
ES_APPM 346 Project 5

NORTHWESTERN UNIVERSITY

NICOLAS GUERRA
MARCH 17, 2021

1 Written Assignment

(a) Brief Explanation

For this boundary value problem, I will use the Shooting Method. The Shooting Method takes a beginning guess, s , for the initial conditions for the variables where the ICs were not originally given. In this case, initial conditions for λ_1 , λ_2 , λ_3 , and λ_4 are guessed to be zero. A variable Z is then created to take into account the derivative of F with respect to s . There will be four Z prime ODEs because there are four initial conditions that were not given. The Z ODEs are the Jacobian multiplied by each individual Z resulting in 32 differential equations. Below is the Jacobian of the system needed for the Z ODEs.

Top Left of Jacobian:

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{2m x_3 x_4 + \frac{2\lambda_3 m x_2}{(m x_2^2 + I)^2}}{m x_2^2 + I} - \frac{2m x_2 \left(\frac{\lambda_3}{m x_2^2 + I} - 2m x_2 x_3 x_4 \right)}{(m x_2^2 + I)^2} & -\frac{2m x_2 x_4}{m x_2^2 + I} & -\frac{2m x_2 x_3}{m x_2^2 + I} \\ 0 & x_3^2 & 2x_2 x_3 & 0 \end{pmatrix}$$

Top Right of Jacobian:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{(m x_2^2 + I)^2} & 0 \\ 0 & 0 & 0 & \frac{1}{m^2} \end{pmatrix}$$


Bottom Left of Jacobian:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{2\lambda_3 m (\lambda_3 (I - 5m x_2^2) - 2m x_2 x_3 x_4 (m x_2^2 (2I - m x_2^2) - 3))}{(m x_2^2 + I)^4} & \frac{2\lambda_3 m x_4 (I - m x_2^2)}{(m x_2^2 + I)^2} - 2\lambda_4 x_3 & \frac{2\lambda_3 m x_3 (I - m x_2^2)}{(m x_2^2 + I)^2} \\ 0 & \frac{2\lambda_3 m x_4 (I - m x_2^2)}{(m x_2^2 + I)^2} - 2\lambda_4 x_3 & -2\lambda_4 x_2 & \frac{2\lambda_3 m x_2}{m x_2^2 + I} \\ 0 & \frac{2\lambda_3 m x_3 (I - m x_2^2)}{(m x_2^2 + I)^2} & \frac{2\lambda_3 m x_2}{m x_2^2 + I} & 0 \end{pmatrix}$$

Bottom Right of Jacobian:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{(m x_2^2 + I)^2} & 0 \\ 0 & 0 & 0 & \frac{1}{m^2} \end{pmatrix}$$

In total, there are now 40 differential equations. Using a guess, s , these 40 differential equations are put through a 4th Order Runge-Kutta solver. The numerically solved terminal values of x and the given terminal values of x are then compared to see if the error, H , is within a certain tolerance. If it is, then the values of x and λ are given. If the error is out of the range of tolerance, the initial guess of s is updated to be $s_{new} = s_{old} - inv\left(\frac{dH}{ds}\right) * H$ where $\frac{dH}{ds}$ would be a matrix with the final values of Z . This process is then looped until the error is within the given tolerance or until $\frac{dH}{ds}$ is very close to being singular. In the case of the latter, the process is stopped and the most recent values of x and λ are returned.

 (b) The method I chose for this project is the Shooting Method, and within the Shooting Method, I created a 4th Order Runge-Kutta Solver. The reason I chose the Shooting Method is because it seems it is a lot easier to implement especially for a first order system. The Jacobian for such a method seems to be easier to construct than the Finite Difference Method because it is very similar to finding the Jacobian of a system for previous methods we did in class. In the Finite Difference method, I would have to calculate a tridiagonal matrix which I am less experienced in finding. Also, since this project is a boundary value problem, the shooting method is able to narrow down a good estimate of initial conditions for λ_1 , λ_2 , λ_3 , and λ_4 to then solve the full system using another solver. In this case, I used an RK4 solver within the Shooting Method because I can take as small steps as I want to being that it is an explicit method with a small radius of convergence. However, this can also be a disadvantage too because if I wanted to take larger step sizes this would be more suited for an implicit solver rather than an explicit one.

(c) 

