Swing-Up Control of Pendulum Using Classical Control Method

Kahdirdan Kahirman
Department of Mechanical Engineering
University of Washington
Seattle, USA
Email: kadierk@uw.edu

Abstract—The swing-up control of the pendulum is a well-researched yet challenging problem in control theory. Numerous paper has been published showing the swing-up method of the single, double, and triple pendulum. The most interesting one is accomplished by Glück Tobias, Eder Andreas and Kugi Andreas which they have successfully swung up the triple pendulum[1]. The key method they used is designing a constrained feedforward controller to swing up the pendulum and switch back to the linear-quadratic-Gaussian (LQG) controller when pendulum reached upright position. In this paper, we will make a brief summary of their work.

Index Terms—Swing-Up, Pendulum, Feedback control, Constrained feedforward control, Kalman filter, linear-quadratic-Gaussian (LQG).

I. INTRODUCTION

The swing-up of the pendulum on a cart is a quite challenging problem in control theory. Numerous method has been brought up to solve this problem. Maeba Tomohide suggested that swing-up controller could be designed by using energy control method based on Lyapunov function[2]. Pathompong Jaiwat suggested that the double pendulum could be swung up by using nonlinear model predictive control method[3]. Tobias Gluck suggests that the swing-up problem of the pendulum could be achieved by using the combination of feedforward and feedback controller. The design of the feedforward controller is the key part of this method which involves solving two-point boundary value problem[1]. This method is a universal way that will work for the single, double and triple pendulum.

II. EQUATION OF MOTION

In this section, we will derive the mathematical model of a single pendulum on a cart. The derivation method of the equation of motion for single pendulum could be extended to $N$ link pendulum. Nevertheless, due to the complexity of the final equation of motion for a multi-link pendulum, we will only discuss the derivation of equation of motion(EOM) of a single pendulum on a cart system.

This report is the project report of ME-535: Computational Techniques In Mechanical Engineering.
where \( v_1 \) is the velocity of the link 1. \( J_1 \) is the inertia of link 1. \( v_1 \) could be represented by

\[
v_1^2 = x_1^2 + y_1^2
\]  
(4)

and

\[
x_1 = \dot{x} + c_1 \cos \theta_1 \dot{\theta}_1
\]  
(5)

\[
y_1 = -c_1 \sin \theta_1 \dot{\theta}_1
\]  
(6)

Potential energy \( P \) could be written as

\[
P = m_1 g y_1
\]  
(7)

where \( g \) is local gravity constant.

Combine equation (1) to (7) we could get the equation of motion of the single pendulum on a cart system

\[
k_0 \ddot{x} + (M + m_1) \ddot{x} + l_1 m_1 (\ddot{x} + 0.5 \sin \theta_1 \dot{\theta}_1^2 + 0.5 \cos \theta_1 \dot{\theta}_1) = F(t)
\]  
(8)

and

\[
k_1 \dot{\theta}_1 + l_1 m_1 (-0.5 g \sin \theta_1 + 0.5 \cos \theta_1 \ddot{x}) + \frac{1}{3} l_1^2 m_1 \dot{\theta}_1 = 0
\]  
(9)

\[\text{C. State Space Representation}\]

As we denoted before, let’s define \( q = [q_0 \ q_1]^T = [x \ \theta]^T \) as our state vector. By combining equation (8) and (9) we could solve for two non-linear equations.

\[
\frac{d}{dt} \begin{bmatrix} q_0 \\
q_1 \end{bmatrix} = \begin{bmatrix} \dot{x} \\
\dot{\theta}_1 \end{bmatrix} = \begin{bmatrix} f_1 \\
f_2 \\
f_3 \\
f_4 \end{bmatrix}
\]  
(10)

\[
\frac{d}{dt} \begin{bmatrix} q_0 \\
q_1 \\
q_0 \\
q_1 \end{bmatrix} = \begin{bmatrix} \dot{x} \\
\dot{\theta}_1 \\
\dot{x} \\
\dot{\theta}_1 \end{bmatrix} = \begin{bmatrix} f_1 \\
f_2 \\
f_3 \\
f_4 \end{bmatrix}
\]  
(11)

Equation (10) and (11) could be written as

\[
\frac{d}{dt} \begin{bmatrix} q_0 \\
q_1 \\
q_0 \\
q_1 \end{bmatrix} = A \begin{bmatrix} q_0 \\
q_1 \\
q_0 \\
q_1 \end{bmatrix} + Bu
\]  
(12)

where \( A \) and \( B \) is the Jacobian matrix of (12), \( u \) is the control input.

