MAE 4730

Cornell University

MAE 4730 Final Report

Author:
Mitchell Dominguez

NetID:
md697

Instructor:
Andy Ruina

December 8, 2017

Contents

1 The Triple Pendulum

1.1 Derivation of the Equations of Motion

1.1.1 Model

The triple pendulum is modeled as shown in Figure 1. One end of the pendulum is pinned at Point O, which is inertially fixed. All hinges are frictionless, and the only external forces acting on the entire system are the gravitational forces on each link of the pendulum. The bars are not modeled as point masses, but rather as distributed masses with a moment of inertia defined about a center of mass (COM) G. In this report, the centers of mass are defined to be at the centers of each bar, although code could easily be modified to solve the pendulum for arbitrary COM location.

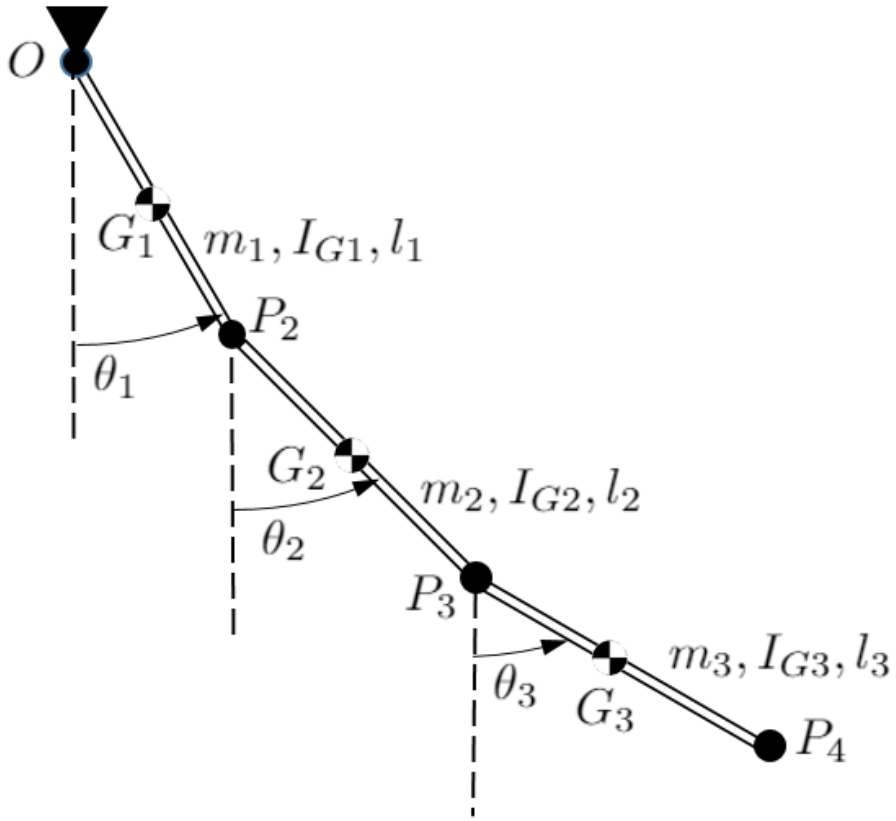

Figure 1: Model of the triple pendulum.

1.1.2 Reference Frames

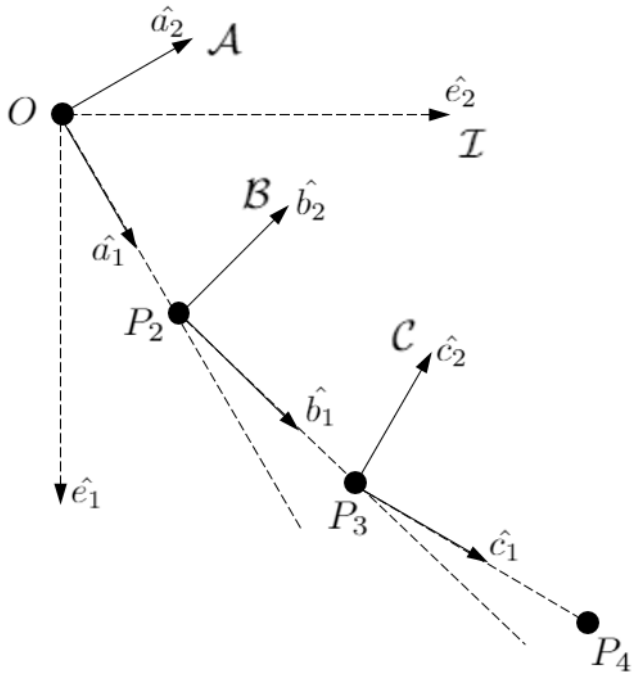

Figure 2: Definition of the inertial frame ($\mathcal{I}$) and the body frames ($\mathcal{A}, \mathcal{B}, \mathcal{C}$) attached to each link of the pendulum.

In order to derive the equations of motion, reference frames first need to be defined for the problem, as can be seen in Figure 2. The frames are related to each other as shown in Table 1. Each angle, defined in Figure 1, is measured from the vertical. As such, the orientation of each body frame is defined solely by the angle it makes with the inertial vertical axis, and is independent of the other body frames.

Table 1: Relationship between the inertial frame and the body frames

	$\hat{\boldsymbol{a_1}}$	$\hat{\boldsymbol{a_2}}$	$\hat{\boldsymbol{b_1}}$	$\hat{\boldsymbol{b_2}}$	$\hat{\boldsymbol{c_1}}$	$\hat{\boldsymbol{c_2}}$
$\hat{\boldsymbol{e_1}}$	$\cos\theta_1$	$-\sin\theta_1$	$\cos\theta_2$	$-\sin\theta_2$	$\cos\theta_3$	$-\sin\theta_3$
$\hat{\boldsymbol{e_2}}$	$\sin\theta_1$	$\cos\theta_1$	$\sin\theta_2$	$\cos\theta_2$	$\sin\theta_3$	$\cos\theta_3$

1.1.3 Free Body Diagrams

Figure 3 shows the free body diagrams (FBDs) for each individual link of the pendulum. These exact free body diagrams will be used in 1.1.7 to solve the pendulum using Differential Algebraic Equations (DAEs). When deriving the Equations of Motion (EOMs) in 1.1.6, these FBDs will be

combined and angular momentum will be balanced such that the constraint forces F_{xi} and F_{yi} are eliminated. When using Lagrange's equation to solve for the EOMs in 1.1.8, these FBDs will not be explicitly used, although the final results match with the aforementioned methods that rely on these free body diagrams.

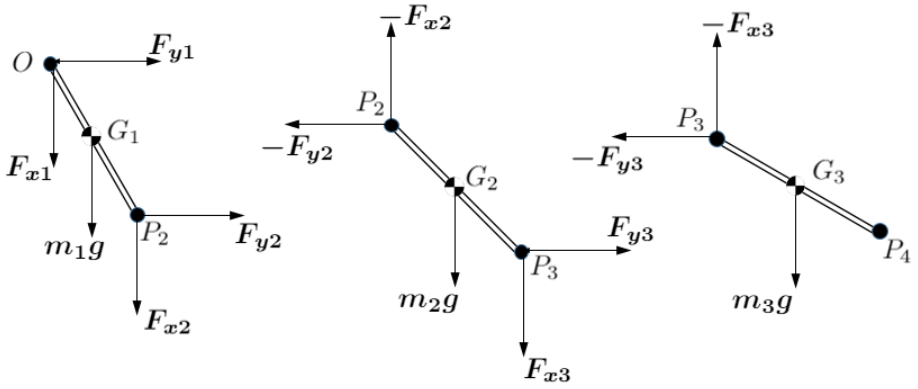

Figure 3: The free body diagrams showing all of the external forces acting on each link of the pendulum.

1.1.4 Polar Kinematics

Before solving for the equations of motion, it is first necessary to calculate the kinematics of the points of interest on the triple pendulum, namely the centers of mass and the endpoints. The kinematics for the three links of the triple pendulum are shown below. While all quantities in Equations 1-6 are relative to the origin of the inertial frame, relative quantities can be obtained by subtraction of inertial ones (e.g. $\boldsymbol{r}_{G3/P2} = \boldsymbol{r}_{G3/O} - \boldsymbol{r}_{P2/O}$). All relative quantities referenced later in the report will be assumed to have been derived as such. The code that was used to derive the triple pendulum polar kinematics with MATLAB's symbolic toolbox can be found in A.2.

Arm 1

$$\begin{aligned} \boldsymbol{r}_{G1/O} &= \frac{l_1}{2}\hat{\boldsymbol{a}}_{\boldsymbol{1}} \\ {}^{\mathcal{I}}\boldsymbol{v}_{G1/O} &= \frac{l_1}{2}\dot{\theta}_1\hat{\boldsymbol{a}}_{\boldsymbol{2}} \\ {}^{\mathcal{I}}\boldsymbol{a}_{G1/O} &= \frac{l_1}{2}\ddot{\theta}_1\hat{\boldsymbol{a}}_{\boldsymbol{2}} - \frac{l_1}{2}\dot{\theta}_1{}^2\hat{\boldsymbol{a}}_{\boldsymbol{1}} \end{aligned} \tag{1}$$

$$\begin{aligned} \boldsymbol{r}_{P2/O} &= l_1\hat{\boldsymbol{a}}_{\boldsymbol{1}} \\ {}^{\mathcal{I}}\boldsymbol{v}_{P2/O} &= l_1\dot{\theta}_1\hat{\boldsymbol{a}}_{\boldsymbol{2}} \\ {}^{\mathcal{I}}\boldsymbol{a}_{P2/O} &= l_1\ddot{\theta}_1\hat{\boldsymbol{a}}_{\boldsymbol{2}} - l_1\dot{\theta}_1{}^2\hat{\boldsymbol{a}}_{\boldsymbol{1}} \end{aligned} \tag{2}$$

Arm 2

$$
\begin{aligned}
\boldsymbol{r}_{G2/O} &= l_1\hat{\boldsymbol{a_1}} + \frac{l_2}{2}\hat{\boldsymbol{b_1}} \\
{}^{\mathcal{I}}\boldsymbol{v}_{G2/O} &= l_1\dot{\theta}_1\hat{\boldsymbol{a_2}} + \frac{l_2}{2}\dot{\theta}_2\hat{\boldsymbol{b_2}} \\
{}^{\mathcal{I}}\boldsymbol{a}_{G2/O} &= l_1\ddot{\theta}_1\hat{\boldsymbol{a_2}} - l_1{\dot{\theta}_1}^2\hat{\boldsymbol{a_1}} + \frac{l_2}{2}\ddot{\theta}_2\hat{\boldsymbol{b_2}} - \frac{l_2}{2}{\dot{\theta}_2}^2\hat{\boldsymbol{b_1}}
\end{aligned} \tag{3}
$$

$$
\begin{aligned}
\boldsymbol{r}_{P3/O} &= l_1\hat{\boldsymbol{a_1}} + l_2\hat{\boldsymbol{b_1}} \\
{}^{\mathcal{I}}\boldsymbol{v}_{P3/O} &= l_1\dot{\theta}_1\hat{\boldsymbol{a_2}} + l_2\dot{\theta}_2\hat{\boldsymbol{b_2}} \\
{}^{\mathcal{I}}\boldsymbol{a}_{P3/O} &= l_1\ddot{\theta}_1\hat{\boldsymbol{a_2}} - l_1{\dot{\theta}_1}^2\hat{\boldsymbol{a_1}} + l_2\ddot{\theta}_2\hat{\boldsymbol{b_2}} - l_2{\dot{\theta}_2}^2\hat{\boldsymbol{b_1}}
\end{aligned} \tag{4}
$$

Arm 3

$$
\begin{aligned}
\boldsymbol{r}_{G3/O} &= l_1\hat{\boldsymbol{a_1}} + l_2\hat{\boldsymbol{b_1}} + \frac{l_3}{2}\hat{\boldsymbol{c_1}} \\
{}^{\mathcal{I}}\boldsymbol{v}_{G3/O} &= l_1\dot{\theta}_1\hat{\boldsymbol{a_2}} + l_2\dot{\theta}_2\hat{\boldsymbol{b_2}} + \frac{l_3}{2}\dot{\theta}_3\hat{\boldsymbol{c_2}} \\
{}^{\mathcal{I}}\boldsymbol{a}_{G3/O} &= l_1\ddot{\theta}_1\hat{\boldsymbol{a_2}} - l_1{\dot{\theta}_1}^2\hat{\boldsymbol{a_1}} + l_2\ddot{\theta}_2\hat{\boldsymbol{b_2}} - l_2{\dot{\theta}_2}^2\hat{\boldsymbol{b_1}} + \frac{l_3}{2}\ddot{\theta}_3\hat{\boldsymbol{c_2}} - \frac{l_3}{2}{\dot{\theta}_3}^2\hat{\boldsymbol{c_1}}
\end{aligned} \tag{5}
$$

$$
\begin{aligned}
\boldsymbol{r}_{P4/O} &= l_1\hat{\boldsymbol{a_1}} + l_2\hat{\boldsymbol{b_1}} + l_3\hat{\boldsymbol{c_1}} \\
{}^{\mathcal{I}}\boldsymbol{v}_{P4/O} &= l_1\dot{\theta}_1\hat{\boldsymbol{a_2}} + l_2\dot{\theta}_2\hat{\boldsymbol{b_2}} + l_3\dot{\theta}_3\hat{\boldsymbol{c_2}} \\
{}^{\mathcal{I}}\boldsymbol{a}_{P4/O} &= l_1\ddot{\theta}_1\hat{\boldsymbol{a_2}} - l_1{\dot{\theta}_1}^2\hat{\boldsymbol{a_1}} + l_2\ddot{\theta}_2\hat{\boldsymbol{b_2}} - l_2{\dot{\theta}_2}^2\hat{\boldsymbol{b_1}} + l_3\ddot{\theta}_3\hat{\boldsymbol{c_2}} - l_3{\dot{\theta}_3}^2\hat{\boldsymbol{c_1}}
\end{aligned} \tag{6}
$$

1.1.5 Cartesian Kinematics

To perform the derivation in 1.1.7, it was also necessary to obtain the kinematics in Cartesian coordinates. These are significantly simpler than those in 1.1.4, and are shown below.

Arm 1

$$
\begin{aligned}
\boldsymbol{r}_{G1/O} &= x_1\hat{\boldsymbol{e_1}} + y_1\hat{\boldsymbol{e_2}} \\
{}^{\mathcal{I}}\boldsymbol{v}_{G1/O} &= \dot{x}_1\hat{\boldsymbol{e_1}} + \dot{y}_1\hat{\boldsymbol{e_2}} \\
{}^{\mathcal{I}}\boldsymbol{a}_{G3/O} &= \ddot{x}_3\hat{\boldsymbol{e_1}} + \ddot{y}_3\hat{\boldsymbol{e_2}}
\end{aligned} \tag{7}
$$

Arm 2

$$\begin{aligned} \boldsymbol{r}_{G2/O} &= x_2\hat{\boldsymbol{e}_1} + y_2\hat{\boldsymbol{e}_2} \\ {}^{\mathcal{I}}\boldsymbol{v}_{G2/O} &= \dot{x}_2\hat{\boldsymbol{e}_1} + \dot{y}_2\hat{\boldsymbol{e}_2} \\ {}^{\mathcal{I}}\boldsymbol{a}_{G2/O} &= \ddot{x}_2\hat{\boldsymbol{e}_1} + \ddot{y}_2\hat{\boldsymbol{e}_2} \end{aligned} \tag{8}$$

Arm 3

$$\begin{aligned} \boldsymbol{r}_{G3/O} &= x_3\hat{\boldsymbol{e}_1} + y_3\hat{\boldsymbol{e}_2} \\ {}^{\mathcal{I}}\boldsymbol{v}_{G3/O} &= \dot{x}_3\hat{\boldsymbol{e}_1} + \dot{y}_3\hat{\boldsymbol{e}_2} \\ {}^{\mathcal{I}}\boldsymbol{a}_{G3/O} &= \ddot{x}_3\hat{\boldsymbol{e}_1} + \ddot{y}_3\hat{\boldsymbol{e}_2} \end{aligned} \tag{9}$$

1.1.6 Angular Momentum Approach

In this section, the EOMs are derived using the Angular Momentum Balance (AMB). The angular momentum balance of a system of n rigid bodies about an arbitrary point C with m external forces is defined as the following:

$$\sum_{j=1}^{m}(\boldsymbol{r}_{j/C} \times \boldsymbol{F}_j^{ext}) = \sum_{i=1}^{n}(I_{Gi}\ddot{\theta}_i\hat{e}_3 + \boldsymbol{r}_{Gi/C} \times m_{Gi}{}^{\mathcal{I}}\boldsymbol{a}_{Gi/O}) \tag{10}$$

To solve the triple pendulum, Equation 10 is used on three systems:

1. The entire pendulum, with $C = O$
2. The two links of the pendulum between P_2 and P_3, with $C = P_2$
3. The last link of the pendulum, which is from P_3 to P_4, with $C = P_3$

Taking the AMB about the selected points eliminates the constraint forces from the equations of motion. To get scalar EOMs from the resulting vector equations, the three equations are each dotted with $\hat{e}_3$. The results of this operation are the full equations of motion of the system. While the equations of motion could theoretically be solved for by hand, the process is incredibly tedious and becomes practically impossible when more links are added to the pendulum. Thus, the symbolic toolbox in MATLAB is used to solve for the EOMs.

The code that was used to solve for the EOMs using AMB can be found in A.3 and A.4. The AMB equations are transformed into matrices using MATLAB's `equationsToMatrix` function, and the resulting A matrix and b vector are then transformed into MATLAB functions by the `matlabFunction` command. This exact approach is used by the other two methods to convert the EOMs into MATLAB functions.