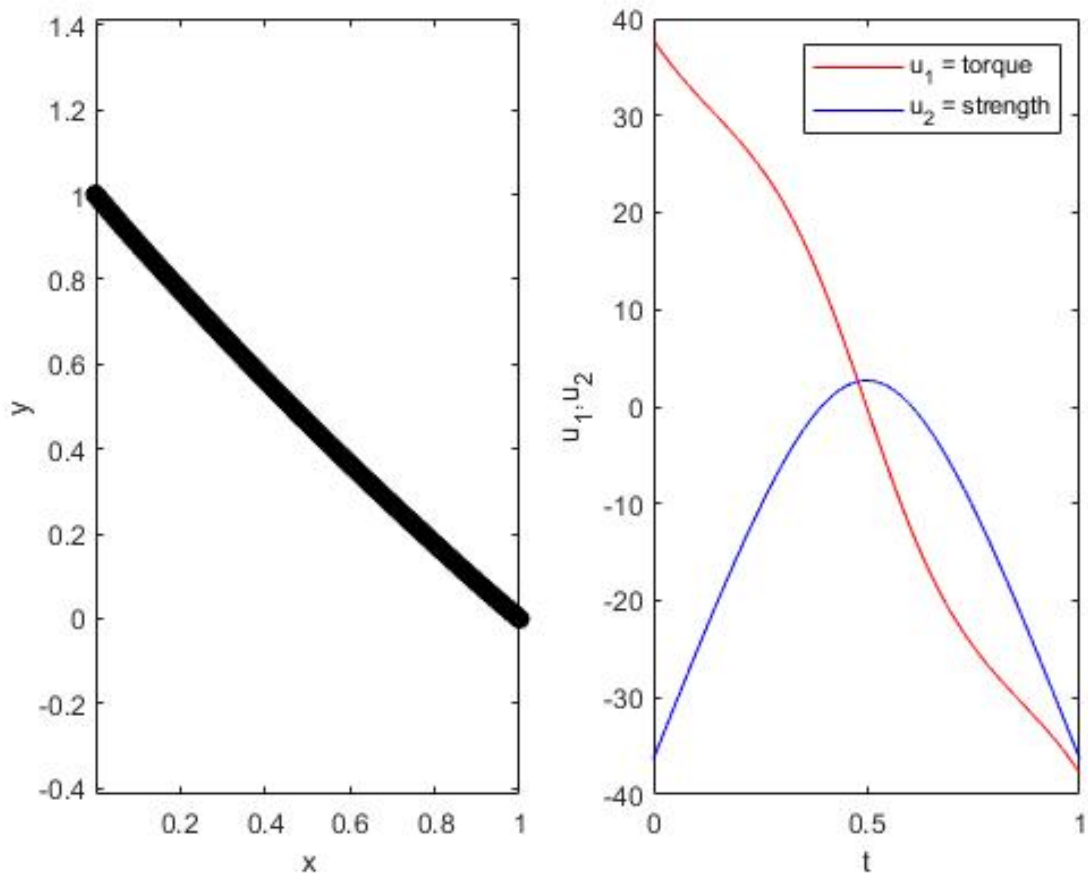


Figure 1: The left graph shows a function of x_1 and x_2 while the right shows the torque and strength as a function of time

After testing my code with the given parameters, the desired graphs were returned suggesting that my code works.



(d) After plugging in the given circle into my shooting method solver, one gets the following graph. The red points indicate when the solver's error did not fall within a certain tolerance and therefore did not converge.

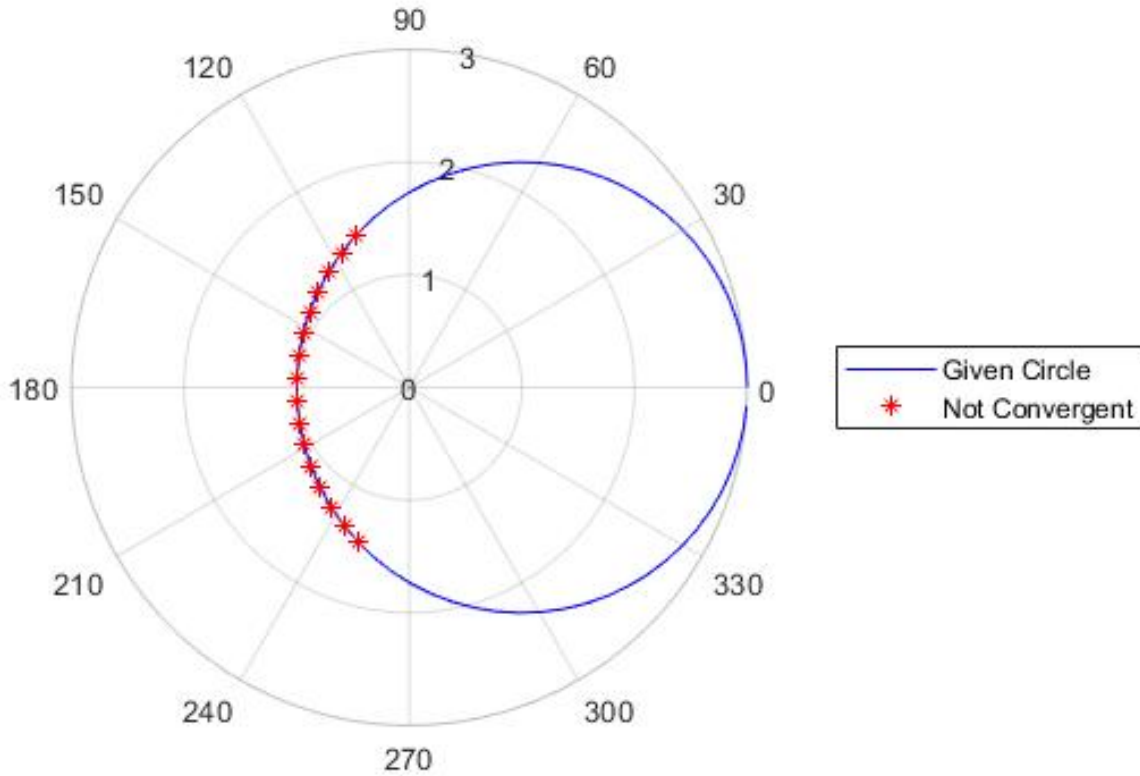


Figure 2: Graph showing which spots on the circle did not converge using the Shooting Method

There is a pattern, and it seems that the Shooting Method fails to maintain a solution with convergence when theta is approximately between just under 120 degrees and just over 240 degrees. This could be because the $\frac{dH}{ds}$ becomes singular or very close to it before arriving at a solution that converges. Such an occurrence may be because the initial guess of $s = 0,0,0$ is not an apt initial prediction for when theta is between these values.