\[\text{D. Define Control Input}\]

In the previous subsection, we have defined our control input as \( F(t) \), an external input. In practice, the cart is driven by a servo motor. Due to the high accuracy of the servo motor control and quick reaction time between pendulum cart and servo drive, we could set cart acceleration \( \ddot{x} \) as our external control input[1]. Then the equation (12) could be written as input-affine system representation

\[
\frac{d}{dt} \begin{bmatrix} q_0 \\
q_1 \\
q_0 \\
q_1 \end{bmatrix} = \begin{bmatrix} \ddot{x} \\
\ddot{\theta}_1 \\
\ddot{x} \\
\ddot{\theta}_1 \end{bmatrix} = \begin{bmatrix} 0 \\
0 \\
1 \\
R_1(\theta_1) \end{bmatrix} \begin{bmatrix} F(t) \end{bmatrix} + \begin{bmatrix} 0 \\
0 \\
1 \\
R_0(\theta_1, \dot{\theta}_1) \end{bmatrix} \dot{\theta}_1
\]  
(13)

where \( \ddot{x} \) serve as control input. Reform the pendulum equation of motion in form of (14) is necessary for our later analysis. The benefit of taking acceleration as control input is that we could get rid of the damping coefficient \( k_0 \) and cart mass \( M \) out of our equations. In practice the precise cart mass is quite hard to determine, thus eliminate the cart mass from our equation is preferred. Also, once we find out the desired control input \( \ddot{x} \) we could then take integration on the theoretical control input \( \ddot{x} \) and get the desired cart position \( x \) and desired cart velocity \( \dot{x} \) for the actual implementation purpose, which means the position control or the velocity control of the servo motor.

\[\text{III. Parameter Estimation}\]

As we mentioned in the previous section, for a better approximation of the actual system, we have to model the damping coefficient into our system model. However, there is no way to directly measure the damping coefficient. Thus the parameter estimation is one of the most crucial parts of the system model derivation. The mass of the pendulum arm \( m_1 \) could be directly measured by using the scale, the length \( l_1 \) and the height of the center of mass \( c_1 \) could be obtained by using Solidworks software, so does the moment of inertia \( J_1 \). The only parameters we have to identify for the single pendulum is the damping constant \( k_1 \). To do the parameter estimation, we first have to mount the pendulum cart and swing the pendulum arm so that we could get measurement data \( D_m \). After that, we could use Least-Square identification to determine the parameters \( k_1 \) such that

\[
\min J = \frac{1}{T_m} \int_0^{T_m} (D_s(k_1, t) - D_m(t))^2 dt
\]  
(14)

where \( D_s(k_1, t) \) is the simulation model of the output data at time \( t \), \( D_m(t) \) is the measurement data of actual system at time \( t \). \( T_m \) is the measurement data time interval[1].

\[\text{IV. Control Method}\]

The key idea of the swing-up control of the pendulum system is designing the feedforward controller for the swing-up process. The schematic diagram could be seen as Fig.2.
A. Feedforward Controller Design

Feedforward controller design is the key part of controller design. To start with, we need to rewrite the (14) into the following form.

\[ \dot{x} = u \] (16)

\[ \ddot{\theta}_1 = R_0(\theta_1, \dot{\theta}_1) + R_1(\theta_1)u \] (17)

where \( u \) is the control input and is equal to the acceleration of the cart.

Let’s assume that we need \( T \) seconds to swing up the pendulum from the subjacent position. Then we could easily obtain some boundary conditions. To start with, we know that the pendulum cart position is at zero position when \( t = 0 \), and the cart does not have initial velocity and initial acceleration. Besides that, we also know when \( t = T \), the pendulum should be swung-up to the upright position, along with the cart position and velocity should be zero. We could represent this in the following form.

\[ x(0) = 0, \dot{x} = 0, \ddot{x}(0) = 0 \] (18)

\[ x(T) = 0, \dot{x}(T) = 0, \ddot{x}(T) = 0 \] (19)

\[ \theta(0) = \pi, \dot{\theta} = 0, \ddot{\theta}(0) = 0 \] (20)

\[ \theta(T) = 0, \dot{\theta}(T) = 0, \ddot{\theta}(T) = 0 \] (21)

Equation (16) to (21) forms a well defined two-point-boundary value problem (BVP). We could solve for our desired control input \( u \) by solving BVP defined by (16) to (21). The difficulties of solving the BVP will differ based on whether we have input constraint or not, the difficulties of solving the constrained problem are higher than the unconstrained problem.