1.1.7 Differential Algebraic Equations Approach

A second method for finding the EOMs of the system is the Differential Algebraic Equations (DAE) approach. This approach is the "brute force" method of solving dynamics problems, breaking the system down into its smallest parts and explicitly including constraint forces in the equations of motion. This is done first by performing an Angular Momentum Balance for each link, with $C = G$, eliminating the third term in Equation 10. Next, a Linear Momentum Balance (LMB) is performed on each link. LMBs that were performed are defined in Equation 11. The resulting vector equations are dotted in each of the Cartesian directions ($\hat{e_1}$ and $\hat{e_2}$) to give two scalar equations per link.

$$\begin{aligned}
\sum \boldsymbol{F} &= {m_1}^{\mathcal{I}}\boldsymbol{a}_{G1/O} = (F_{x1} + F_{x2} + m_1 g)\hat{\boldsymbol{e_1}} + (F_{y1} + F_{y2})\hat{\boldsymbol{e_2}} \\
\sum \boldsymbol{F} &= {m_2}^{\mathcal{I}}\boldsymbol{a}_{G2/O} = (-F_{x2} + F_{x3} + m_2 g)\hat{\boldsymbol{e_1}} + (-F_{y2} + F_{y3})\hat{\boldsymbol{e_2}} \\
\sum \boldsymbol{F} &= {m_3}^{\mathcal{I}}\boldsymbol{a}_{G3/O} = (-F_{x3} + m_3 g)\hat{\boldsymbol{e_1}} - F_{y3}\hat{\boldsymbol{e_2}}
\end{aligned} \tag{11}$$

Finally, constraint equations are included in the DAE system of equations. The constraints that were used set the Cartesian and polar accelerations of each center of mass equal to each other, and dot the resulting vector relations with $\hat{e_1}$ and $\hat{e_2}$ to obtain two scalar equations for each link.

Using the DAE approach, the equations of motion are a system of $5n$ equations, where n is the number of links of the pendulum. At each time step of the numerical integration to solve the EOMs, the values of the zeroth and first derivatives are plugged into the system of equations and the second derivatives are solved for using MATLAB's backslash command. DAEs are impossible to solve analytically by hand. The code used to find the DAEs for the triple pendulum can be found in A.3 and A.5.

1.1.8 Lagrange Approach

A third way of finding the EOMs of the triple pendulum is to use Lagrange's equations, which are defined in Equation 12.

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta_i}}\right) - \frac{\partial L}{\partial \theta_i} = 0 \tag{12}$$

There are n Lagrange equations for a pendulum with n links, where each θ_i and $\dot{\theta}_i$ in Equation 12 correspond to the angles from the vertical made by each link. The L in Equation 12 refers to the Lagrangian, which is defined in Equation 13. T refers to the kinetic energy of the system, and V refers to the potential energy of the system. T decomposes into rotational and translational kinetic energy, which are the first two terms on the right hand side of of Equation 13, and V is simply the

gravitational potential energy, which is the rightmost term.

$$L = T - V = T_{rot} + T_{trans} - V = \frac{1}{2}\sum_{k=1}^{n}(I_k\omega_k^2 + m_k v_k^2 - 2m_k g y_k) \tag{13}$$

Since the Lagrange method relies on calculations involving energy, the results of calculating each of 12 are already a scalar system of equations. The resulting equations are the EOMs of the system, and match exactly with the EOMs from the AMB if simplified properly. However, like the AMB approach, the calculation of the EOMs from the Lagrange method is best done on the computer using MATLAB's symbolic toolbox. The results of this method are that the MATLAB functions generated by the AMB and Lagrange approach are not exactly the same, although analytically they should be identical. This causes the solutions of numerical integration to eventually diverge. The MATLAB code to find the EOMs of the triple pendulum using Lagrange's method can be found in A.3 and A.6.

1.2 MATLAB Results for Test Set 1

The first test case for the triple pendulum was to give it the following parameters and initial conditions:

- All $m = 1$
- All $L = 1$
- $g = 10$
- All $\theta_{init} = \pi/2$
- All $\dot{\theta}_{init} = 0$

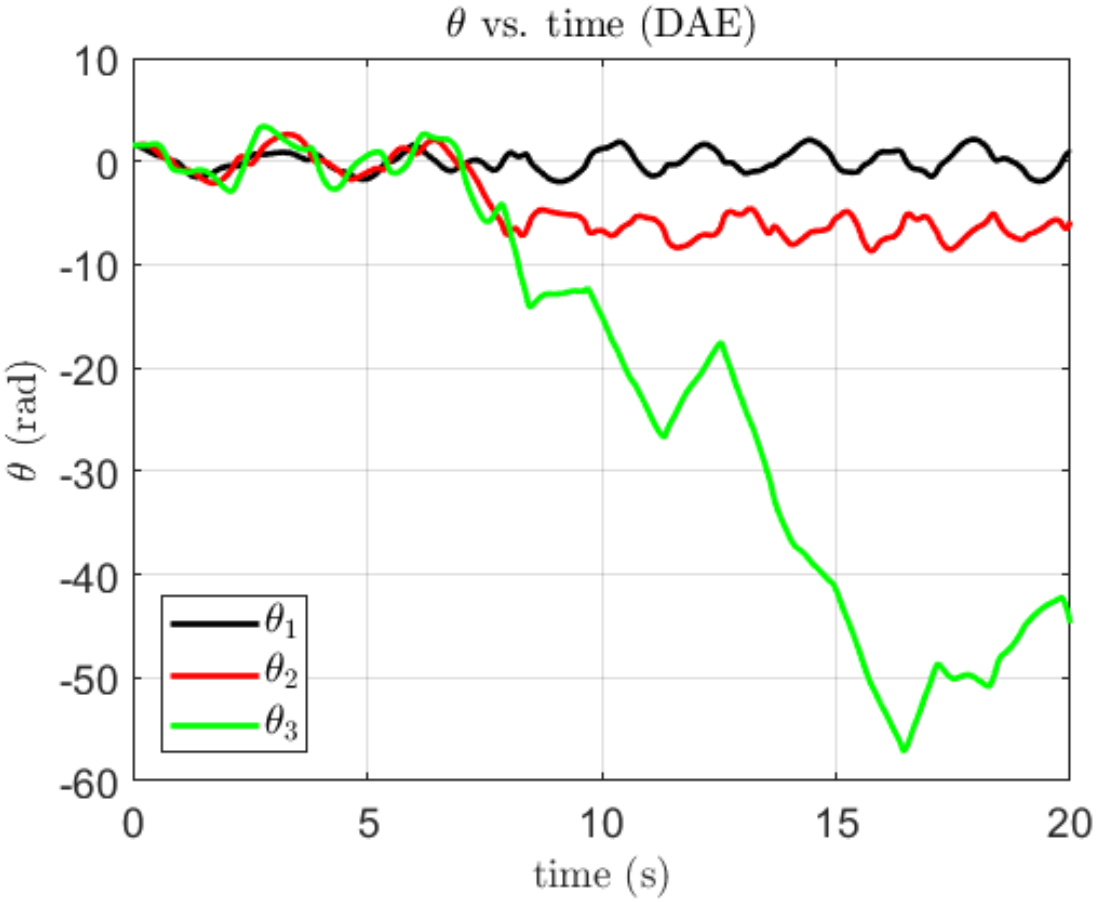

Figure 4: Output of DAE method for Set 1 of the test parameters and initial conditions.

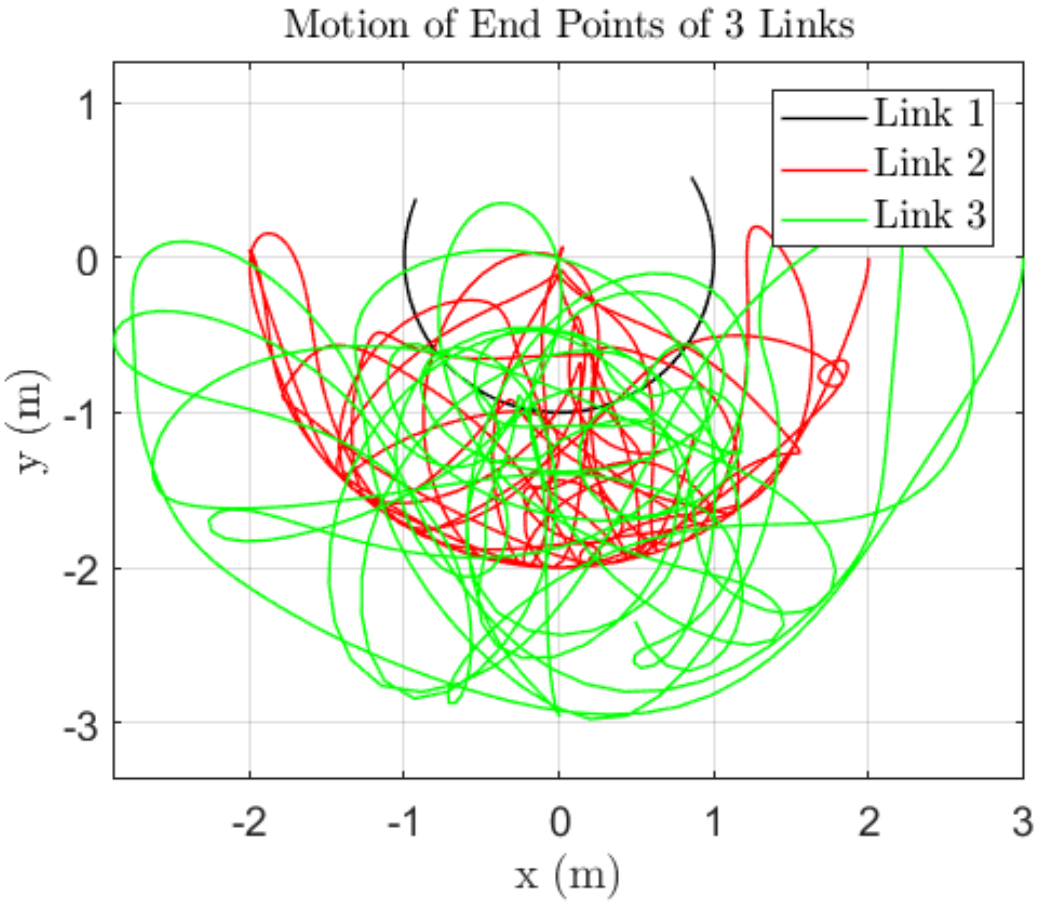

Figure 5: The path traced by the end of each pendulum link is shown here. This particular solution was generated by the AMB, but the other solutions look identical to this one.

The angular output of the DAE method can be seen in Figure 4. The angular output of this method so closely matches with that of Lagrange and the Angular Momentum Balance that the differences between Figure 4 and the corresponding plots generated by AMB and Lagrange look identical. However, when using numerical methods, there is always error inherent in the solution. Thus, the plots shown in Figure 6 were generated to make sense of the errors that were taking place.

The path traced by the end of each link of the pendulum is shown in Figure 5. It is clear that this is a chaotic system, since the pendulum never traces out the same path twice for this initial condition.

1.2.1 Analysis of Error

As can be seen in Figure 6, the error between the θ outputs for each method seem to match very closely until the error suddenly seems to blow up. However, when the integration was stopped just before the error seemed to rapidly increase, the same plot shape emerged, just with a smaller scale on the vertical axis. This can be seen in Figure 7. The conclusion that can be drawn from

this finding is that the error between the different methods grows at a roughly exponential rate. The roughly linear trend in Figure 8, which plots the log of error of θ_1 vs. time confirms that the error does in fact grow exponentially.

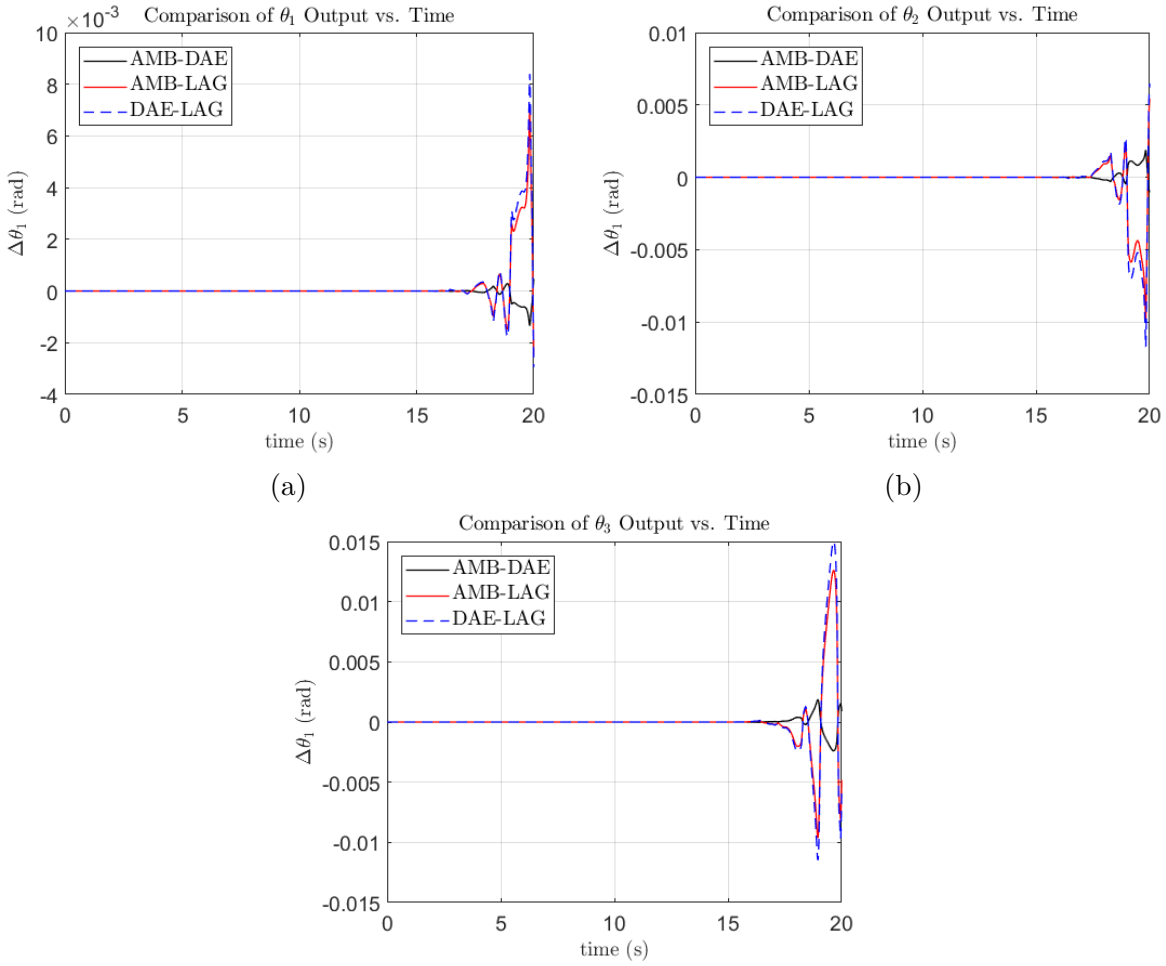

Figure 6: 6a, 6b, and 6c compare the angular output of the three different methods for θ_1, θ_2, and θ_3 respectively. The error seems to be zero until it suddenly blows up around 16-17 seconds, but this turns out to be a misleading plot, as Figure 7 demonstrates.

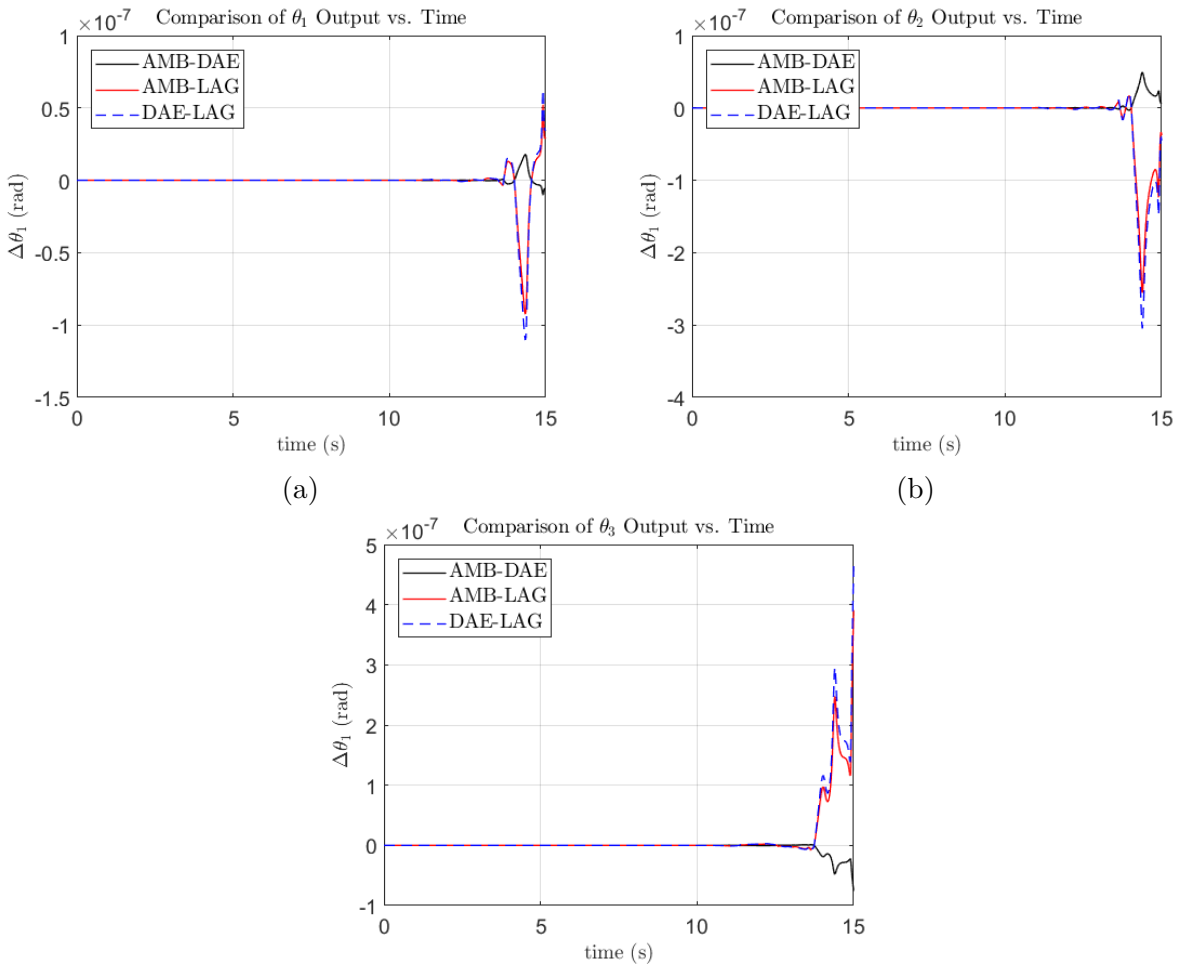

Figure 7: 7a, 7b, and 7c compare the angular output of the three different methods for θ_1, θ_2, and θ_3 respectively, but for a shorter integration time than in Figure 6. As can be seen in this figure, the error still seems to suddenly get much larger just before the end of integration. However, the time scale on these plots is 10^{-7}, whereas the time scale on the previous figure's plots is 10^{-3}. This is indicative of an exponentially growing error.

