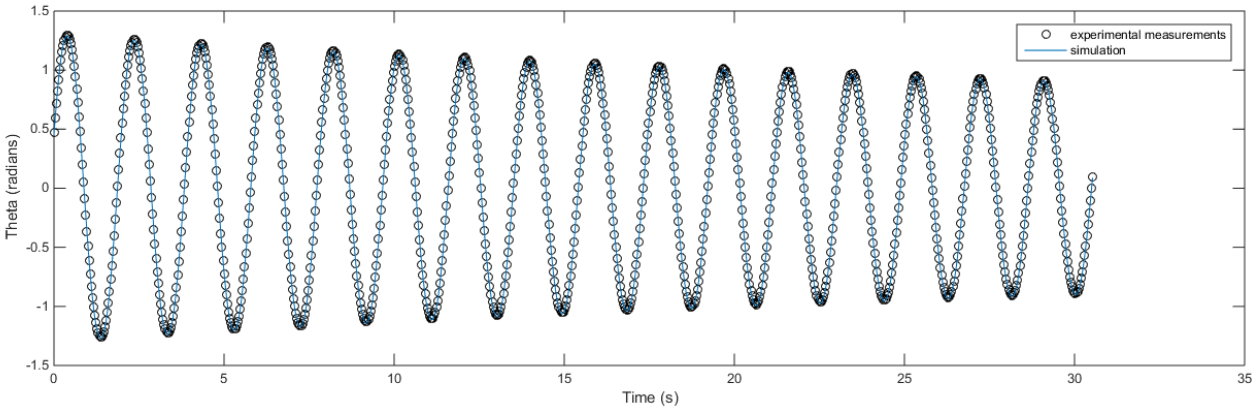

Figure 4: Swinging Pendulum: simulation vs experiment

As can be seen in figure 4, the simulation results for the falling ball closely match the experimental results.

The final values for β, given an initial value range for `fminsearch` of $\beta_{\text{initial}} = (0.4689\ (2, 5)\ (-.005, 0.025)\ .7857)^{\text{T}}$, were as follows:

$$\beta = (\theta_0\ \dot{\theta}_0\ \alpha\ L_{\text{eff}})^{\text{T}} = (0.4551\ 3.9596\ 0.0071\ 0.7832)^{\text{T}}$$

This led to a value of $J(\beta) = 0.0703$. This very small value of $J(\beta)$, along with our many-starting-values procedure, gives us confidence in our final fitting parameters.

Using these final fitting parameters, we predict a value of $\theta(61.13) = -0.4129$.

3.2 Sensitivity Analysis

3.2.1 Convergence

In order to analyze the sensitivity, we must first confirm that not all reasonable starting values for β converge to the same final value. Therefore, using our many-starting-values procedure to generate β values for a range of β_{initial} as well as a value for $J(\beta)$ in each case, we generate a contour plot of the final values of $J(\beta)$ as a function of the β_{initial} values.

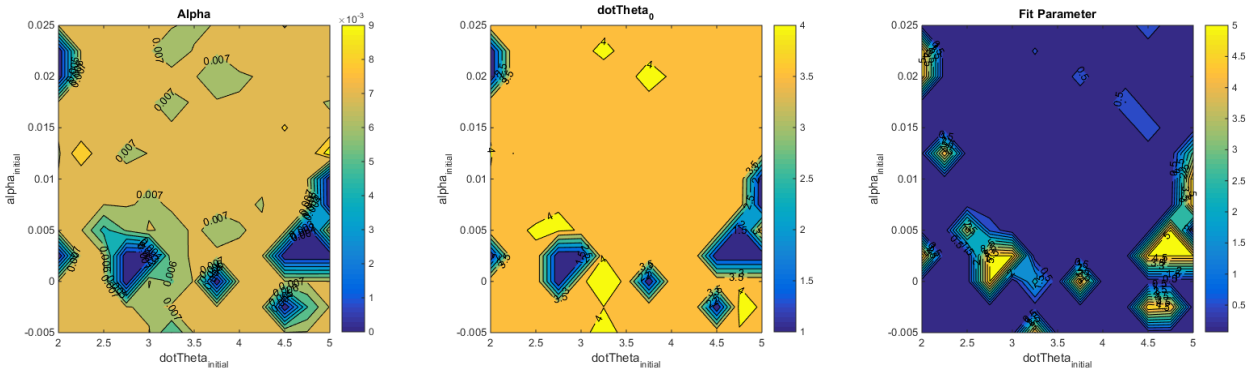

Figure 5: Map of the converged values of $J(\beta)$ as a function of β_{initial}, varying $\dot{\theta}_0$ and α. For β_{initial} that did not converge, $J(\beta) = 5$ was set.

As can be seen by figure 5, not all values of β_{initial} converged to the same place, and many did not converge at all. This reveals both the effectiveness of our many-starting-values procedure as well as the necessity of this procedure. Though one may try a "hit and miss" strategy to find a working β_{initial}, only by testing a very large set of β_{initial} can we be sure we have reached the global minimum.

3.2.2 Sensitivity

In order to show the necessity of actually applying `fminsearch` as well as to examine the sensitivity of $J(\beta)$ with respect to fluctuations in β, we present the following two plots: Clearly, the values of $J(\beta)$ observed here are significantly higher than those found in the optimized functions. Clearly also, they vary significantly across different values of β, so we know that $J(\beta)$ is a sensitive measure to fluctuations in β.