1) Unconstrained Case: Suppose that we don’t have any input constraint for the pendulum which means our cart rail is infinite long and our servo motor does not have any output limits, then we could define our control input \( u \) as a function with unknown parameters such that \( u \) will satisfy boundary condition (18) and (19). The number of free parameters we should have is the number of the boundary condition of first-order equation minus the number of first-order equations we have. For example, from (14) we could know we have 4 first order ODE thus we could have 8 boundary conditions for \( x, \dot{x}, \theta, \dot{\theta} \) at \( t = 0 \) and \( t = T \). Thus, for single pendulum, we could have \( 8 - 4 = 4 \) free parameters.

The control input \( u \) could be any function that satisfy the boundary condition. For example, we could define our control input \( u \) as

\[ u = a_0 + a_1 \cos \frac{\pi t}{T} + \sum_{i=1}^{4} p_i \cos \frac{i\pi t}{T} \] (22)

where \( p_1, p_2, p_3, p_4 \) are free parameters we add and \( a_0 \) and \( a_1 \) is a function of free parameters. The control input could also be constructed by using the polynomial function.

2) Constrained Case: In practice, we always have to consider the constraint for our pendulum system. For example, our cart rail is not infinitely long and our motor has maximum output limit. Those are some important parameters that we have to take into consideration when we designing the feedforward control input. The basic idea of constrained feedforward controller design is map the unconstrained control input into a new constrained coordinate such that the new projected control input satisfy the system constraint [1]. The procedure could be summarized as follow.

First let’s introduce a new saturation function to represent the desired cart position. That is, we map the desired the cart position \( x^* \) to a new coordinate. This could be represented by

\[ x^* = \psi_1(\zeta_1, \psi_1^\pm) \] (23)

where \( \psi_1 \) is the new state in projected coordinate. If we take derivative with respect to time in above equation, we could have the cart velocity and cart acceleration as follow

\[ \dot{x}^* = \frac{\partial \psi_1}{\partial \zeta_1} \zeta_1 \] (24)

\[ \ddot{x}^* = \frac{\partial \psi_1}{\partial \zeta_1} + \frac{\partial^2 \psi_1}{\partial \zeta_1^2} \zeta_1^2 \] (25)

According to (24) and (25), we could then define two new saturation function \( \psi_2 \) and \( \psi_3 \) where

\[ \dot{\zeta}_1 = \psi_2(z, \psi_3^\pm) \] (26)

and

\[ \dot{\zeta}_2 = \psi_3(z, \psi_3^\pm) \] (27)

The \( z \) is the new input of our system. The pendulum’s cart position, cart velocity, cart acceleration has to satisfy the constraint

\[ x^- < x^* < x^+, \dot{x}^- < \dot{x}^* < \dot{x}^+, \ddot{x}^- < \ddot{x}^* < \ddot{x}^+ \] (28)

where \( x^- , x^+ \) is the minimum and maximum position of the cart, \( \dot{x}^- , \dot{x}^+ \) is the minimum and maximum velocity of the cart, \( \ddot{x}^- , \ddot{x}^+ \) is the minimum and maximum acceleration of the cart, which is also our minimum and maximum control input. The \( \psi_i^\pm \) is the maximum and minimum boundary of the saturation function. The first and the second saturation function could be represented by

\[ \psi_i(\zeta_i, \psi_i^\pm) = \psi_i^+ - \frac{\psi_i^+ - \psi_i^-}{1 + \exp c \psi_i} \] (29)

for \( i = 1, 2, \) and \( c = 4/(\psi_i^+ - \psi_i^-) \) so that \( \partial \psi_i/\partial \zeta_i = 1 \) at \( \zeta_i = 0 \). The final saturation function could be chosen as a ramp shape saturation function since we don’t have any constraint on the motor jerk. The saturation function \( \psi_3 \) could be seen represented as follow

\[ \psi_3 = \begin{cases} \psi_3^+, & z > \psi_3^+ \\ z, & z \in [\psi_3^-, \psi_3^+] \\ \psi_3^-, & z < \psi_3^- \end{cases} \] (30)

The saturation function \( \psi_i \) for \( i = 1, 2 \) could be seen as Fig.3[4].
The $z$ is the new setup function we have to use in the constrained case. The setup function could be any function that satisfies the condition

$$z(0, p) = z(T, p) = 0$$  \hspace{1cm} (31)

One possible choice of the the function $z$ is

$$z(t, p) = \sum_{i=1}^{n} p_i \sin \frac{h_i \pi t}{T}$$  \hspace{1cm} (32)

where $p_i$ is the free parameter, $h_i$ is the frequency of the sinusoid series, $T$ is the swing up time.