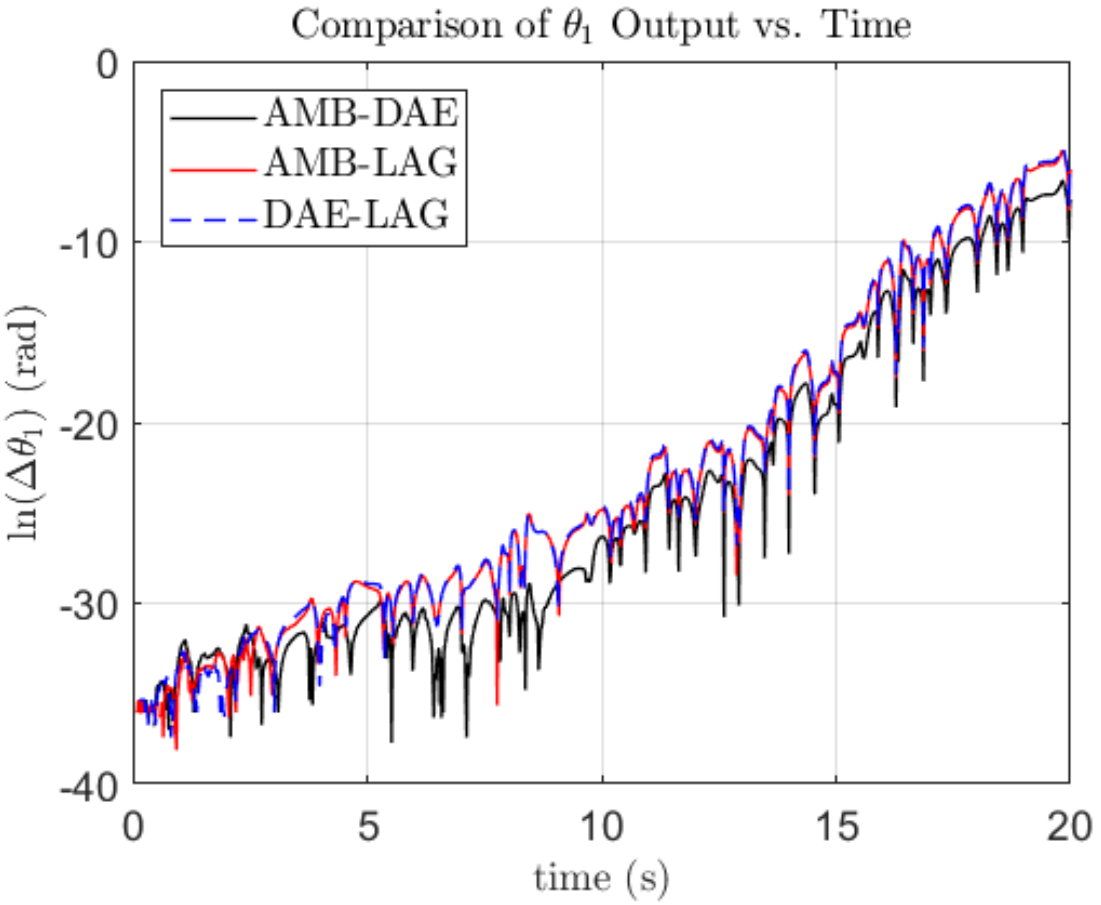

Figure 8: A plot of the natural log of error in θ_1 over time. The linear trend of this plot proves that the error between solution methods for the triple pendulum grows roughly exponentially.

1.2.2 Validation of Results

While the nature of the error between the solutions has been characterized, the accuracy of the solutions of any of the methods has not yet proven. The method that will be used to check the validity of the solutions is conservation of energy. Since total energy has to be conserved in both the triple pendulum and the four bar linkage, a simple way to check for accuracy is to calculate the energy based on the `ode45` output and see if it remains constant. Using a similar method of calculating energy as for the Lagrange Method, except for data vectors instead of symbolic variables, the kinetic and potential energies are calculated. Instead of subtracting them, however, the kinetic and potential energies of all the links are summed at each time step. The result of this calculation is shown in Figure 9. As seen in the figure, energy is conserved down to the nanojoule. This proves that the solutions resulting from the methods described in the previous sections are both valid (within numerical error) and implemented correctly.

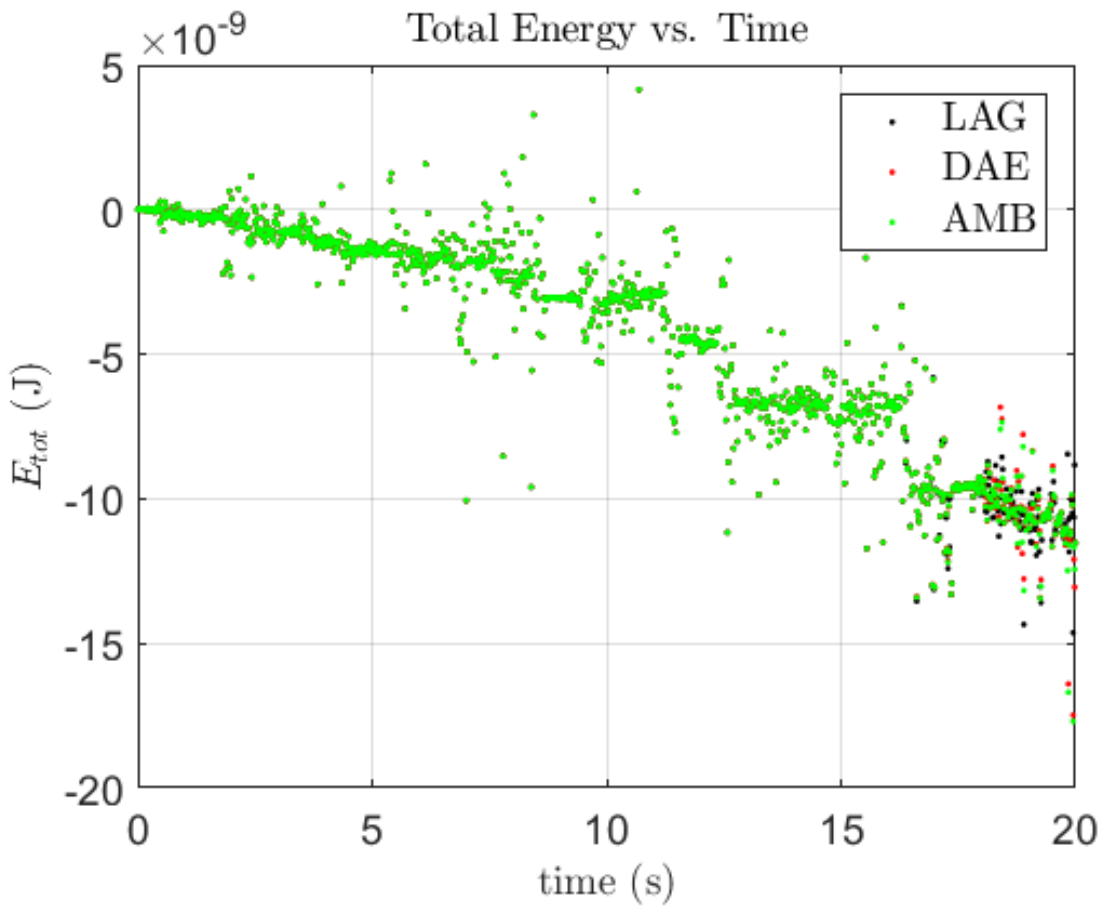

Figure 9: The free body diagrams showing all of the external forces acting on each link of the pendulum.

1.3 MATLAB Results for Test Set 2

A second test case was used to test the triple pendulum solutions. The parameters and initial conditions of this test case are described below:

- All $m = 1$
- All $L = 1$
- $g = 0$
- All $\theta_{init} = \pi/2$
- All $\dot{\theta}_{init} = 1$

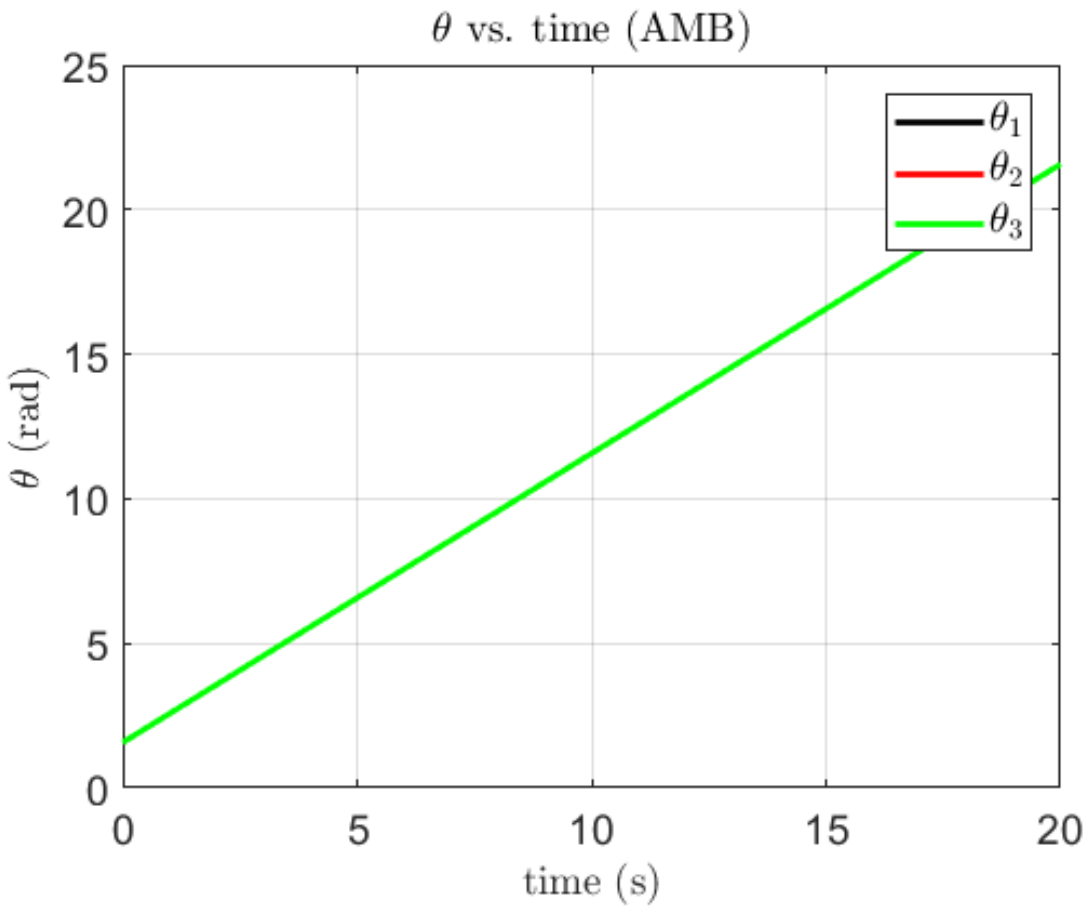

Figure 10: Angular output of the AMB method for Set 2 of the test parameters and initial conditions. Without the influence of gravity, the angles all keeps changing at their initial angular velocities, which are all 1 rad/s.

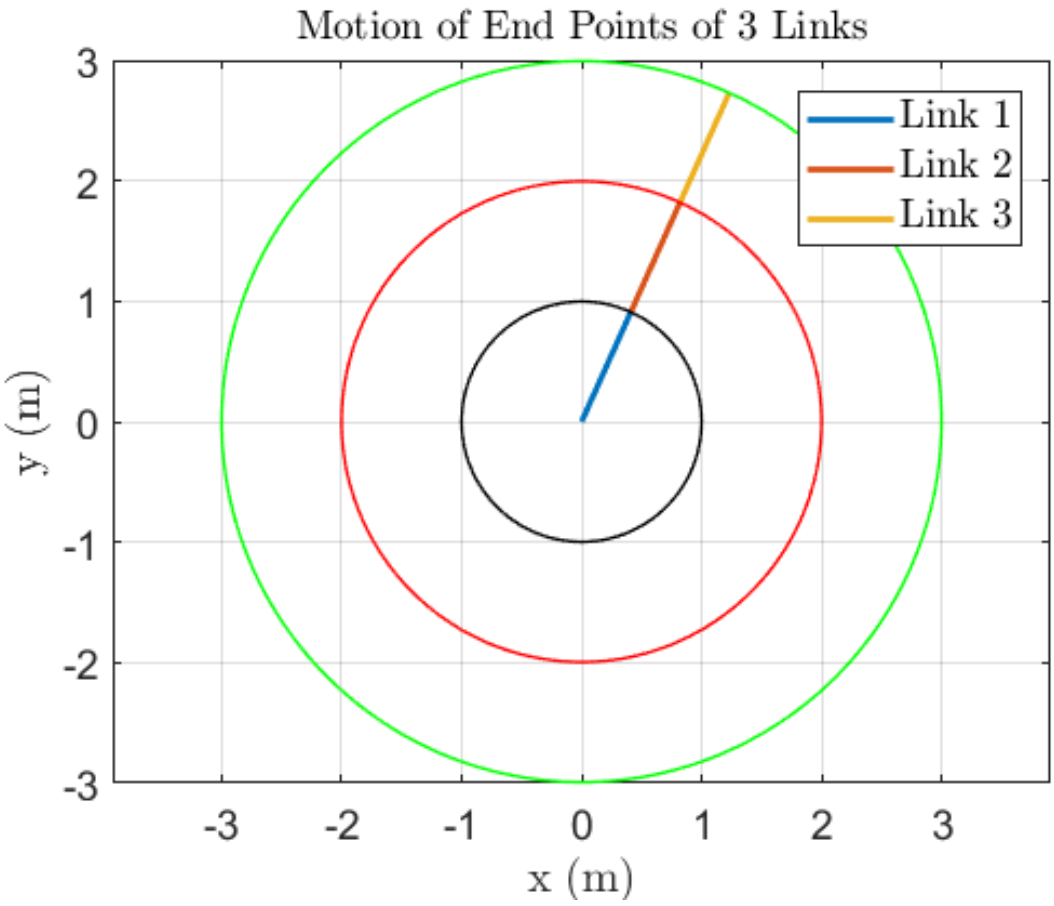

Figure 11: The path traced by the end of each pendulum link is shown here. This particular solution was generated by the Lagrange method, although the other two methods generated identical solutions.

The results of this test case can be easily predicted with intuition. Without the influence of gravity, there are no external forces acting on the triple pendulum. Thus, if every single link of the pendulum is given the same angular velocity, one would expect the pendulum to swing in a circle in the same manner as if it were a single rigid rod. The angle would linearly increase and the path traced out by the endpoint of each link would be a circle. This behavior was confirmed by the output of the numerical integration, as can be seen in Figures 10 and 11.

1.3.1 Analysis of Error

As can be seen in Figure 12, the error in the solution seems to grow linearly with time, which is much slower than for the gravitational case over long integration times. Since the order of magnitude of the error at t=20s is 10^{-10}, this shows that the case where gravity is set to zero results in a significantly more accurate solution than for Test Set 1.

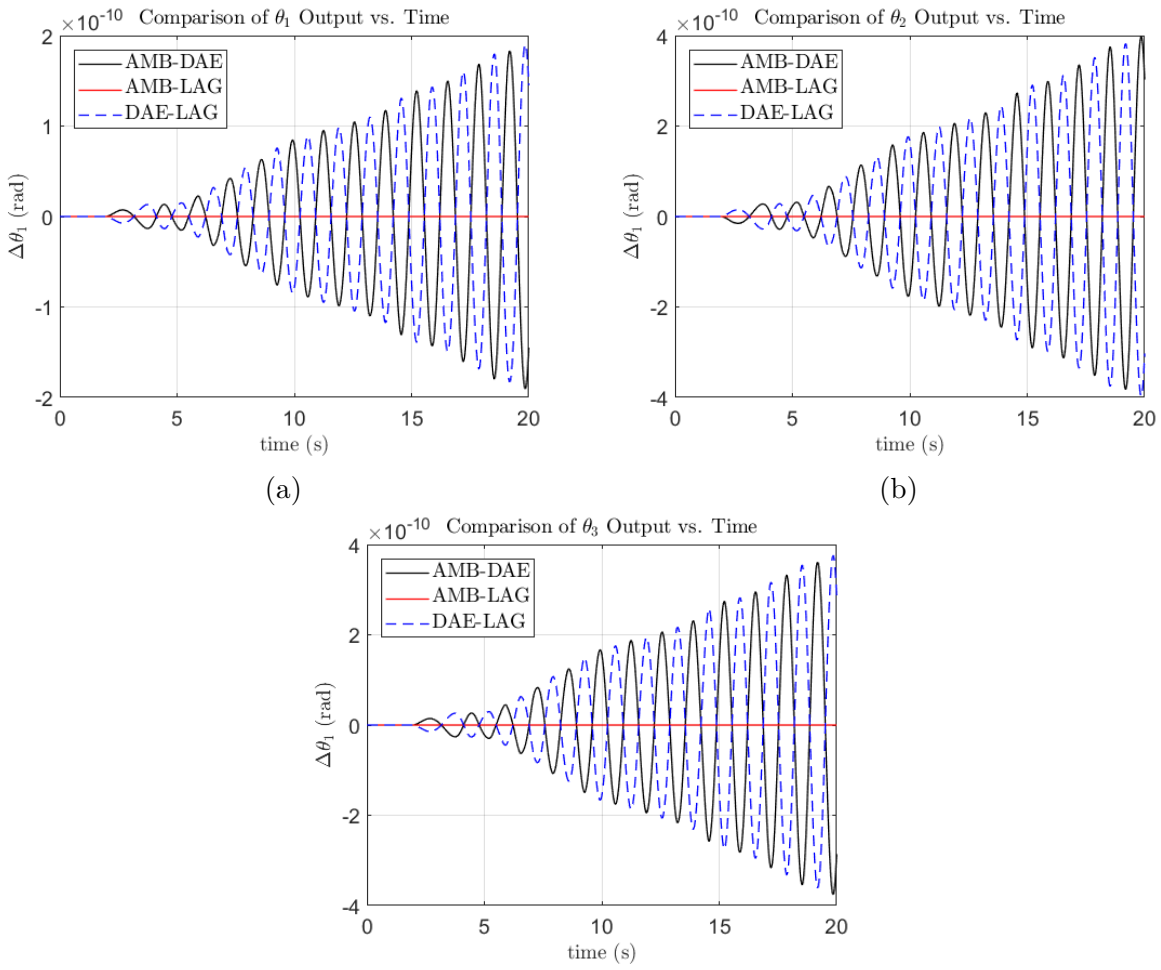

Figure 12: 12a, 12b, and 12c compare the angular output of the three different methods for θ_1, θ_2, and θ_3 respectively. Unlike for the case were $g \neq 0$, this error appears to grow linearly with time.