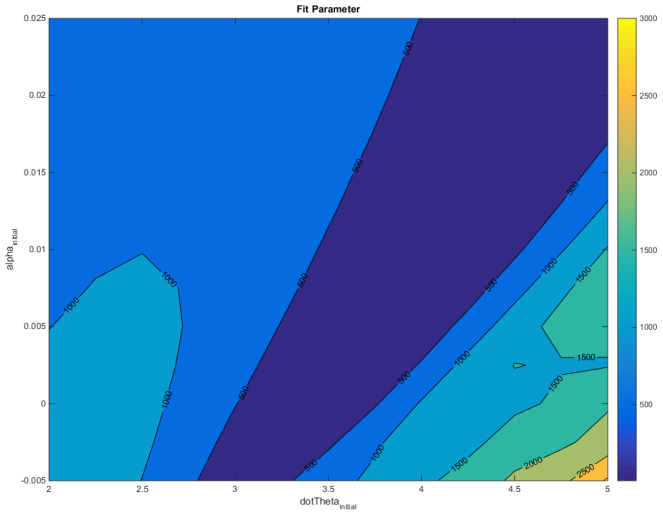

Figure 6: Plot of $J(\beta)$ of various values of β without applying `fminsearch`.

Zooming in on the solution (figure 7, on the following page), we again calculate $J(\beta)$ without applying `fminsearch`, which allows us to see the local variations and be sure that we have converged well. This also reveals to us that this particular model is more sensitive to fluctuations in $\dot{\theta}_0$ per order of magnitude than α - this makes sense, given that changing α will only slightly affect the later magnitudes, while changing $\dot{\theta}_0$ will essentially offset all results, changing the overall outcome significantly.

3.2.3 Parallelization

Finally, a quick aside on parallelization - clearly, each of the `fminsearch` instances do not rely on each other and can run simultaneously. Therefore, applying `parfor` to create multiple instances of `fminsearch` allows us to efficiently parallelize the multiple-starting-values procedure.

Without parallelization, it took 215.8 seconds to converge $J(\beta)$ for 14 starting values of β. With parallelization, this was reduced to a mere 56.7 seconds, running on a quad core machine. This equates to a 380% increase in speed, at a parallelization efficiency of 95%.

Given a more complex problem with more β parameters which are unsure, this parallelization, especially given more robust computing resources (a `MATLAB` worker pool), would allow previously untractable problems to be solved in a reasonable time-frame.

3.3 Error Analysis

Here, we'll take a look at the possible sources of error in our fitting.

Noise: Of course, with any measurements, there is noise associated. However,

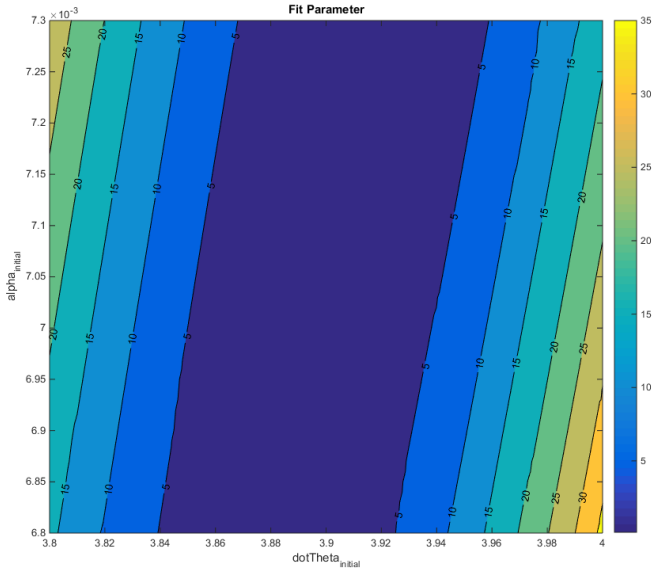

Figure 7: Zoomed in plot of $J(\beta)$ of various values of β without applying `fminsearch`, showing the detail around the global minimum.

because we fit to the entire dataset rather than a small portion, the effects of noise will be significantly reduced. This is especially true for the swinging pendulum, where it is key for us to get the period correct - our measure of $J(\beta)$ is so sensitive to getting a correct period that any noise will be almost completely eliminated.

Model Complexity: Our models could be over- or under-fitting the data, which would lead to errors in our fits. Though this is possible, given the simplicity of the systems studied, it seems unlikely that we are significantly underfitting the data. Furthermore, because our models were based on fundamental physics principles, we are certainly not overfitting the data.

Numerical Precision: As with any computer calculation, we will inherently be limited by the precision of our numbers. In terms of precision of the numbers, this is significantly minimized by setting proper tolerances in `ODE45` and `fminsearch`. In terms of granularity of data, we have no real control over the granularity of data provided. Because we were able to fit the provided data accurately and with very little residual, it can safely be concluded that the data was sufficiently non-granular to provide an accurate fit.

4 Conclusions

Overall, we were very effective in implementing a general procedure for ODE solution given fitting parameters and a dataset to fit to. Two example systems were fit successfully.

Furthermore, the sensitivity of $J(\beta)$ was successfully analyzed in the swinging pendulum model, as well as the convergence criteria for inputs to $J(\beta)$.

4.1 Extensions

Though the provided examples were extremely illustrative in showing the potential of our methods, to truly illustrate the necessity of scanning many β_{initial}. One such system which would be interesting to study would the a double pendulum.

One piece which we did not cover in this work is mathematical error analysis and confidence intervals. It would be useful when applying the presented methods to additionally provide confidence intervals to the simulated data, allowing the methods to be more useful to general scientific practice.

5 Appendix 1: Matlab Codes

Please see included `.zip` file.