3) Numerical Solution: To solve the BVP problem in both constrained or unconstrained case, we need to use the BVP5C solver in Matlab. Matlab’s BVP5C solver uses a collocation method to solve BVP problem. The key part that determines whether you could find out the converged solution or not is the determination of the swing-up time $T$ and set-up function $u$. As we mentioned before, any function $u$ that satisfies the boundary condition could be used as the set-up function and we could use that to solve for BVP problem[1]. However, the choose of the set-up function somehow restricts the space of possible solution. Moreover, the swing-up time is also one key parameter that we have to determine. If the swing-up time is too short then the motor might not able to provide enough energy to swing up the pendulum. Nevertheless, if the swing-up time is too long, the solution of BVP problem may not exist. Thus, the practical way to determine the swing-up time is using brute force method in certain time region between 0.5 and 5, with the time step as 0.01. If we choose our control input like (22) as a trigonometric function, then we also have to determine the frequency of sinusoidal function by running a a for a loop.

B. Feedback Controller Design and State Estimation Design

The feedback controller design of the pendulum along with state estimator is a standard task in control theory. Due to the simplicity of the design, we will not go into the mathematical derivation details of the feedback controller and state estimator. Nevertheless, we will provide some instruction on how to design the controller and estimator.

1) Feedback Controller Design: The linearize version of equation of motion of the pendulum could be seen as (13). From the (13) we could obtain the Jacobian matrix of the pendulum system. We could provide this Jacobian matrix $A, B$ to Matlab and solve for feedback gain $K$. To obtain the feedback gain we also have to define the $Q$ and $R$ matrix such that we could minimize

$$\min J = \int_{0}^{\infty} (x^T Q x + u^T R u) \, dt$$  \hspace{1cm} (33)

under constraint (13). In Matlab we could directly solve for feedback gain using $K = lqr(A, B, Q, R)$. Please note that our Jacobian matrix is time variant, thus the feedback gain we obtained should also be time variant. It’s better we denote the time variant matrix matrix $A$ as $A_t$, and $B$ as $B_t$.

2) State Estimator Design: In this subsection, we will implement Kalman filter to our non-linear system for state estimation. We will first do a brief introduction to Kalman filter algorithm and then implement it in the closed-loop linearized system. Kalman filter was invented by Swerling and Kalman, it is a technique for filtering and prediction in linear Gaussian systems[5]. The algorithm of the Kalman filter could be seen as follow:

Algorithm 1: Algorithm for Kalman filter[5]

\begin{itemize}
  \item \textbf{Input:} $x_{t-1}, \sum_{t-1}, u_t, y_t$
  \item \textbf{Output:} $x_t, \sum_t$ (output)
  \item \textbf{Initialisation :}
  \begin{itemize}
    \item $\bar{x}_t = A_t x_{t-1} + B_t u_t$
    \item $\sum_t = A_t \sum_{t-1} A_t^T + Q_t$
    \item $K_t = \sum_t C_t^T (C_t \sum_t C_t^T + R_t)^{-1}$
    \item $x_t = \bar{x}_t + K_t (y_t - C_t \bar{x}_t)$
    \item $\sum_t = (I - K_t C_t) \sum_t$
  \end{itemize}
  \item \textbf{return} $x_t, \sum_t$
\end{itemize}

In Algorithm 1, $A_t$, $B_t$ is our state transition matrix and control input matrix. The $y_t$ is our measurement, $u_t$ is our control input, $\sum_t$ is our covariance. $Q_t$ is our mean of posterior state, $R_t$ is covariance of measurement noise. $\bar{x}_t$ is our predicted belief, $\sum_t$ is our predicted covariance. Step 1 to 3 is usually called prediction while step 4 an 5 is called measurement update[5].

3) Remark:

1) The feedback gain $K$ is obtained by linearizing the non-linear pendulum model around the desired trajectory. The corresponding $K$ values will be store in Simulink lookup table to decrease the computational time needed.