1.3.2 Validation of Results

As with Set 1, the validity of the results for Set 2 will be checked with conservation of energy. As can be seen in Figure 13, energy is conserved to within 10^{-13} Joules. Energy is conserved even more than from Test Set 1, and thus it is proven that the AMB, DAE, and Lagrange methods have been implemented correctly and give accurate solutions.

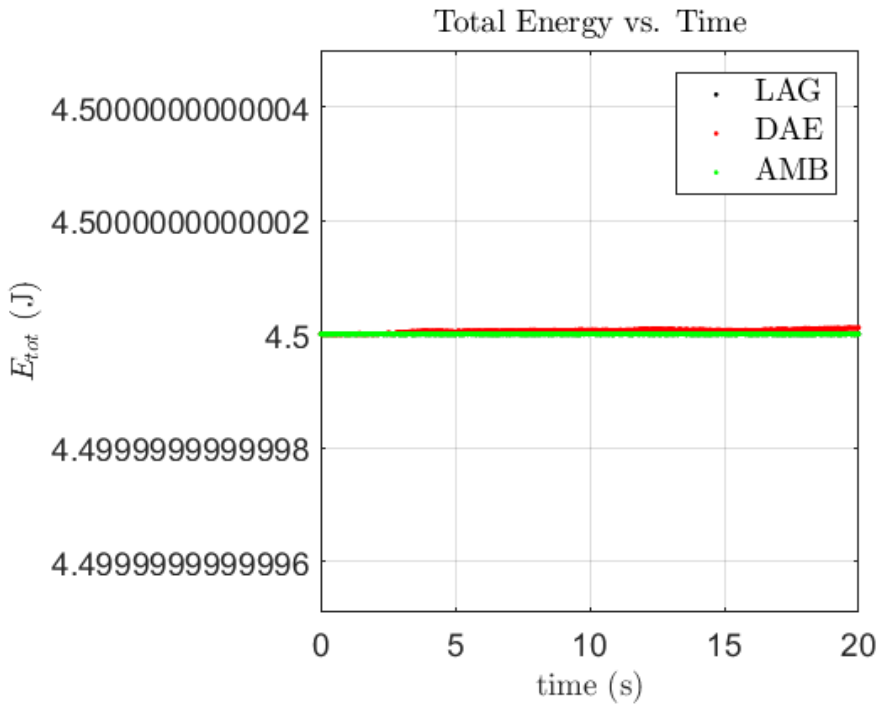

Figure 13: The total energy of the three link pendulum under Test Set 2, where gravity is turned off and each link is given the same initial angular velocity of 1 rad/s. This plot demonstrates that energy is conserved throughout this solution, and thus the solution is valid.

2 The Four Bar Linkage

2.1 Model

The four bar linkage is a special case of the triple pendulum in which the end of the pendulum is pinned to an inertial point. Because of this, all of the information from sections 1.1.2, 1.1.4, and 1.1.5 still applies to this problem.

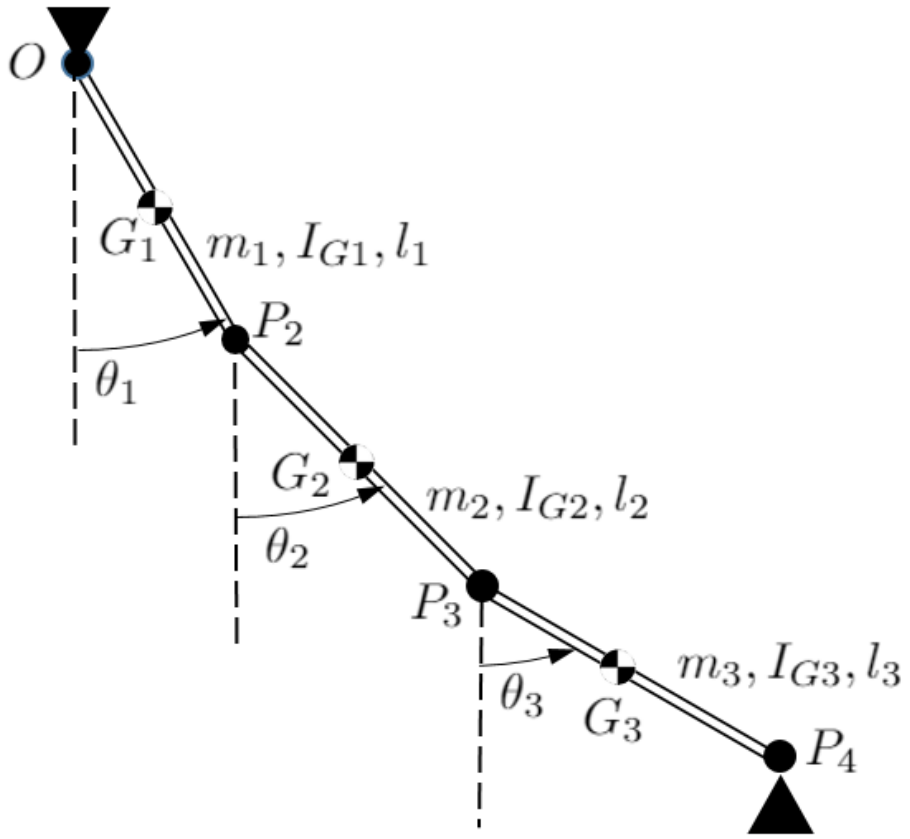

Figure 14: Model of the four bar linkage.

2.2 Free Body Diagrams

The free body diagram for the four bar linkage is identical to that of the triple pendulum except for the added constraint forces at P_4. This similarity means that the DAE solution of the four bar linkage is almost exactly the same as that of the triple pendulum.

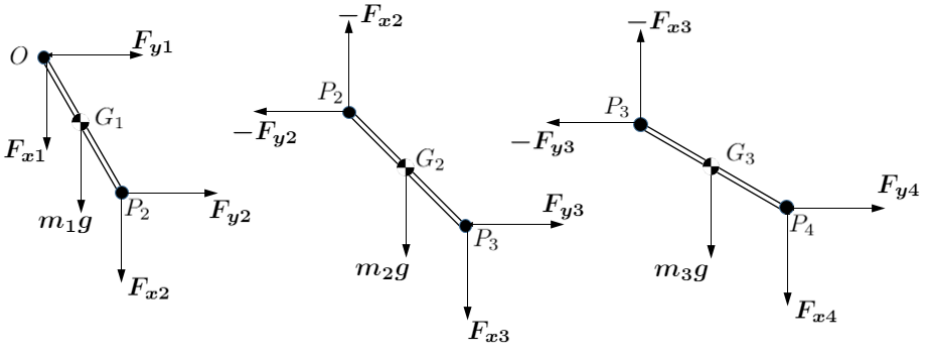

Figure 15: The free body diagrams showing all of the external forces acting on each link of the four bar linkage.

2.3 Differential Algebraic Equations Approach

The method for solving the four bar linkage using DAEs is nearly the same as the method to solve the triple pendulum. The only differences are that that there are two more constraint forces (one in each Cartesian direction) to account for in the LMB and AMB for the last link (P_3-P_4), and that there are two more constraint equations to add to the DAE system.

Accounting for the additional constraint forces means adding two more cross products to the left side of Equation 10 for the AMB about G_3 and also adding $F_{x4}\hat{\boldsymbol{e_1}} + F_{y4}\hat{\boldsymbol{e_2}}$ to the right side of the last equation in 11. The constraint equation for P_4 is derived from the fact that the position of the point relative to the origin has to be constant. This derivation is shown below in 14. The code that calculates the equations of motion of the four bar linkage can be found in B.4 and B.5.

$$\begin{aligned} \boldsymbol{r}_{P4/O} &= constant \\ {}^{\mathcal{I}}\boldsymbol{v}_{P4/O} &= 0 \\ {}^{\mathcal{I}}\boldsymbol{a}_{P4/O} &= 0 \end{aligned} \tag{14}$$

2.4 Selection of Initial Angular Velocities

It is important to note that because the constraint of zero acceleration of P_4 does not necessarily pin the point inertially, suitable initial conditions have to be chosen in order to avoid violating the intended constraint. While any initial θ_i can be chosen without violating the constraint, only one $\dot{\theta}_i$ can be arbitrarily chosen, since the four bar linkage is a one degree-of-freedom (DOF) system. The following derivation demonstrates how to solve for $\dot{\theta}_2$ and $\dot{\theta}_3$ given an arbitrarily chosen θ_1, θ_2, θ_3, and $\dot{\theta}_1$. One can apply the method to solve for any two of the three angular velocities given one arbitrarily chosen one.

The first step is to recall that P_4 must have zero velocity.

$${}^{\mathcal{I}}\boldsymbol{v_{P4/O}} = \boldsymbol{0} \tag{15}$$

This can be expanded as follows:

$${}^{\mathcal{I}}\boldsymbol{v}_{P2/P0} + {}^{\mathcal{I}}\boldsymbol{v}_{P3/P2} + {}^{\mathcal{I}}\boldsymbol{v}_{P4/P3} = \boldsymbol{0} \tag{16}$$

$$l_1\dot{\theta}_1\hat{\boldsymbol{a_2}} + l_2\dot{\theta}_2\hat{\boldsymbol{b_2}} + l_3\dot{\theta}_3\hat{\boldsymbol{c_2}} = \boldsymbol{0} \tag{17}$$

Using the information from Table 1 to convert from body frames to the inertial frame, and ignoring the third element of each vector (which is zero), the following system of equations is obtained:

$$l_1\dot{\theta}_1 \begin{bmatrix} -\sin(\theta_1) \\ \cos(\theta_1) \end{bmatrix} + l_2\dot{\theta}_2 \begin{bmatrix} -\sin(\theta_2) \\ \cos(\theta_2) \end{bmatrix} + l_3\dot{\theta}_3 \begin{bmatrix} -\sin(\theta_3) \\ \cos(\theta_3) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{18}$$

The only two unknowns in Equation 18 are the two unknown angular velocities, which are easily solved for in MATLAB. The code that performs the above calculation can be found in B.2.

2.5 MATLAB Results for Test Set 3

The test set used for the four bar linkage had the following initial conditions and parameters:

- $l_1 = 2,\ l_2 = 1,\ l_3 = 0.5$
- $m_1 = 1,\ m_2 = 1,\ m_3 = 3$
- g $= 5$
- $\theta_1 = \pi - \frac{1}{64},\ \theta_2 = \pi/2,\ \theta_3 = 0$
- $\dot{\theta}_3 = 2$

Under these given initial conditions, the other two angular velocities were calculated to be $\dot{\theta}_1 = 0.5001$ and $\dot{\theta}_2 = -0.0156$. Figures 16, 17, and 18 show the results of the numerical integration. This motion was highly interesting, with the two longer links driving the pinned third link around in a full circular pattern.

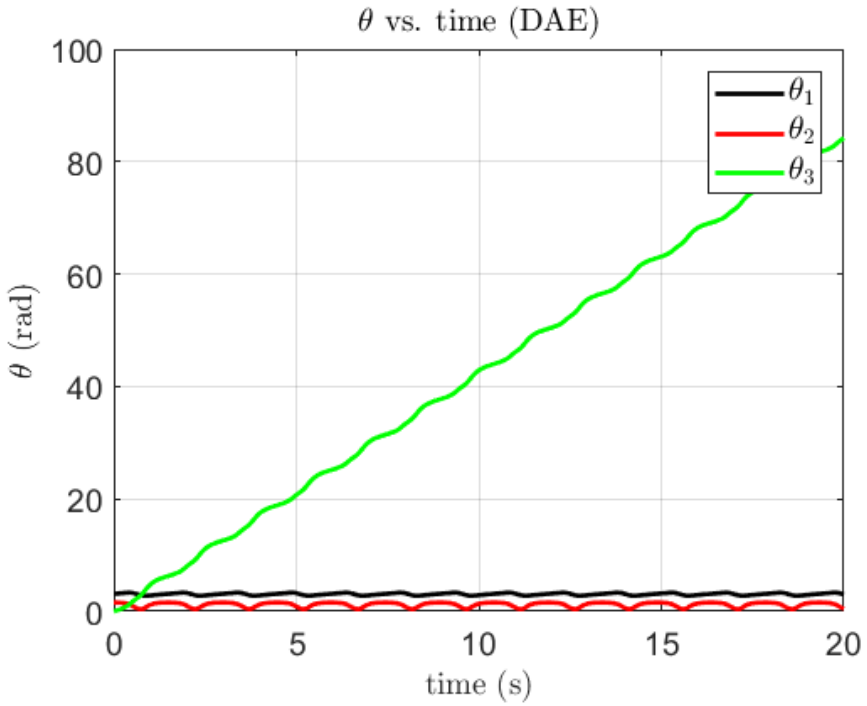

Figure 16: The angular output of the DAE solver for the four bar linkage. θ_1 and θ_2 oscillate back and forth while the third link keeps spinning in a circle.

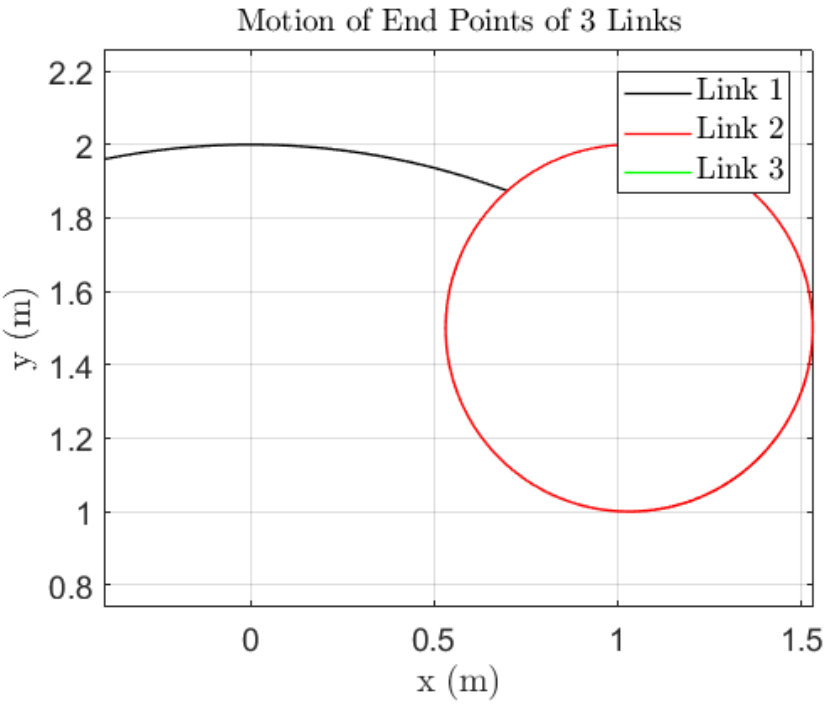

Figure 17: The path of the ends of the links of the four bar linkage. Since the constraint was not violated during the setting of the initial conditions, the motion of the end of link 3 that is P_4 is not visible. While it does move because of numerical inaccuracies, its motion is negligible compared to the motions of P_2 and P_3.

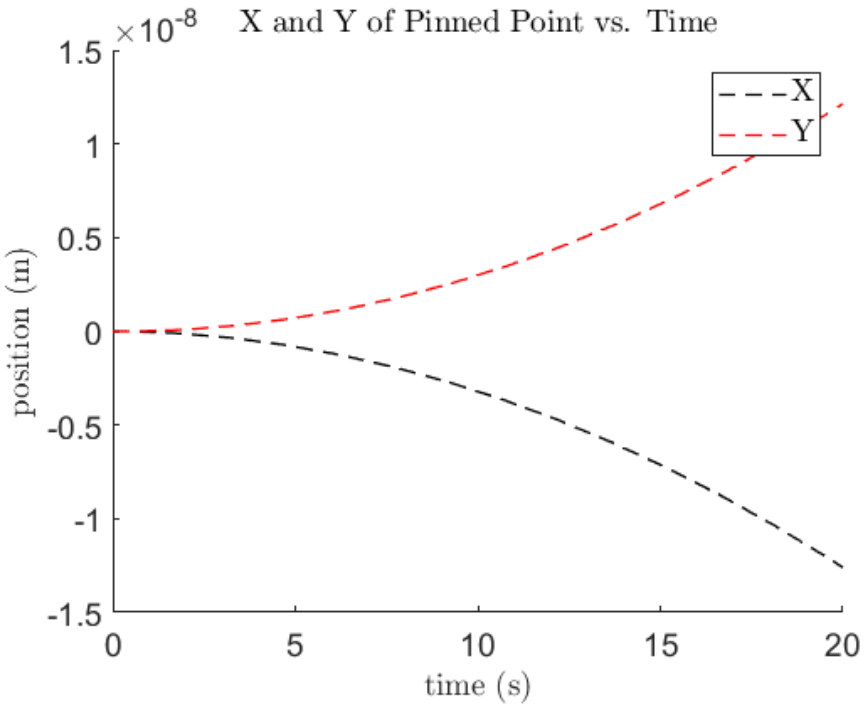

Figure 18: This plot of the movement of the X and Y coordinates of the pinned point over time demonstrates that the constraint is satisfied. Movement on the order of 10^{-8} meters is, for all intents and purposes, zero.

2.5.1 Validation of Results

The same energy conservation check as was applied to the triple pendulum can be used for the four bar linkage. Figure 19 shows that while the total energy of the system trends away from its initial level much faster than with the triple pendulum, it is still conserved, since it only deviates by an order of magnitude of 10^{-6} Joules. The quick deviation from the original energy level is likely a result of the numerical error inherent when the third link jerks roughly around P_4 and large accelerations are present.

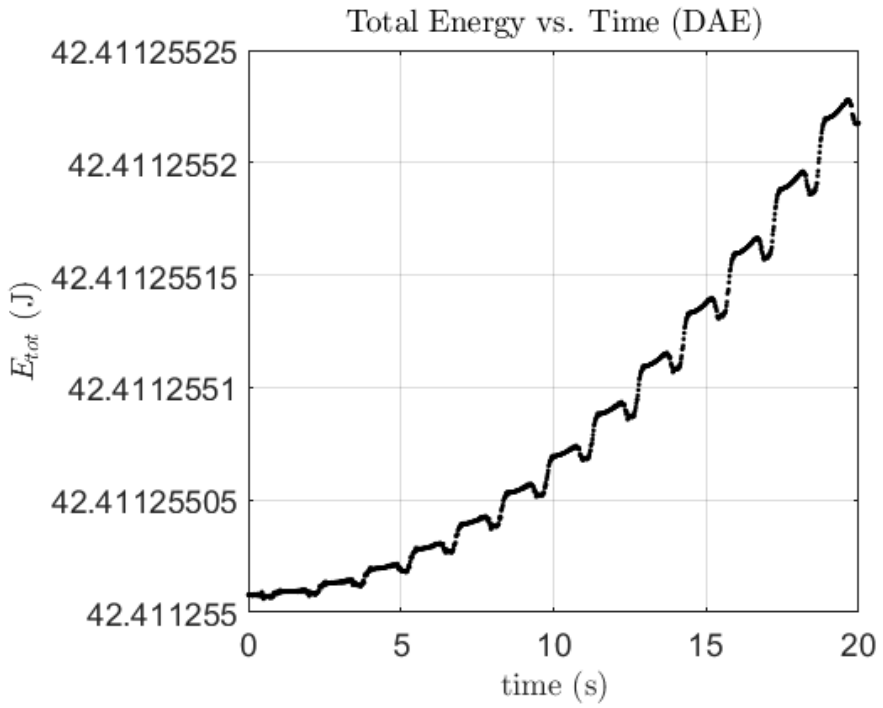

Figure 19: Plot of total energy of the four bar linkage vs. time

3 The N-Link Pendulum

3.1 Lagrange Approach

3.2 MATLAB Results

To test the n-link code, the following parameters and initial conditions were used:

- n = 20
- All $m = 1$
- All $L = 1$
- g = 10
- All $\theta = \pi/2$
- All $\dot{\theta} = 0$

Under these parameters and initial conditions, the pendulum demonstrated highly chaotic motion, as can be seen in Figures 20 and 21.

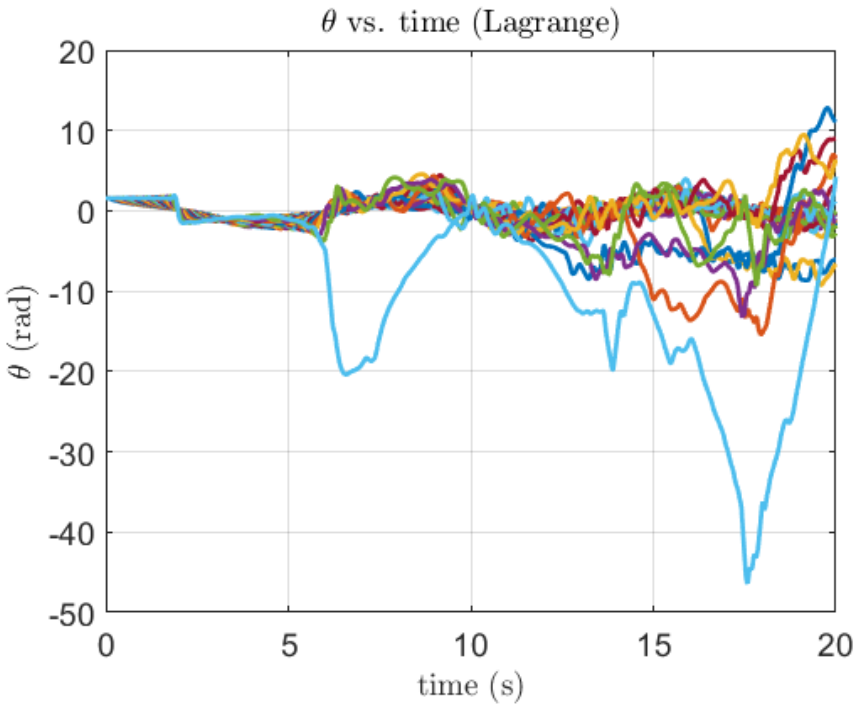

Figure 20: The angular results of the numerical integration for n = 20 with Lagrange's method.

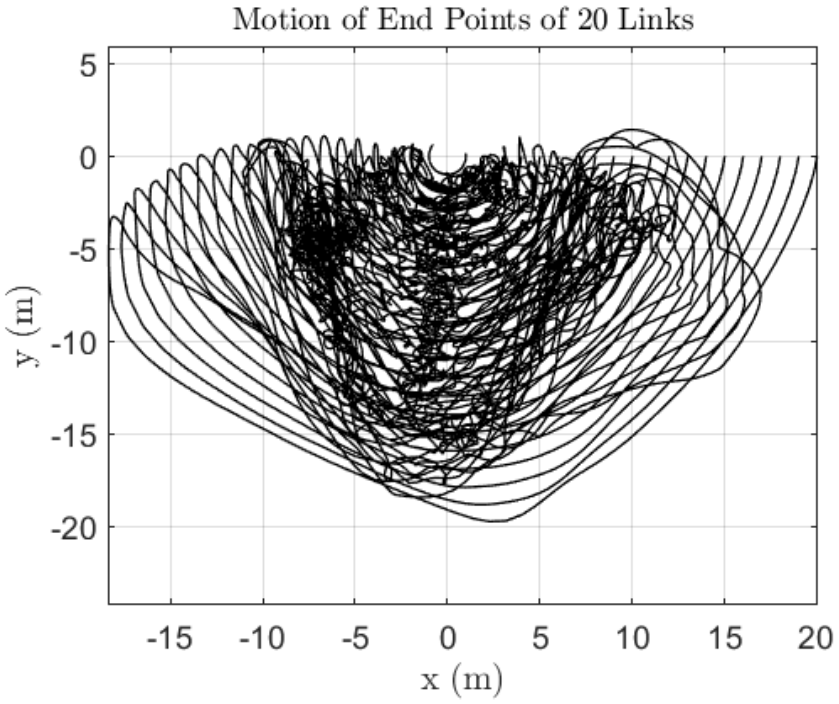

Figure 21: Path of the ends of each link of the 20-link pendulum as it is released from rest at a horizontal orientation.

3.2.1 Validation of Results

Similar to every other system outlined in this report, the n-link pendulum is also subject to the conservation of energy. Thus, the accuracy of the solution can be judged based on how close the system's energy stays to the initial energy. Figure 22 shows that the system's energy is conserved. Thus, the 20-link pendulum was successfully solved.

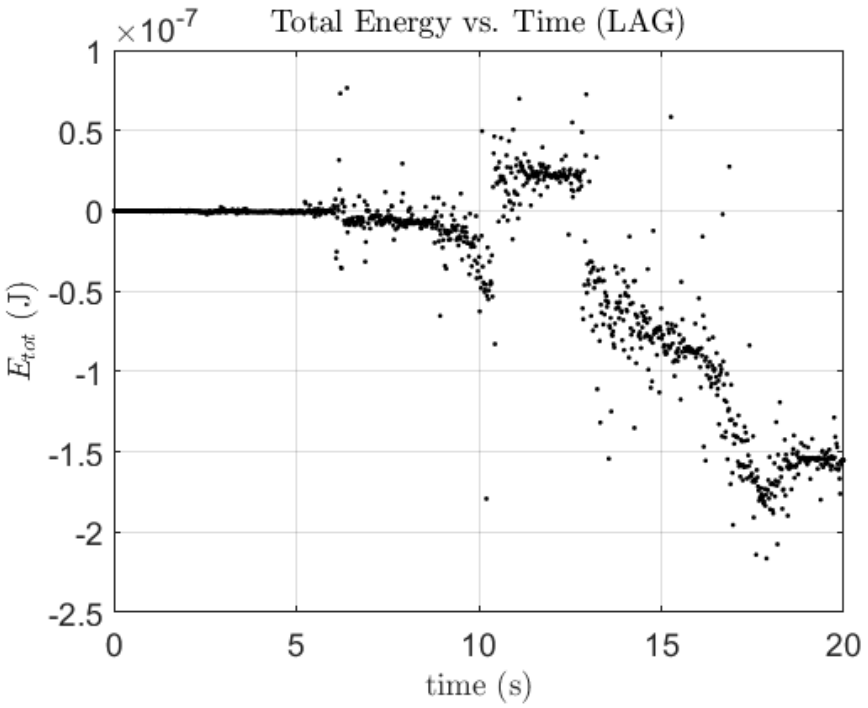

Figure 22: Plot of total energy of the n-link pendulum vs. time

A Triple Pendulum Code

A.1 Set Parameters and Initial Conditions

Listing 1: md697_triple_pendulum_DEMO.m

```
%% Mitchell Dominguez - md697 - MAE 4730 - FINAL PROJECT -
   PRESENTATION
clear;
close all;
set(0,'defaulttextinterpreter','latex');
%% TRIPLE PENDULUM DEMO 1
% Set parameters
```

```
p.L = [1 1 1]; % lengths
p.m = [1 1 1]; % masses
p.g = 1; % acceleration due to gravity
p.n = 3; % number of links
p.I_G = (p.m .*p.L.^2)/12;

% Initial Conditions
t = 50; % Time to integrate for

% init_ths = [deg2rad(70); deg2rad(40); deg2rad(160)];
init_ths = [pi/2;pi/2;pi/2];
init_thDots = [0;0;0];

z0 = [init_ths;init_thDots]; % Initial conditions

% Output settings
rederive = 0;
plots = 1;
animate = 1;

% Run triple pendulum
md697_run_triple_pendulum(p,t,z0,rederive,plots,animate)
```

A.2 Calculate Kinematics

Listing 2: md697_triple_pendulum_kinematics.m

```
%% Mitchell Dominguez - md697 - MAE 4730 - FINAL PROJECT - Triple
   Pendulum Kinematics
% This script works with md697_triple_pendulum_deriver.m

% G1/O
rG1o = (L(1)/2)*f1(:,1);
vG1o = jacobian(rG1o,th(1))*thDot(1);
aG1o = jacobian(vG1o,[th(1) thDot(1)])*[thDot(1) thDDot(1)].';

% P2/O
rP2o = 2*rG1o;
vP2o = 2*vG1o;
aP2o = 2*aG1o;
```

```
% G2/P2
rG2p2 = (L(2)/2)*f1(:,2);
vG2p2 = jacobian(rG2p2,th(2))*thDot(2);
aG2p2 = jacobian(vG2p2,[th(2) thDot(2)])*[thDot(2) thDDot(2)].';

% G2/O
rG2o = rP2o + rG2p2;
vG2o = vP2o + vG2p2;
aG2o = aP2o + aG2p2;

% P3/P2
rP3p2 = 2*rG2p2;
vP3p2 = 2*vG2p2;
aP3p2 = 2*aG2p2;

% P3/O
rP3o = rP2o + rP3p2;
vP3o = vP2o + vP3p2;
aP3o = aP2o + aP3p2;

% G3/P3
rG3p3 = (L(3)/2)*f1(:,3);
vG3p3 = jacobian(rG3p3,th(3))*thDot(3);
aG3p3 = jacobian(vG3p3,[th(3) thDot(3)])*[thDot(3) thDDot(3)].';

% G3/O
rG3o = rP3o + rG3p3;
vG3o = vP3o + vG3p3;
aG3o = aP3o + aG3p3;

% G3/P2
rG3p2 = rP3p2 + rG3p3;
vG3p2 = vP3p2 + vG3p3;
aG3p2 = aP3p2 + aG3p3;
```

A.3 Derive Equations of Motion

Listing 3: md697_triple_pendulum_deriver.m

```
%% Mitchell Dominguez - md697 - MAE 4730 - FINAL PROJECT - Triple
   Pendulum Deriver
% close all;
% clear;
function md697_triple_pendulum_deriver
%% Set up symbolic scalars
n = 3; % Number of pendulum links
syms g; % acceleration due to gravity
L = sym('L', [1 n], 'positive'); % array of link lengths
m = sym('m', [1 n], 'positive'); % array of link masses
th = sym('th', [1 n], 'real'); % array of thetas
thDot = sym('thD', [1 n], 'real'); % array of theta dots
thDDot = sym('thDD', [1 n], 'real'); % array of theta double dots
xDDot = sym('xDDot', [1 n], 'real'); % x double dot of each COM
yDDot = sym('yDDot', [1 n], 'real'); % y double dot of each COM
Fx = sym('Fx', [1 n], 'real'); % Constraint forces in x direction
Fy = sym('Fy', [1 n], 'real'); % Constraint forces in y direction
I_G = (m .*L.^2)/12; % Moments of inertia about centers of mass

%% Set up symbolic vectors
f1 = [cos(th);sin(th);zeros(1,n)]; % lambda vectors for each body
   frame
f2 = [-sin(th);cos(th);zeros(1,n)]; % normal vectors for each body
   frame
e1 = [1;0;0]; % e1
e2 = [0;1;0]; % e2
e3 = [0;0;1]; % e3 (same for each frame)

iWf = [zeros(2,n);thDot]; % I omega (frame)
iAf = [zeros(2,n);thDDot]; % I alpha (frame)

%% Kinematics
md697_triple_pendulum_kinematics % Calculate kinematics
md697_n_pend_sym_kinematics % Calculate kinematics (n link)

%% Simplifying terms
mg = e1*m.*g; % matrix of gravity force terms
```

```
IA = sym(zeros(3,n));
IA(3,:) = thDDot.*I_G; % matrix of moment of inertia * I alpha (frame)
    terms

ma = (diag(m) * [aG1o,aG2o,aG3o].').'; % matrix of m*a terms

%% AMB Equations
md697_triple_pendulum_AMB_eqns

%% DAE Equations
md697_triple_pendulum_DAE_eqns

%% LAG Equations
md697_triple_pendulum_LAG_eqns

%% Generate Equations of Motion
% AMB
[AMB_A,AMB_b] = equationsToMatrix(AMB_eqns,thDDot);
AMB_A = simplify(AMB_A);
AMB_b = simplify(AMB_b);

% DAE
[DAE_A,DAE_b] = equationsToMatrix(DAE_eqns,[xDDot, yDDot, thDDot, Fx,
   Fy]);
DAE_A = simplify(DAE_A);
DAE_b = simplify(DAE_b);

% LAG
[LAG_A,LAG_b] = equationsToMatrix(LAG_eqns,thDDot);
LAG_A = simplify(LAG_A);
LAG_b = simplify(LAG_b);


%% Create MATLAB Function for EOM
% AMB
matlabFunction(AMB_A,'file','Amat_AMB','Vars',{L,m,g,th,thDot});
matlabFunction(AMB_b,'file','bvec_AMB','Vars',{L,m,g,th,thDot});
```

```
% DAE
matlabFunction(DAE_A,'file','Amat_DAE','Vars',{L,m,th});
matlabFunction(DAE_b,'file','bvec_DAE','Vars',{L,m,g,th,thDot});

% DAE
matlabFunction(LAG_A,'file','Amat_LAG','Vars',{L,m,th});
matlabFunction(LAG_b,'file','bvec_LAG','Vars',{L,m,g,th,thDot});
```

```
end
```

A.4 Find Angular Momentum Balance Equations

Listing 4: md697_triple_pendulum_AMB_eqns.m

```
%% Mitchell Dominguez - md697 - MAE 4730 - FINAL PROJECT
% ANGULAR MOMENTUM BALANCE EQUATIONS
% About O (P1)
Mo = cross(rG1o,mg(:,1)) + cross(rG2o,mg(:,2)) + cross(rG3o,mg(:,3));
HDo = sum(IA(:,1:n),2) + cross(rG1o,ma(:,1)) + cross(rG2o,ma(:,2)) +
   cross(rG3o,ma(:,3));
AMB_eqns{1} = simplify(Mo(3) == HDo(3));

% About P2
Mp2 = cross(rG2p2,mg(:,2)) + cross(rG3p2,mg(:,3));
HDp2 = sum(IA(:,2:n),2) + cross(rG2p2,ma(:,2)) + cross(rG3p2,ma(:,3));
AMB_eqns{2} = simplify(Mp2(3) == HDp2(3));

% About P3
Mp3 = cross(rG3p3,mg(:,3));
HDp3 = sum(IA(:,3:n),2) + cross(rG3p3,ma(:,3));
AMB_eqns{3} = simplify(Mp3(3) == HDp3(3));
```

A.5 Find Differential Algebraic Equations

Listing 5: md697_triple_pendulum_DAE_eqns.m

```
%% Mitchell Dominguez - md697 - MAE 4730 - FINAL PROJECT
% DAE EQUATIONS
% LMB link 1
DAE_eqns{1} = Fx(1) + m(1)*g + Fx(2) == m(1)*xDDot(1);
DAE_eqns{2} = Fy(1) + Fy(2) == m(2)*yDDot(1);

% LMB link 2
DAE_eqns{3} = -Fx(2) + m(2)*g + Fx(3) == m(2)*xDDot(2);
DAE_eqns{4} = -Fy(2) + Fy(3) == m(2)*yDDot(2);

% LMB link 3
DAE_eqns{5} = -Fx(3) + m(3)*g == m(3)*xDDot(3);
DAE_eqns{6} = -Fy(3) == m(3)*yDDot(3);

% AMB link 1
temp_7 = -(L(1)/2)*cross(f1(:,1),Fx(1)*e1 + Fy(1)*e2)...
    + (L(1)/2)*cross(f1(:,1),Fx(2)*e1 + Fy(2)*e2);
DAE_eqns{7} = temp_7(3) == IA(3,1);

% AMB link 2
temp_8 = -(L(2)/2)*cross(f1(:,2),-Fx(2)*e1 - Fy(2)*e2)...
    + (L(2)/2)*cross(f1(:,2),Fx(3)*e1 + Fy(3)*e2);
DAE_eqns{8} = temp_8(3) == IA(3,2);

% AMB link 3
temp_9 = -(L(3)/2)*cross(f1(:,3),-Fx(3)*e1 - Fy(3)*e2);
DAE_eqns{9} = temp_9(3) == IA(3,3);

% Constraint COM 1
DAE_eqns{10} = xDDot(1) == aG1o(1);
DAE_eqns{11} = yDDot(1) == aG1o(2);

% Constraint COM 2
DAE_eqns{12} = xDDot(2) == aG2o(1);
DAE_eqns{13} = yDDot(2) == aG2o(2);
```

```
% Constraint COM 3
DAE_eqns{14} = xDDot(3) == aG3o(1);
DAE_eqns{15} = yDDot(3) == aG3o(2);
```