2) We will use the Extended-Kalman Filter as state estimator. We could also use Unscented Kalman Filter or particle filter for better estimation due to the high non-linearity of the pendulum system. Nevertheless, this will increase the calculation time needed during real-time control, thus it’s better that we use the EKF first and make sure we have enough computational power.

3) The pendulum is not controllable during the certain time period during the swing-up process. Thus, we will only apply the LQG control when the pendulum is near the region of the upright position. Which means we first use...
feedforward controller to swing up the pendulum. When the pendulum is close enough to the upright position we will apply both the feedforward and feedback controller. And after swing-up time $T$ we will use LQG controller only.

C. Summary

The design process of pendulum swing up controller could be summarized as follow.

1) Derive the equation of motion of the pendulum system and write it as an input-affine system representation.
2) Using Least-Square method to find out the system parameters.
3) Determine the boundary condition of the system and the number of free parameters we could use for set-up function.
4) Design the set-up function.
5) Using Matlab parallel computing toolbox and BVP5C solver to solve the numerical solution of desired control input.
6) Design LQG controller.
7) Run the simulation and verify the solution.

V. SIMULATION RESULT

In this section, we will provide the simulation result of desired swing-up trajectory of the single, double and triple pendulum.

A. Single Pendulum

Let’s choose the system parameters as follow. Pendulum arm length $l_1 = 0.4m$, cart mass $M = 1kg$, pendulum arm mass $m_1 = 0.8kg$, cart rail damping coefficient 0.005, pendulum arm bearing damping coefficient $k_1 = 0.001$, local gravity $g = 9.81$. Swing-up time of pendulum is chosen as $T = 3.15s$ after testing a bunch of swing-up time parameters. After running the BVP5C solver in Matlab we could get the result shown as Fig.4 to Fig.8.

As we could see from the Fig.4 our desired cart acceleration $\ddot{x}$ satisfy the boundary condition which is zero at the initial time and final time. From Fig.5 we could see that the initial
pendulum arm angle is $-\pi$, and it reaches the 0 position at $T = 3.15s$. From Fig.6 we could see that the desired cart position also satisfy the boundary conditions. The cart could go back to its initial position at final time $T$.

From Fig.6 we could see that the desired cart position also satisfy the boundary conditions. The cart could go back to its initial position at final time $T$.

Due to the simplicity of the single pendulum, the swing up time is pretty easy to determine and the numerical solution provided by the Matlab BVP5C solver is smooth, and the final answer could converge.

The set-up function we used for this single pendulum is

$$V_s = \sum_{h=1}^{4} p_i \sin \frac{h_i \pi t}{T}$$

(34)

where $h_i$ is $1, 2, 3, 4$ correspondingly. The free parameters is solved by Matlab BVP5C solver and its value is $p_1 = 1.2021, p_2 = 2.0341, p_3 = -1.9431, p_4 = -2.9345$. The constraint of the system we used in this example is: cart rail length $x \in [-0.7, 0.7], \dot{x} \in [-3, 3], \ddot{x} \in [-21, 21]$. We could see from the final result shown in Fig.4 to Fig.8 that our calculation result fits our constraint.

Since solving the BVP problem with given constraint is relatively hard, and the perfect solution may not exist, it is recommended that we solve for the desired control input without any constraint first and see whether the solution we get satisfy the constraint. If the solution we get does not satisfy the boundary conditions then we could choose to design constrained feedforward controller.

B. Double Pendulum

For simulation of the double pendulum, let’s assume that $l_1 = 0.25m, l_2 = 0.25m, M = 1kg, m_1 = 0.1kg, m_2 = 0.1kg, k_0 = k_1 = k_2 = 0.005, g = 9.81, J_1 = \frac{1}{12}m_1 l_1^2, J_2 = \frac{1}{12}m_2 l_2^2$. The constraint of the system we used in this example is: cart rail length $x \in [-0.7, 0.7], \dot{x} \in [-5, 5], \ddot{x} \in [-40, 40]$. The final result using this set of parameters could be seen from Fig.9 to Fig.15. The swing-up time is determined using the for loop and its value is $T = 1.06s$. The set-up function for the double pendulum has 6 free-parameters, and it has the form:

$$V_s = \sum_{h=1}^{6} p_i \sin \frac{h_i \pi t}{T}$$

(35)

From the Fig.9 we could see the desired cart position that we calculated. It shows that our cart starts at an initial position and goes back to the initial position at the final moment.
The corresponding control input we get could be seen in Fig.11. We could see from the plot that we reached maximum control input constraint. This suggests that physical constraint will also affect the solution we get.