A.6 Find Lagrange Equations

Listing 6: md697_triple_pendulum_LAG_eqns.m

```
%% Mitchell Dominguez - md697 - MAE 4730 - FINAL PROJECT
% LAGRANGE EQUATIONS
% Set up Lagrangian (triple pendulum specific)
% Ek = (1/2)*m(1)*dot(vG1o,vG1o) + (1/2)*I_G(1)*thDot(1)^2 ...
%     + (1/2)*m(2)*dot(vG2o,vG2o) + (1/2)*I_G(2)*thDot(2)^2 ...
%     + (1/2)*m(3)*dot(vG3o,vG3o) + (1/2)*I_G(3)*thDot(3)^2;
% Ep = -m(1)*g*rG1o(1) - m(2)*g*rG2o(1) - m(3)*g*rG3o(1);

% Ek = 0; % initialize kinetic energy
% Ep = 0; % initialize potential energy
syms Ek Ep
for i = 1:n
   Ek = Ek + 0.5*m(i)*dot(vG(:,i),vG(:,i)) + 0.5*I_G(i)*thDot(i)^2;
   Ep = Ep - m(i)*g*rG(1,i);
end

Lag = Ek-Ep;

% Equations
LAG_eqns = simplify(jacobian(Lag, th.').' ...
    -  jacobian(jacobian(Lag,thDot).',[th thDot])*[thDot thDDot].' ...
    == zeros(n,1));
```

A.7 Run Triple Pendulum

Listing 7: md697_run_triple_pendulum.m

```
%% Mitchell Dominguez - md697 - MAE 4730 - FINAL PROJECT - Run Triple
   Pendulum
% Run triple pendulum using functions derived from
% md697_triple_pendulum_deriver
```

```
%
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%% Run using md697_triple_pendulum_DEMO
   %%%%%%%%%%%%%%%%%%%%%%%
%
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function md697_run_triple_pendulum(p,t,z0,rederive,plots,animate)
%% Derive EOM
% rederive = input('Re-derive EOM? Enter 0 for no and 1 for yes\n');
if rederive ~= 0
    md697_triple_pendulum_deriver
end
% clear; % Re-use variables from kinematics so need to clear symbolic
   vars
%% Set parameters
% p.L = [1 1 1]; % lengths
% p.m = [1 1 1]; % masses
% p.g = 1; % acceleration due to gravity
% p.n = 3; % number of links
% p.I_G = (p.m .*p.L.^2)/12;
%
% % tspan = [0 100];
% t = 20;
numpts = 60*t; % 60 points per second
tspan = linspace(0,t,numpts);

%% Set initial conditions
% init_ths = [pi/2; pi/2; pi/2];
% % init_ths = [pi;pi;pi];
% init_thDots = [0;0;0];
%
% z0 = [init_ths;init_thDots];


%% Integrate
```

```
% AMB
options_AMB = odeset('AbsTol',1e-10,'RelTol',1e-10);
[t_AMB,z_AMB] = ode45(@RHS_AMB,tspan,z0,options_AMB,p);
ths_AMB = z_AMB(:,1:3);
thdots_AMB = z_AMB(:,4:6);

% DAE
options_DAE = odeset('AbsTol',1e-10,'RelTol',1e-10);
[t_DAE,z_DAE] = ode45(@RHS_DAE,tspan,z0,options_DAE,p);
ths_DAE = z_DAE(:,1:p.n);
thdots_DAE = z_DAE(:,p.n+1:2*p.n);

% LAG
options_LAG = odeset('AbsTol',1e-10,'RelTol',1e-10);
[t_LAG,z_LAG] = ode45(@RHS_LAG,tspan,z0,options_LAG,p);
ths_LAG = z_LAG(:,1:3);
thdots_LAG = z_LAG(:,4:6);

%% Check conservation of energy
% Find positions, velocities of points of interest
[rP_LAG,rG_LAG,~,vG_LAG,~,~] = md697_n_pend_num_kinematics(p.n,ths_LAG
   ,thdots_LAG,1,p);
[rP_DAE,rG_DAE,~,vG_DAE,~,~] = md697_n_pend_num_kinematics(p.n,ths_DAE
   ,thdots_DAE,1,p);
[rP_AMB,rG_AMB,~,vG_AMB,~,~] = md697_n_pend_num_kinematics(p.n,ths_AMB
   ,thdots_AMB,1,p);

% Energy conservation
Ek_LAG = 0; % initialize kinetic energy
Ep_LAG = 0; % initialize potential energy
Ek_DAE = 0; % initialize kinetic energy
Ep_DAE = 0; % initialize potential energy
Ek_AMB = 0; % initialize kinetic energy
Ep_AMB = 0; % initialize potential energy
for i = 1:p.n
    Ek_LAG = Ek_LAG + 0.5*p.m(i)*dot(vG_LAG{i},vG_LAG{i}) + 0.5*p.I_G(
       i)*(thdots_LAG(:,i).').^2;
    Ep_LAG = Ep_LAG - p.m(i)*p.g*rG_LAG{i}(1,:);
```

```
    Ek_DAE = Ek_DAE + 0.5*p.m(i)*dot(vG_DAE{i},vG_DAE{i}) + 0.5*p.I_G(
       i)*(thdots_DAE(:,i).').^2;
    Ep_DAE = Ep_DAE - p.m(i)*p.g*rG_DAE{i}(1,:);
    Ek_AMB = Ek_AMB + 0.5*p.m(i)*dot(vG_AMB{i},vG_AMB{i}) + 0.5*p.I_G(
       i)*(thdots_AMB(:,i).').^2;
    Ep_AMB = Ep_AMB - p.m(i)*p.g*rG_AMB{i}(1,:);
end
E_tot_LAG = Ek_LAG+Ep_LAG; % SHOULD BE CONSTANT
E_tot_DAE = Ek_DAE+Ep_DAE; % SHOULD BE CONSTANT
E_tot_AMB = Ek_AMB+Ep_AMB; % SHOULD BE CONSTANT


%% Plot Results
while plots == 1
    figure(1)
    hold on
    plot(t_AMB,ths_AMB(:,1),'k','LineWidth',2)
    plot(t_AMB,ths_AMB(:,2),'r','LineWidth',2)
    plot(t_AMB,ths_AMB(:,3),'g','LineWidth',2)
    title('$$\theta$$ vs. time (AMB)')
    xlabel('time (s)')
    ylabel('$$\theta$$ (rad)')
    leg = legend('$$\theta_1$$','$$\theta_2$$','$$\theta_3$$','
       Location','SouthWest');
    set(leg,'Interpreter','latex')
    set(findall(gcf,'-property','FontSize'),'FontSize',14)
    grid on
    box on

    figure(2)
    hold on
    plot(t_DAE,ths_DAE(:,1),'k','LineWidth',2)
    plot(t_DAE,ths_DAE(:,2),'r','LineWidth',2)
    plot(t_DAE,ths_DAE(:,3),'g','LineWidth',2)
    title('$$\theta$$ vs. time (DAE)')
    xlabel('time (s)')
    ylabel('$$\theta$$ (rad)')
    leg = legend('$$\theta_1$$','$$\theta_2$$','$$\theta_3$$','
```

```
   Location','SouthWest');
set(leg,'Interpreter','latex')
set(findall(gcf,'-property','FontSize'),'FontSize',14)
grid on
box on

figure(3)
hold on
plot(t_LAG,ths_LAG(:,1),'k','LineWidth',2)
plot(t_LAG,ths_LAG(:,2),'r','LineWidth',2)
plot(t_LAG,ths_LAG(:,3),'g','LineWidth',2)
title('$$\theta$$ vs. time (Lagrange)')
xlabel('time (s)')
ylabel('$$\theta$$ (rad)')
leg = legend('$$\theta_1$$','$$\theta_2$$','$$\theta_3$$','
   Location','SouthWest');
set(leg,'Interpreter','latex')
set(findall(gcf,'-property','FontSize'),'FontSize',14)
grid on
box on

figure(4)
hold on
plot(t_LAG,E_tot_LAG,'k.','LineWidth',2)
plot(t_DAE,E_tot_DAE,'r.','LineWidth',2)
plot(t_AMB,E_tot_AMB,'g.','LineWidth',2)
title('Total Energy vs. Time')
xlabel('time (s)')
ylabel('$$E_{tot}$$ (J)')
leg = legend('LAG','DAE','AMB');
set(leg,'Interpreter','latex')
set(findall(gcf,'-property','FontSize'),'FontSize',14)
grid on
box on
% close all

figure(5)
hold on
```

```
plot(t_AMB,ths_AMB(:,1)-ths_DAE(:,1),'k','LineWidth',1)
plot(t_AMB,ths_AMB(:,1)-ths_LAG(:,1),'r','LineWidth',1)
plot(t_DAE,ths_DAE(:,1)-ths_LAG(:,1),'g--','LineWidth',1)
title('Comparison of $$\theta_1$$ Output vs. Time')
xlabel('time (s)')
ylabel('$$\Delta\theta_1$$ (rad)')
leg = legend('AMB-DAE','AMB-LAG','DAE-LAG','Location','NorthWest')
   ;
set(leg,'Interpreter','latex')
set(findall(gcf,'-property','FontSize'),'FontSize',14)
grid on
box on

figure(19)
hold on
plot(t_AMB,ths_AMB(:,2)-ths_DAE(:,2),'k','LineWidth',1)
plot(t_AMB,ths_AMB(:,2)-ths_LAG(:,2),'r','LineWidth',1)
plot(t_DAE,ths_DAE(:,2)-ths_LAG(:,2),'b--','LineWidth',1)
title('Comparison of $$\theta_2$$ Output vs. Time')
xlabel('time (s)')
ylabel('$$\Delta\theta_1$$ (rad)')
leg = legend('AMB-DAE','AMB-LAG','DAE-LAG','Location','NorthWest')
   ;
set(leg,'Interpreter','latex')
set(findall(gcf,'-property','FontSize'),'FontSize',14)
grid on
box on

figure(20)
hold on
plot(t_AMB,ths_AMB(:,3)-ths_DAE(:,3),'k','LineWidth',1)
plot(t_AMB,ths_AMB(:,3)-ths_LAG(:,3),'r','LineWidth',1)
plot(t_DAE,ths_DAE(:,3)-ths_LAG(:,3),'b--','LineWidth',1)
title('Comparison of $$\theta_3$$ Output vs. Time')
xlabel('time (s)')
ylabel('$$\Delta\theta_1$$ (rad)')
leg = legend('AMB-DAE','AMB-LAG','DAE-LAG','Location','NorthWest')
   ;
```

```
set(leg,'Interpreter','latex')
set(findall(gcf,'-property','FontSize'),'FontSize',14)
grid on
box on



figure(6)
hold on
plot(t_AMB,ths_AMB(:,1),'k','LineWidth',1)
plot(t_DAE,ths_DAE(:,1),'r','LineWidth',1)
plot(t_LAG,ths_LAG(:,1),'b--','LineWidth',1)
title('$$\theta_1$$ for AMB, DAE, and Lagrange')
xlabel('time (s)')
ylabel('$$\theta_1$$ (rad)')
leg = legend('AMB','DAE','LAG');
set(leg,'Interpreter','latex')
set(findall(gcf,'-property','FontSize'),'FontSize',14)
grid on
box on

figure(7)
hold on
plot(t_AMB,ths_AMB(:,2),'k','LineWidth',1)
plot(t_DAE,ths_DAE(:,2),'r','LineWidth',1)
plot(t_LAG,ths_LAG(:,2),'g--','LineWidth',1)
title('$$\theta_2$$ for AMB, DAE, and Lagrange')
xlabel('time (s)')
ylabel('$$\theta_2$$ (rad)')
leg = legend('AMB','DAE','LAG');
set(leg,'Interpreter','latex')
set(findall(gcf,'-property','FontSize'),'FontSize',14)
grid on
box on

figure(8)
hold on
```

```
    plot(t_AMB,ths_AMB(:,3),'k','LineWidth',1)
    plot(t_DAE,ths_DAE(:,3),'r','LineWidth',1)
    plot(t_LAG,ths_LAG(:,3),'g--','LineWidth',1)
    title('$$\theta_3$$ for AMB, DAE, and Lagrange')
    xlabel('time (s)')
    ylabel('$$\theta_3$$ (rad)')
    leg = legend('AMB','DAE','LAG');
    set(leg,'Interpreter','latex')
    set(findall(gcf,'-property','FontSize'),'FontSize',14)
    grid on
    box on

%     figure(9)
%     hold on
%     axis equal
%     plot(rP{1}(2,:),-rP{1}(1,:),'k','LineWidth',1)
%     plot(rP{2}(2,:),-rP{2}(1,:),'r','LineWidth',1)
%     plot(rP{3}(2,:),-rP{3}(1,:),'g','LineWidth',1)
%     title('Motion of End Points of Links')
%     leg = legend('Link 1','Link 2','Link 3');
%     set(leg,'Interpreter','latex')
%     xlabel('x (m)')
%     ylabel('y (m)')

    plots = 0;
end


%% Animate Results
% Animate AMB pendulum
if animate == 1
    md697_animate_pendulum_R2(t_AMB,p,rG_LAG,rP_LAG,'Angular Momentum
       Balance')
end

%% RHS Functions
function zdot = RHS_AMB(~,z,p)
thetas = z(1:p.n).';
```

```
thetadots = z(p.n+1:end).';

A = Amat_AMB(p.L,p.m,p.g,thetas,thetadots);
b = bvec_AMB(p.L,p.m,p.g,thetas,thetadots);

thetaddots = A\b;

zdot = [thetadots.';thetaddots];
end

function zdot = RHS_DAE(~,z,p)
%     exes = z(1:p.n).';
%     wise = z(1*p.n+1:2*p.n).';
%     thetas = z(2*p.n+1:3*p.n);
%     xdots = z(3*p.n+1:4*p.n);
%     ydots = z(4*p.n+1:5*p.n);
%     thetadots = z(5*p.n+1:6*p.n);
%     Fexes = z(6*p.n+1:7*p.n);
%     Fwise = z(7*p.n+1:8*p.n);
thetas = z(1:p.n);
thetadots = z(p.n+1:end);
A = Amat_DAE(p.L,p.m,thetas.');
b = bvec_DAE(p.L,p.m,p.g,thetas.',thetadots.');

c = A\b; % [xDDot;yDDot;thDDot;Fexes;Fwise]
theta2dots = c(2*p.n+1:3*p.n);

%     zdot = [xdots;ydots;thetadots;c];
zdot = [thetadots;theta2dots];
end

function zdot = RHS_LAG(~,z,p)
thetas = z(1:p.n).';
thetadots = z(p.n+1:end).';

A = Amat_LAG(p.L,p.m,thetas);
b = bvec_LAG(p.L,p.m,p.g,thetas,thetadots);
```

```
thetaddots = A\b;

zdot = [thetadots.';thetaddots];
end



end
```

B Four Bar Linkage Code

B.1 Set Parameters and Initial Conditions

Listing 8: md697_four_bar_DEMO.m

```
%% Mitchell Dominguez - md697 - MAE 4730 - FINAL PROJECT -
   PRESENTATION
clear;
close all;
set(0,'defaulttextinterpreter','latex');
%% FOUR BAR LINKAGE DEMO 1
% Set parameters
p.L = [1 1 2]; % lengths
p.m = [1 1 1]; % masses
p.g = 5; % acceleration due to gravity
p.n = 3; % number of links
p.I_G = (p.m .*p.L.^2)/12;

t = 20;

% Set initial conditions
init_ths = [pi-1/64;pi/2;0];
% FILL IN ONE OF thD1,thD2,or thD3 and pass in [] for the other values
% init_thDots = md697_four_bar_init_conds(p,init_ths,thD1, thD2, thD3)

init_thD1 = [];
init_thD2 = [];
init_thD3 = [];
```

```
init_thDots = md697_four_bar_init_conds(p,init_ths,init_thD1,
   init_thD2, init_thD3);

z0 = [init_ths;init_thDots];

% Output settings
rederive = 0;
plots = 1;
animate = 1;

md697_run_four_bar_linkage(p,t,z0,rederive,plots,animate)
```

B.2 Calculate Remaining Initial Angular Velocities

Listing 9: md697_four_bar_init_conds.m

```
%% Mitchell Dominguez - md697 - MAE 4730 - FINAL PROJECT
% Solve for initial conditions for four bar linkage

function init_thDots = md697_four_bar_init_conds(p,ths,thD1, thD2,
   thD3)
% Pass in desired values for an angular velocity. The function will
   then
% solve for the other angular velocities that fit

if (~isempty(thD1) && isempty(thD2) && isempty(thD3)) % Solve for thD2
    and thD3
    syms thD2 thD3 real;
    eqn = p.L(3)*thD3*[-sin(ths(3));cos(ths(3))]...
        + p.L(2)*thD2*[-sin(ths(2));cos(ths(2))]...
        + p.L(1)*thD1*[-sin(ths(1));cos(ths(1))] == zeros(2,1);
    thetadots = solve(eqn,[thD2;thD3]);

    init_thDots = [thD1;double(thetadots.thD2);double(thetadots.thD3)
       ];
elseif (isempty(thD1) && ~isempty(thD2)&&  isempty(thD3)) % Solve for
   thD1 and and thD3
    syms thD1 thD3 real;
    eqn = p.L(3)*thD3*[-sin(ths(3));cos(ths(3))]...
        + p.L(2)*thD2*[-sin(ths(2));cos(ths(2))]...
```

```
        + p.L(1)*thD1*[-sin(ths(1));cos(ths(1))] == zeros(2,1);
    thetadots = solve(eqn,[thD1;thD3]);

    init_thDots = [double(thetadots.thD1);thD2;double(thetadots.thD3)
       ];
elseif (isempty(thD1) && isempty(thD2) && ~isempty(thD3))% Solve for
   thD1 and thD2
    syms thD1 thD2 real;
    eqn = p.L(3)*thD3*[-sin(ths(3));cos(ths(3))]...
        + p.L(2)*thD2*[-sin(ths(2));cos(ths(2))]...
        + p.L(1)*thD1*[-sin(ths(1));cos(ths(1))] == zeros(2,1);
    thetadots = solve(eqn,[thD1;thD2]);

    init_thDots = [double(thetadots.thD1);double(thetadots.thD2);thD3
       ];
else % Put in too many angular velocities by accident (or none
    % in this case, calculate initial conditions using first given
       thetadot
    % AND GIVE WARNING
    if ~(isempty(thD1)) % if there was a given thD1
        init_thDots = md697_four_bar_init_conds(p,ths,thD1, [], []);
        warning(['Too many initial angular velocities given. Initial '
           ...
            'conditions were calculated using the given thD1'])
    elseif ~(isempty(thD2)) % if there was no given thD1, but a given
       thD2
        init_thDots = md697_four_bar_init_conds(p,ths,[], thD2, []);
        warning(['Too many initial angular velocities given. Initial '
           ...
            'conditions were calculated using the given thD2'])
    elseif ~(isempty(thD3)) % if there were no given thD1 or thD2, but
        given thD3
        init_thDots = md697_four_bar_init_conds(p,ths, [], [], thD3);
        warning(['Too many initial angular velocities given. Initial '
           ...
            'conditions were calculated using the given thD3'])
    else % no initial angular velocities given
        init_thDots = zeros(3,1);
```

```
        warning(['No initial angular velocities given. Initial '...
            'angular velocity was set to [0;0;0]'])
    end
end

end
```

B.3 Calculate Kinematics

Listing 10: md697_four_bar_kinematics.m

```
%% Mitchell Dominguez - md697 - MAE 4730 - FINAL PROJECT - Triple
   Pendulum Kinematics
% This script works with md697_triple_pendulum_deriver.m

% G1/O
rG1o = (L(1)/2)*f1(:,1);
vG1o = jacobian(rG1o,th(1))*thDot(1);
aG1o = jacobian(vG1o,[th(1) thDot(1)])*[thDot(1) thDDot(1)].';

% P2/O
rP2o = 2*rG1o;
vP2o = 2*vG1o;
aP2o = 2*aG1o;

% G2/P2
rG2p2 = (L(2)/2)*f1(:,2);
vG2p2 = jacobian(rG2p2,th(2))*thDot(2);
aG2p2 = jacobian(vG2p2,[th(2) thDot(2)])*[thDot(2) thDDot(2)].';

% G2/O
rG2o = rP2o + rG2p2;
vG2o = vP2o + vG2p2;
aG2o = aP2o + aG2p2;

% P3/P2
rP3p2 = 2*rG2p2;
vP3p2 = 2*vG2p2;
aP3p2 = 2*aG2p2;
```

```
% P3/O
rP3o = rP2o + rP3p2;
vP3o = vP2o + vP3p2;
aP3o = aP2o + aP3p2;

% G3/P3
rG3p3 = (L(3)/2)*f1(:,3);
vG3p3 = jacobian(rG3p3,th(3))*thDot(3);
aG3p3 = jacobian(vG3p3,[th(3) thDot(3)])*[thDot(3) thDDot(3)].';

% G3/O
rG3o = rP3o + rG3p3;
vG3o = vP3o + vG3p3;
aG3o = aP3o + aG3p3;

% G3/P2
rG3p2 = rP3p2 + rG3p3;
vG3p2 = vP3p2 + vG3p3;
aG3p2 = aP3p2 + aG3p3;

% P4/O
rP4o = rP3o + 2*rG3p3;
vP4o = vP3o + 2*vG3p3;
aP4o = aP3o + 2*aG3p3;
```

B.4 Derive Equations of Motion

Listing 11: md697_four_bar_deriver.m

```
%% Mitchell Dominguez - md697 - MAE 4730 - FINAL PROJECT
% FOUR BAR LINKAGE DAE EQUATIONS FOUR BAR LINKAGE
% close all;
% clear;
function md697_four_bar_deriver
%% Set up symbolic scalars
n = 3; % Number of pendulum links
syms g; % acceleration due to gravity
L = sym('L', [1 n], 'positive'); % array of link lengths
m = sym('m', [1 n], 'positive'); % array of link masses
th = sym('th', [1 n], 'real'); % array of thetas
```

```
thDot = sym('thD', [1 n], 'real'); % array of theta dots
thDDot = sym('thDD', [1 n], 'real'); % array of theta double dots
xDDot = sym('xDDot', [1 n], 'real'); % x double dot of each COM
yDDot = sym('yDDot', [1 n], 'real'); % y double dot of each COM
Fx = sym('Fx', [1 n+1], 'real'); % Constraint forces in x direction
Fy = sym('Fy', [1 n+1], 'real'); % Constraint forces in y direction
I_G = (m .*L.^2)/12; % Moments of inertia about centers of mass

%% Set up symbolic vectors
f1 = [cos(th);sin(th);zeros(1,n)]; % lambda vectors for each body
   frame
f2 = [-sin(th);cos(th);zeros(1,n)]; % normal vectors for each body
   frame
e1 = [1;0;0]; % e1
e2 = [0;1;0]; % e2
e3 = [0;0;1]; % e3 (same for each frame)

iWf = [zeros(2,n);thDot]; % I omega (frame)
iAf = [zeros(2,n);thDDot]; % I alpha (frame)

%% Kinematics
md697_four_bar_kinematics % Calculate kinematics

%% Simplifying Terms
IA = sym(zeros(3,n));
IA(3,:) = thDDot.*I_G; % matrix of moment of inertia * I alpha (frame)
    terms

%% DAEs
md697_four_bar_DAE;

%% Generate EOM Matrix Equation
% DAE
[DAE_A,DAE_b] = equationsToMatrix(DAE_eqns,[xDDot, yDDot, thDDot, Fx,
   Fy]);
DAE_A = simplify(DAE_A);
DAE_b = simplify(DAE_b);
```

```
%% Create MATLAB Function for EOM
% DAE
matlabFunction(DAE_A,'file','Amat_DAE_4bar','Vars',{L,m,th});
matlabFunction(DAE_b,'file','bvec_DAE_4bar','Vars',{L,m,g,th,thDot});




end
```

B.5 Find Differential Algebraic Equations

Listing 12: md697_four_bar_DAE.m

```
%% Mitchell Dominguez - md697 - MAE 4730 - FINAL PROJECT
% 4 Bar Linkage DAEs
% LMB link 1
DAE_eqns{1} = Fx(1) + m(1)*g + Fx(2) == m(1)*xDDot(1);
DAE_eqns{2} = Fy(1) + Fy(2) == m(2)*yDDot(1);

% LMB link 2
DAE_eqns{3} = -Fx(2) + m(2)*g + Fx(3) == m(2)*xDDot(2);
DAE_eqns{4} = -Fy(2) + Fy(3) == m(2)*yDDot(2);

% LMB link 3
DAE_eqns{5} = Fx(4) -Fx(3) + m(3)*g == m(3)*xDDot(3); % Changed this
   line
DAE_eqns{6} = Fy(4) -Fy(3) == m(3)*yDDot(3); % Changed this line

% AMB link 1
temp_7 = -(L(1)/2)*cross(f1(:,1),Fx(1)*e1 + Fy(1)*e2)...
    + (L(1)/2)*cross(f1(:,1),Fx(2)*e1 + Fy(2)*e2);
DAE_eqns{7} = temp_7(3) == IA(3,1);

% AMB link 2
temp_8 = -(L(2)/2)*cross(f1(:,2),-Fx(2)*e1 - Fy(2)*e2)...
    + (L(2)/2)*cross(f1(:,2),Fx(3)*e1 + Fy(3)*e2);
DAE_eqns{8} = temp_8(3) == IA(3,2);

% AMB link 3
```

```
temp_9 = -(L(3)/2)*cross(f1(:,3),-Fx(3)*e1 - Fy(3)*e2)...
    + (L(3)/2)*cross(f1(:,3),Fx(4)*e1 + Fy(4)*e2); % Changed this line
DAE_eqns{9} = temp_9(3) == IA(3,3);

% Constraint COM 1
DAE_eqns{10} = xDDot(1) == aG1o(1);
DAE_eqns{11} = yDDot(1) == aG1o(2);

% Constraint COM 2
DAE_eqns{12} = xDDot(2) == aG2o(1);
DAE_eqns{13} = yDDot(2) == aG2o(2);

% Constraint COM 3
DAE_eqns{14} = xDDot(3) == aG3o(1);
DAE_eqns{15} = yDDot(3) == aG3o(2);

% Constraint End Point
DAE_eqns{16} = aP4o(1) == 0;
DAE_eqns{17} = aP4o(2) == 0;
```

B.6 Run Four Bar Linkage

Listing 13: md697_run_four_bar_linkage.m

```
%% Mitchell Dominguez - md697 - MAE 4730 - FINAL PROJECT - Run Four
   Bar Linkage
% Run four bar linkage using functions derived from
% md697_four_bar_deriver

%
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%% Run using md697_four_bar_DEMO
   %%%%%%%%%%%%%%%%%%%%%%%%%
%
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function md697_run_four_bar_linkage(p,t,z0,rederive,plots,animate)
%% Derive EOM
```

```
% rederive = input('Re-derive EOM? Enter 0 for no and 1 for yes\n');
if rederive ~= 0
    md697_four_bar_deriver
end
% clear; % Re-use variables from kinematics so need to clear symbolic
   vars
% %% Set parameters
% p.L = [1 9 1]; % lengths
% p.m = [1 5 1]; % masses
% p.g = 5; % acceleration due to gravity
% p.n = 3; % number of links
% I_G = (p.m .*p.L.^2)/12;
%
% % tspan = [0 100];
% t = 100;
numpts = 60*t; % 60 points per second
tspan = linspace(0,t,numpts);

% %% Set initial conditions
% % init_ths = [88*pi/180 + pi/2 ; pi/2; 0];
% % init_ths = [pi;pi;pi];
% % init_ths = [pi/2;deg2rad(90+3);deg2rad(270+90)];
% init_ths = [deg2rad(90+70);pi/2;deg2rad(90-20)];
% % FILL IN ONE OF thD1,thD2,or thD3 and pass in [] for the other
   values
% % init_thDots = md697_four_bar_init_conds(p,init_ths,thD1, thD2,
   thD3)
%
% init_thD1 = [5];
% init_thD2 = [];
% init_thD3 = [];
%
% init_thDots = md697_four_bar_init_conds(p,init_ths,init_thD1,
   init_thD2, init_thD3);
%
% z0 = [init_ths;init_thDots];

%% Integrate
```

```
% DAE
options = odeset('AbsTol',1e-10,'RelTol',1e-10);
[t,z] = ode45(@RHS_DAE,tspan,z0,options,p);
ths = z(:,1:p.n);
thdots = z(:,p.n+1:2*p.n);

%% Check conservation of energy
% Find positions, velocities of points of interest
[rP,rG,vP,vG,~,~] = md697_n_pend_num_kinematics(p.n,ths,thdots,1,p);

% Energy conservation
Ek = 0; % initialize kinetic energy
Ep = 0; % initialize potential energy
for i = 1:p.n
    Ek = Ek + 0.5*p.m(i)*dot(vG{i},vG{i}) + 0.5*p.I_G(i)*(thdots(:,i)
       .').^2;
    Ep = Ep - p.m(i)*p.g*rG{i}(1,:);
end
E_tot = Ek+Ep; % SHOULD BE CONSTANT

%% Plot Results
while plots ~= 0
    figure(1)
    hold on
    plot(t,ths(:,1),'k','LineWidth',2)
    plot(t,ths(:,2),'r','LineWidth',2)
    plot(t,ths(:,3),'g','LineWidth',2)
    title('$$\theta$$ vs. time (DAE)')
    xlabel('time (s)')
    ylabel('$$\theta$$ (rad)')
    leg = legend('$$\theta_1$$','$$\theta_2$$','$$\theta_3$$');
    set(leg,'Interpreter','latex')
    set(findall(gcf,'-property','FontSize'),'FontSize',14)
    grid on
    box on

    figure(2)
    hold on
```

```
    plot(t,E_tot,'k.','LineWidth',2)
    title('Total Energy vs. Time (DAE)')
    xlabel('time (s)')
    ylabel('$$E_{tot}$$ (J)')
    set(findall(gcf,'-property','FontSize'),'FontSize',14)
    grid on
    box on

    figure(3)
    hold on
    axis equal
    plot(rP{1}(2,:),-rP{1}(1,:),'k','LineWidth',1)
    plot(rP{2}(2,:),-rP{2}(1,:),'r','LineWidth',1)
    plot(rP{3}(2,:),-rP{3}(1,:),'g','LineWidth',1)
    leg = legend('Link 1','Link 2','Link 3');
    set(leg,'Interpreter','latex')
    set(findall(gcf,'-property','FontSize'),'FontSize',14)
    title('Motion of End Points of Links')
    xlabel('x (m)')
    ylabel('y (m)')

    figure(4)
    hold on
    plot(t,rP{3}(2,:)-rP{3}(2,1),'k--','LineWidth',1)
    plot(t,-rP{3}(1,:)+rP{3}(1,1),'r--','LineWidth',1)
    title('X and Y of Pinned Point vs. Time')
    xlabel('time (s)')
    ylabel('position (m)')
    leg = legend('X','Y');
    set(leg,'Interpreter','latex')
    set(findall(gcf,'-property','FontSize'),'FontSize',14)
    plots = 0;
end


%% Animate Results
if animate ~=0
    md697_animate_pendulum_R2(t,p,rG,rP,'Four Bar Linkage Animation')
```

```
end



%% RHS Files
    function zdot = RHS_DAE(~,z,p)
        thetas = z(1:p.n);
        thetadots = z(p.n+1:end);
        A = Amat_DAE_4bar(p.L,p.m,thetas.');
        b = bvec_DAE_4bar(p.L,p.m,p.g,thetas.',thetadots.');

        c = A\b; % [xDDot;yDDot;thDDot;Fexes;Fwise]
        theta2dots = c(2*p.n+1:3*p.n);

        %     zdot = [xdots;ydots;thetadots;c];
        zdot = [thetadots;theta2dots];
    end


end
```

C N-Link Pendulum Code

C.1 Set Parameters and Initial Conditions

Listing 14: md697_n_pend_DEMO.m

```
%% Mitchell Dominguez - md697 - MAE 4730 - FINAL PROJECT -
   PRESENTATION
clear;
close all;
set(0,'defaulttextinterpreter','latex');
%% TRIPLE PENDULUM DEMO 1
% Set parameters
p.n = 20; % number of links
p.L = ones(1,p.n); % lengths
p.m = ones(1,p.n); % masses
p.g = 10; % acceleration due to gravity
p.I_G = (p.m .*p.L.^2)/12;
```

```
% Initial Conditions
t = 20; % Time to integrate for

init_ths = pi/2*ones(1,p.n).';
init_thDots = zeros(1,p.n).';
% init_thDots = [5;-2;1];

z0 = [init_ths;init_thDots]; % Initial conditions

% Output settings
rederive = 0;
plots = 1; % KEEP LIKE THIS UNTIL PLOTTING IS FIGURED OUT
animate = 1;

% Run triple pendulum
md697_run_n_pend(p,t,z0,rederive,plots,animate)
```

C.2 Calculate Kinematics

Listing 15: md697_n_pend_sym_kinematics.m

```
%% Mitchell Dominguez - md697 - MAE 4730 - FINAL PROJECT
%% N-Link Pendulum Symbolic Kinematics
% This script works with md697_triple_pendulum_deriver.m

% Positions rel O (P1)
rG = sym(zeros(3,n)); % r_Gi/o
rP = sym(zeros(3,n)); % r_Pi/o
rG(:,1) = (L(1)/2)*f1(:,1);
rP(:,1) = 2*rG(:,1);
for i = 2:n
    rG(:,i) = rP(:,i-1) + (L(i)/2)*f1(:,i);
    rP(:,i) = rP(:,i-1) + L(i)*f1(:,i);
end

% Velocities rel O (P1)
vG = sym(zeros(3,n));
vP = sym(zeros(3,n));
vG(:,1) = jacobian(rG(:,1),th(1))*thDot(1);
```

```
vP(:,1) = 2*vG(:,1);
for i = 2:n
    vG(:,i) = vP(:,i-1) + jacobian(rG(:,i)-rP(:,i-1),th(i))*thDot(i);
    vP(:,i) = vP(:,i-1) + jacobian(rP(:,i)-rP(:,i-1),th(i))*thDot(i);
end

% Accelerations rel O (P1)
aG = sym(zeros(3,n));
aP = sym(zeros(3,n));
aG(:,1) = jacobian(vG(:,1),[th(1) thDot(1)])*[thDot(1) thDDot(1)].';
aP(:,1) = 2*aG(:,1);
for i = 2:n
    aG(:,i) = aP(:,i-1) + jacobian(vG(:,i)-vP(:,i-1),[th(i) thDot(i)])
       *[thDot(i) thDDot(i)].';
    aP(:,i) = aP(:,i-1) + jacobian(vP(:,i)-vP(:,i-1),[th(i) thDot(i)])
       *[thDot(i) thDDot(i)].';
end
% aG1o = jacobian(vG1o,[th(1) thDot(1)])*[thDot(1) thDDot(1)].';
```

C.3 Derive Equations of Motion

Listing 16: md697_n_pend_deriver.m

```
%% Mitchell Dominguez - md697 - MAE 4730 - FINAL PROJECT - Triple
   Pendulum Deriver
%% Mitchell Dominguez - md697 - MAE 4730 - FINAL PROJECT - Triple
   Pendulum Deriver
% close all;
% clear;
function md697_n_pend_deriver(n)
%% Set up symbolic scalars
% n = 12; % Number of pendulum links
syms g; % acceleration due to gravity
L = sym('L', [1 n], 'positive'); % array of link lengths
m = sym('m', [1 n], 'positive'); % array of link masses
th = sym('th', [1 n], 'real'); % array of thetas
thDot = sym('thD', [1 n], 'real'); % array of theta dots
thDDot = sym('thDD', [1 n], 'real'); % array of theta double dots
xDDot = sym('xDDot', [1 n], 'real'); % x double dot of each COM
yDDot = sym('yDDot', [1 n], 'real'); % y double dot of each COM
```

```
Fx = sym('Fx', [1 n], 'real'); % Constraint forces in x direction
Fy = sym('Fy', [1 n], 'real'); % Constraint forces in y direction
I_G = (m .*L.^2)/12; % Moments of inertia about centers of mass

%% Set up symbolic vectors
f1 = [cos(th);sin(th);zeros(1,n)]; % lambda vectors for each body
   frame
f2 = [-sin(th);cos(th);zeros(1,n)]; % normal vectors for each body
   frame
e1 = [1;0;0]; % e1
e2 = [0;1;0]; % e2
e3 = [0;0;1]; % e3 (same for each frame)

iWf = [zeros(2,n);thDot]; % I omega (frame)
iAf = [zeros(2,n);thDDot]; % I alpha (frame)

%% Kinematics
% md697_triple_pendulum_kinematics % Calculate kinematics
md697_n_pend_sym_kinematics % Calculate kinematics (n link)

%% Simplifying terms
mg = e1*m.*g; % matrix of gravity force terms

IA = sym(zeros(3,n));
IA(3,:) = thDDot.*I_G; % matrix of moment of inertia * I alpha (frame)
    terms

ma = (diag(m) * aG.').'; % matrix of m*a terms

%% AMB Equations
% md697_triple_pendulum_AMB_eqns

%% DAE Equations
% md697_triple_pendulum_DAE_eqns

%% LAG Equations
md697_triple_pendulum_LAG_eqns
```

```
%% Generate Equations of Motion
% % AMB
% [AMB_A,AMB_b] = equationsToMatrix(AMB_eqns,thDDot);
% AMB_A = simplify(AMB_A);
% AMB_b = simplify(AMB_b);
%
% % DAE
% [DAE_A,DAE_b] = equationsToMatrix(DAE_eqns,[xDDot, yDDot, thDDot, Fx
    , Fy]);
% DAE_A = simplify(DAE_A);
% DAE_b = simplify(DAE_b);

% LAG
[LAG_A,LAG_b] = equationsToMatrix(LAG_eqns,thDDot);
LAG_A = simplify(LAG_A);
LAG_b = simplify(LAG_b);


%% Create MATLAB Function for EOM
% % AMB
% matlabFunction(AMB_A,'file','Amat_AMB','Vars',{L,m,g,th,thDot});
% matlabFunction(AMB_b,'file','bvec_AMB','Vars',{L,m,g,th,thDot});
%
% % DAE
% matlabFunction(DAE_A,'file','Amat_DAE','Vars',{L,m,th});
% matlabFunction(DAE_b,'file','bvec_DAE','Vars',{L,m,g,th,thDot});

% DAE
matlabFunction(LAG_A,'file',['Amat_LAG_',num2str(n),'_link'],'Vars',{L
    ,m,th});
matlabFunction(LAG_b,'file',['bvec_LAG_',num2str(n),'_link'],'Vars',{L
    ,m,g,th,thDot});
```

```
end
```

C.4 Find Lagrange Equations

This code is the same as was used to find the Lagrange equations for the triple pendulum. See A.6 for the code that was used to do this.