We could also get the angular velocity of the pendulum arm. The angular velocity of the pendulum arm could be seen from Fig.14 and Fig.15.

The angle of two pendulum arm could be seen from Fig.12 and Fig.13. We could see from the Fig.12 that the pendulum arm angle can't reach the perfect upright position but it's pretty close to it. Again this result shows the difficulty to get the perfect solution.

We could see from the Fig.14 and Fig.15 that the angular velocity is relatively high when compared with Fig.7. And it's not a very smooth line. It has a sudden change around t = 0.8s.

C. Triple Pendulum

Due to the high non-linearity of the triple pendulum system, the desired control input is even more difficult to achieve. For the demonstration purpose, we will use the same parameters shown in [1]. Take \( m_1 = 0.876kg \), \( m_2 = 0.938kg \), \( m_3 = 0.553kg \), \( c_1 = 0.215m \), \( c_2 = 0.269m \), \( c_3 = 0.226m \), \( l_1 = 0.323m \), \( l_2 = 0.419m \), \( l_3 = 0.484m \), \( J_1 = 0.013Nms^2 \), \( J_2 = 0.024Nms^2 \), \( J_3 = 0.018Nms^2 \), \( d_1 = 0.215 \), \( d_2 = 0.002 \), \( d_3 = 0.002 \). The swing-up time we used is \( T = 3.5s \). The set-up function has 8 free parameters and has the form

\[
V_s = \sum_{h=1}^{8} p_i \sin \frac{h \pi t}{T}.
\]
where \( h_i \) is 2 to 9 correspondingly. The constraint of the system we used in this example is: cart rail length \( x \in [-0.7, 0.7] \), \( \dot{x} \in [-5, 5] \), \( \ddot{x} \in [-22, 22] \). The result we obtained could be seen from Fig.16 to Fig.18.

From Fig.16 to Fig.18 we could see that the pendulum arm angular position we solved is not a smooth curve. Moreover, none of the pendulum arms could reach the upright position and it’s not even close to it. We could compare our answer to Fig.19[1].

From the comparison, we could see a huge difference. The reason this mismatch happens might be:

1) The mismatch between gravity constant \( g \). The paper [1] did not mention what’s the \( g \) that they used thus we might used a different parameters. Even a small mismatch (for example, 0.001) between the parameters will cause a huge difference in BVP numerical result for triple pendulum.

2) Numerical error during the solving process.

To fix this issue, I have to recheck the code I wrote and I am currently working on that. I might also have to use a different BVP solver package to fix this issue.

VI. NUMERICAL ERROR ANALYSIS

When we simulate the pendulum system we might encounter numerical error caused by ODE function or BVP5C function. In this section, we will discuss the possible numerical error caused by ODE solver in Matlab. We will also look into the possible way to improve the accuracy, efficiency and robustness of BVP5C solver.

A. Numerical Error Analysis of ODE Function

As we mentioned before, due to the high non-linearity of the pendulum equation of motion (especially for double and triple pendulum), it is necessary to verify the accuracy of the ODE solver we used in the simulation process. In Matlab, the most frequently used ODE solver is ode45. The ode45 solver solves the non-stiff differential equation using 4th order Runge Kuta method. Since the pendulum arm could have a very quick change in its angular velocity, the non-stiff ODE solver might not be accurate enough for stiff problem like this. Thus, we could try to use a stiff ODE solver like ode15s or ode23tb and compare the result with ode45. Moreover, we could also use a higher order Runge Kuta method to improve
the numerical accuracy, like ode87, which integrates a system of ordinary differential equations using 8-7th order Dormand and Prince formulas. The ode87 file we used could be find at Matlab file exchange website[6]. The basic idea of how to test the simulation accuracy is using the idea of energy conservation. In simulation environment we will mount the pendulum cart and release the pendulum arm form certain position, then according to the law of energy conservation, we should note that the total energy of the pendulum arm won’t change during the swinging process. That is, the initial energy minus the pendulum energy at any other time period should be zero. And please note this is true only if we set our viscous damping coefficient to zero. The result we obtained could be seen from Fig.20 to Fig.23.

From Fig.20 to 23 we could see that all the ode solver works fine and passes the numerical error test. Nevertheless, since the ode15s uses a low order method, it has a larger numerical error. Thus it is preferred to use the higher order method.

B. Numerical Error Analysis of BVP5C Function

As we could see from the simulation result in section V-B and C, we could see sometimes it’s quite difficult to solve the BVP problem for the double and triple with given constraint. In most case the Matlab BVP solver will throw error to us and indicate it encounters singular Jacobian matrix during the numerical solving process. Or it will tell us that the final result it solved can’t converge. The only way I have find so far to fix this issue is to try a different setup function for the pendulum which could be seen from [1], or try to provide analytical Jacobian matrix to BVP solver to improve the efficiency, robustness, and accuracy. The Matlab’s BVP solver will calculate the Jacobian matrix numerically if the user don’t provide it. Thus, for some really complicated problem like the one we dealing with, the key of success is the provision of the Jacobian matrix[7]. The detail of providing the Jacobian for BVP solver could be seen from [7]. The final result we obtained could be seen as follow.

1) Single Pendulum: The single pendulum swing up problem is actually pretty easy to solve. After we use the same parameters like we used in section V A, we could have a similar result like before. Moreover, since we provided the Jacobian matrix to the BVP solver, it becomes more robust, since it allow us to solve for additional desired trajectory for the swing up time for T=3.11 s, T=3.12 s, T=3.13 s, and T=3.14 s without throwing any error, which is not the case if the Jacobian matrix is not provided. Also, it takes less time to compute the problem. After providing the Jacobian matrix it takes about 3.2 seconds to calculate the solution, which is about 25 % faster than without using Jacobian matrix, which is 4.2 seconds.

2) Double Pendulum: As we mentioned in the section V B, the desired trajectory we got for the double pendulum swing up problem is not smooth, and when we obtain those figures, Matlab throws a warning suggesting that the solution is not converged. This usually indicates the solution we get is not desirable. Even though it looks like the pendulum is quite close to the upright position, when you apply the solved trajectory
(that can’t converge) for simulation, it will give you a pretty
different trajectory than what you obtained before. This is
due to the fact that this result can’t satisfy the equation of
motion of the double pendulum. As we mentioned before, try
a new set-up function or providing Jacobian matrix will fix
that problem. After providing the Jacobian matrix and using
the set up function with the form (22), the new result could
be seen from Fig.24 to Fig.27 (please note that the swing up
time is also changed).

3) Triple Pendulum: The Jacobian matrix related to con-
strained triple pendulum swing up problem is extremally
complex. I used Matlab symbolic calculation toolbox, and it
took me around 12 hours to get the final calculation result.
Moreover, the Jacobian matrix I calculated is too long that the
Matlab command window can’t express it fully. Thus, I used
Matlab’s ‘matlabFunction’ command to automatically generate
the function file. This process will take about 15 minutes. After
I provided this calculation result to the Matlab, the Matlab
gives me an error and indicates that it’s out of memory. This
case is really rare, I believe this problem is caused by the
fact that the three saturation function we used will greatly
complicate the Jacobian matrix since it’s coupled with the
equation of motion of the triple pendulum. The reference [1]
doesn’t point out how they solved this issue nor did they
released the source code of their research, thus, unfortunatelly
I am bot able to recreate the result they got. It is likely that
the correct way to represent the Jacobian matrix is the key to
success.

VII. Discussion

In this paper, we briefly discussed how to design a swing-
up controller for the pendulum system. And we could get the
desired solution for the single pendulum system. Nevertheless,
we still have some difficulties when designing the feedforward
controller for triple pendulum.

What we have to know is that the method used in this paper
is a universal method for designing the swing-up controller.
We could also choose other methods such as non-linear model
predictive control to swing-up the pendulum like [3] did.

Right now the key thing I have to do is double check the
code and see whether there’s any mistake I made that could
explain why the BVP solver having a trouble to solve for
perfect control input for triple pendulum and try to use a
different method for swing-up problem.

VIII. Future Scope

The future scope of the project is try to use machine learning
algorithm to swing-up the pendulum.
ACKNOWLEDGMENT

The mathematical derivation of control method in this paper has a large portion of similarities with [1]. Glück, T., Eder, A., Kugi are the ones who brings up the swing up method of the triple pendulum. The author of this paper does not claim any right of bring up the swing up method, for the detailed mathematical derivation of swing-up method, please check the reference [1]. The simulation result we obtained in this paper though, is original work.

REFERENCES