C.5 Run N-Link Pendulum

Listing 17: md697_run_n_pend.m

```
%% Mitchell Dominguez - md697 - MAE 4730 - FINAL PROJECT - Run Triple
   Pendulum
% Run triple pendulum using functions derived from
% md697_triple_pendulum_deriver

%
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%% Run using md697_DEMO
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function md697_run_n_pend(p,t,z0,rederive,plots,animate)
%% Derive EOM
% rederive = input('Re-derive EOM? Enter 0 for no and 1 for yes\n');
if rederive ~= 0
    md697_n_pend_deriver(p.n)
end
% clear; % Re-use variables from kinematics so need to clear symbolic
   vars
%% Set parameters
% p.L = [1 1 1]; % lengths
% p.m = [1 1 1]; % masses
% p.g = 1; % acceleration due to gravity
```

```
% p.n = 3; % number of links
% p.I_G = (p.m .*p.L.^2)/12;
%
% % tspan = [0 100];
% t = 20;
numpts = 60*t; % 60 points per second
tspan = linspace(0,t,numpts);

%% Set initial conditions
% init_ths = [pi/2; pi/2; pi/2];
% % init_ths = [pi;pi;pi];
% init_thDots = [0;0;0];
%
% z0 = [init_ths;init_thDots];


%% Integrate
% % AMB
% options_AMB = odeset('AbsTol',1e-10,'RelTol',1e-10);
% [t_AMB,z_AMB] = ode45(@RHS_AMB,tspan,z0,options_AMB,p);
% ths_AMB = z_AMB(:,1:3);
% thdots_AMB = z_AMB(:,4:6);

% % DAE
% options_DAE = odeset('AbsTol',1e-10,'RelTol',1e-10);
% [t_DAE,z_DAE] = ode45(@RHS_DAE,tspan,z0,options_DAE,p);
% ths_DAE = z_DAE(:,1:p.n);
% thdots_DAE = z_DAE(:,p.n+1:2*p.n);

% LAG
options_LAG = odeset('AbsTol',1e-10,'RelTol',1e-10);
[t_LAG,z_LAG] = ode45(@RHS_LAG,tspan,z0,options_LAG,p);
ths_LAG = z_LAG(:,1:p.n);
thdots_LAG = z_LAG(:,p.n+1:2*p.n);

%% Check conservation of energy
% Find positions, velocities of points of interest
```

```
[rP,rG,vP,vG,~,~] = md697_n_pend_num_kinematics(p.n,ths_LAG,thdots_LAG
   ,1,p);

% Energy conservation
Ek = 0; % initialize kinetic energy
Ep = 0; % initialize potential energy
for i = 1:p.n
    Ek = Ek + 0.5*p.m(i)*dot(vG{i},vG{i}) + 0.5*p.I_G(i)*(thdots_LAG
       (:,i).').^2;
    Ep = Ep - p.m(i)*p.g*rG{i}(1,:);
end
E_tot = Ek+Ep; % SHOULD BE CONSTANT


%% Plot Results
while plots == 1
%     figure(1)
%     hold on
%     plot(t_AMB,ths_AMB(:,1),'k','LineWidth',2)
%     plot(t_AMB,ths_AMB(:,2),'r','LineWidth',2)
%     plot(t_AMB,ths_AMB(:,3),'g','LineWidth',2)
%     title('\theta vs. time (AMB)')
%     xlabel('time (s)')
%     ylabel('\theta (rad)')
%     leg = legend('\theta_1','\theta_2','\theta_3')
%     grid on
%     box on

%     figure(2)
%     hold on
%     plot(t_DAE,ths_DAE(:,1),'k','LineWidth',2)
%     plot(t_DAE,ths_DAE(:,2),'r','LineWidth',2)
%     plot(t_DAE,ths_DAE(:,3),'g','LineWidth',2)
%     title('\theta vs. time (DAE)')
%     xlabel('time (s)')
%     ylabel('\theta (rad)')
%     leg = legend('\theta_1','\theta_2','\theta_3')
%     grid on
```

```
%     box on

    figure(3)
    hold on
%     legendnames = cell(1,p.n);
    for i = 1:p.n
        plot(t_LAG,ths_LAG(:,i),'LineWidth',2)
%         legendnames{i} = ['$$\theta_',num2str(i),'$$'];
    end
%     plot(t_LAG,ths_LAG(:,2),'r','LineWidth',2)
%     plot(t_LAG,ths_LAG(:,3),'g','LineWidth',2)
    title('$$\theta$$ vs. time (Lagrange)')
    xlabel('time (s)')
    ylabel('$$\theta$$ (rad)')
%     leg = legend('\theta_1','\theta_2','\theta_3')
%     leg = legend(legendnames);
%     set(leg,'Interpreter','latex')
    set(findall(gcf,'-property','FontSize'),'FontSize',14)
    grid on
    box on

    figure(4)
    hold on
    plot(t_LAG,E_tot,'k.','LineWidth',2)
    title('Total Energy vs. Time (LAG)')
    xlabel('time (s)')
    ylabel('$$E_{tot}$$ (J)')
    set(findall(gcf,'-property','FontSize'),'FontSize',14)
    grid on
    box on
    % close all

%     figure(5)
%     hold on
%     plot(t_AMB,ths_AMB(:,1)-ths_DAE(:,1),'k','LineWidth',2)
%     plot(t_AMB,ths_AMB(:,1)-ths_LAG(:,1),'r','LineWidth',2)
%     plot(t_DAE,ths_DAE(:,1)-ths_LAG(:,1),'g','LineWidth',2)
%     title('Comparison of \theta_1 Output vs. Time')
```

```
%     xlabel('time (s)')
%     ylabel('\Delta\theta_1 (rad)')
%     leg = legend('AMB-DAE','AMB-LAG','DAE-LAG')
%     grid on
%     box on

%     figure(6)
%     hold on
%     plot(t_AMB,ths_AMB(:,1),'k','LineWidth',1)
%     plot(t_DAE,ths_DAE(:,1),'r','LineWidth',1)
%     plot(t_LAG,ths_LAG(:,1),'g--','LineWidth',1)
%     title('\theta_1 for AMB, DAE, and Lagrange')
%     xlabel('time (s)')
%     ylabel('\theta_1 (rad)')
%     leg = legend('AMB','DAE','LAG')
%     grid on
%     box on

%     figure(7)
%     hold on
%     plot(t_AMB,ths_AMB(:,2),'k','LineWidth',1)
%     plot(t_DAE,ths_DAE(:,2),'r','LineWidth',1)
%     plot(t_LAG,ths_LAG(:,2),'g--','LineWidth',1)
%     title('\theta_2 for AMB, DAE, and Lagrange')
%     xlabel('time (s)')
%     ylabel('\theta_2 (rad)')
%     leg = legend('AMB','DAE','LAG')
%     grid on
%     box on

%     figure(8)
%     hold on
%     plot(t_AMB,ths_AMB(:,3),'k','LineWidth',1)
%     plot(t_DAE,ths_DAE(:,3),'r','LineWidth',1)
%     plot(t_LAG,ths_LAG(:,3),'g--','LineWidth',1)
%     title('\theta_3 for AMB, DAE, and Lagrange')
%     xlabel('time (s)')
%     ylabel('\theta_3 (rad)')
```

```
%     leg = legend('AMB','DAE','LAG')
%     grid on
%     box on

%     figure(9)
%     hold on
%     axis equal
%     plot(rP{1}(2,:),-rP{1}(1,:),'k','LineWidth',1)
%     plot(rP{2}(2,:),-rP{2}(1,:),'r','LineWidth',1)
%     plot(rP{3}(2,:),-rP{3}(1,:),'g','LineWidth',1)
%     title('Motion of End Points of Links')
%     leg = legend('Link 1','Link 2','Link 3')
%     xlabel('x (m)')
%     ylabel('y (m)')

    plots = 0;
end

%% Save Results
% save([num2str(p.n),'_link_pendulum_data.mat']);

%% Animate Results
% Animate AMB pendulum
if animate == 1
    md697_animate_pendulum_R2(t_LAG,p,rG,rP,['Lagrange ',num2str(p.n),
       ' Link Pendulum'])
end



%% RHS Functions
function zdot = RHS_AMB(~,z,p)
thetas = z(1:p.n).';
thetadots = z(p.n+1:end).';

A = Amat_AMB(p.L,p.m,p.g,thetas,thetadots);
b = bvec_AMB(p.L,p.m,p.g,thetas,thetadots);
```

```
thetaddots = A\b;

zdot = [thetadots.';thetaddots];
end

function zdot = RHS_DAE(~,z,p)
%     exes = z(1:p.n).';
%     wise = z(1*p.n+1:2*p.n).';
%     thetas = z(2*p.n+1:3*p.n);
%     xdots = z(3*p.n+1:4*p.n);
%     ydots = z(4*p.n+1:5*p.n);
%     thetadots = z(5*p.n+1:6*p.n);
%     Fexes = z(6*p.n+1:7*p.n);
%     Fwise = z(7*p.n+1:8*p.n);
thetas = z(1:p.n);
thetadots = z(p.n+1:end);
A = Amat_DAE(p.L,p.m,thetas.');
b = bvec_DAE(p.L,p.m,p.g,thetas.',thetadots.');

c = A\b; % [xDDot;yDDot;thDDot;Fexes;Fwise]
theta2dots = c(2*p.n+1:3*p.n);

%     zdot = [xdots;ydots;thetadots;c];
zdot = [thetadots;theta2dots];
end

function zdot = RHS_LAG(~,z,p)
thetas = z(1:p.n).';
thetadots = z(p.n+1:end).';

A_call = ['Amat_LAG_',num2str(p.n),'_link(p.L,p.m,thetas)'];
b_call = ['bvec_LAG_',num2str(p.n),'_link(p.L,p.m,p.g,thetas,thetadots
   )'];

A = eval(A_call);
b = eval(b_call);
% A = Amat_LAG(p.L,p.m,thetas);
% b = bvec_LAG(p.L,p.m,p.g,thetas,thetadots);
```

```
thetaddots = A\b;

zdot = [thetadots.';thetaddots];
end



end
```

D Miscellaneous Code

D.1 Calculate Position and Velocity at Each Time Step

Listing 18: md697_n_pend_num_kinematics.m

```
%% Mitchell Dominguez - md697 - MAE 4730 - FINAL PROJECT
%% N-Link Pendulum Symbolic Kinematics
% This function is called from md697_run_triple_pendulum OR
% md697_run_n_pendulum
function [rP,rG,vP,vG,aP,aG] = md697_n_pend_num_kinematics(n,th,thDot,
   thDDot,p)
%% Set up vectors
f1 = cell(1,n);
f2 = cell(1,n);
for i = 1:n
    f1{i} = [cos(th(:,i)).';sin(th(:,i)).';zeros(length(th(:,i)),1)
       .']; % Tangent vectors for each body frame
    f2{i} = [-sin(th(:,i)).';cos(th(:,i)).';zeros(length(th(:,i)),1)
       .']; % normal vectors for each body frame
%     iWf{i} = [zeros(2,length(thDot(:,i)));thDot(:,i).']; % I omega (
   frame)
%     iAf{i} = [zeros(2,n);thDDot.']; % I alpha (frame)
end

% Positions rel O (P1)
rG = cell(1,n);
rP = cell(1,n);
rG{1} = (p.L(1)/2)*f1{1};
```

```
rP{1} = 2*rG{1};
for i = 2:n
    rG{i} = rP{i-1} + (p.L(i)/2)*f1{i};
    rP{i} = rP{i-1} + p.L(i)*f1{i};
end

% Velocities rel O (P1)
% vG = zeros(3,n);
% vP = zeros(3,n);
% vG(:,1) = jacobian(rG(:,1),th(1))*thDot(1);
% vP(:,1) = 2*vG(:,1);
vG = cell(1,n);
vP = cell(1,n);
vG{1} = (p.L(1)/2)*[thDot(:,1).';thDot(:,1).';thDot(:,1).'].*f2{1};
vP{1} = 2*vG{1};
for i = 2:n
    %     vG{i} = vP{i-1} + jacobian(rG(:,i)-rP(:,i-1),th(i))*thDot(i)
       ;
    %     vP{i} = vP{i-1} + jacobian(rP(:,i)-rP(:,i-1),th(i))*thDot(i)
       ;
    vG{i} = vP{i-1} + (p.L(i)/2)*[thDot(:,i).';thDot(:,i).';thDot(:,i)
       .'].*f2{i};
    vP{i} = vP{i-1} + p.L(i)*[thDot(:,i).';thDot(:,i).';thDot(:,i)
       .'].*f2{i};
end
%
% % Accelerations rel O (P1)
% aG = zeros(3,n);
% aP = zeros(3,n);
% aG(:,1) = jacobian(vG(:,1),[th(1) thDot(1)])*[thDot(1) thDDot(1)].';
% aP(:,1) = 2*aG(:,1);
% for i = 2:n
%     aG(:,i) = aP(:,i-1) + jacobian(vG(:,i)-vP(:,i-1),[th(i) thDot(i)
   ])*[thDot(i) thDDot(i)].';
%     aP(:,i) = aP(:,i-1) + jacobian(vP(:,i)-vP(:,i-1),[th(i) thDot(i)
   ])*[thDot(i) thDDot(i)].'
% end
% % aG1o = jacobian(vG1o,[th(1) thDot(1)])*[thDot(1) thDDot(1)].';
```

```
aP = 1;
aG = 1;

end
```

D.2 Animate Results

Listing 19: md697_animate_pendulum_R2.m

```
%% Mitchell Dominguez - md697 - MAE 4730 - FINAL PROJECT - Animate N
   Link Pendulum
% Does what the title says. Animates the N link pendulum (or 4 bar
   linkage
% if you are so inclined)
function md697_animate_pendulum_R2(t,p,rG,rP,titleText)
if p.n == 3
    figure()
    hold on
    axis equal
    plot(rP{1}(2,:),-rP{1}(1,:),'k','LineWidth',1)
    plot(rP{2}(2,:),-rP{2}(1,:),'r','LineWidth',1)
    plot(rP{3}(2,:),-rP{3}(1,:),'g','LineWidth',1)
    title('Motion of End Points of 3 Links')
    leg = legend('Link 1','Link 2','Link 3');
    set(leg,'Interpreter','latex')
    set(findall(gcf,'-property','FontSize'),'FontSize',14)
    xlabel('x (m)')
    ylabel('y (m)')
    grid on
    box on
else
    figure()
    hold on
    axis equal
    for i = 1:p.n
        plot(rP{i}(2,:),-rP{i}(1,:),'k','LineWidth',1)
    end
    title(['Motion of End Points of ',num2str(p.n),' Links'])
    set(findall(gcf,'-property','FontSize'),'FontSize',14)
    xlabel('x (m)')
```

```
    ylabel('y (m)')
    grid on
    box on
end

figure()
hold on
axis equal
axis([-sum(p.L),sum(p.L),-sum(p.L),sum(p.L)])
xlabel('X(m)')
ylabel('Y(m)')
title(titleText)
set(findall(gcf,'-property','FontSize'),'FontSize',14)

%% Animate
link = cell(1,p.n);

for i = 1:p.n
    link{i} = plot(0,0,'LineWidth',2);
end

% for i = 1:p.n
%     set(link{i}, 'Xdata',[0 rP{i}(2,1)],...
%         'Ydata',[0 -rP{i}(1,1)] )
% end
for k = 1:1
    set(link{1}, 'Xdata',[0 rP{1}(2,k)],...
        'Ydata',[0 -rP{1}(1,k)] )
    for i = 2:p.n
        set(link{i}, 'Xdata', [rP{i-1}(2,k), rP{i}(2,k)],...
            'Ydata', [-rP{i-1}(1,k), -rP{i}(1,k)])
    end

    drawnow
%     pause(5e-3)

end
pause(2.5)
```

```
tic
tnow = toc;
% Linearly interpolating to animate in real time
while tnow<=t(end) % While time is less than or equal to end time

    set(link{1}, 'Xdata',[0 interp1(t,rP{1}(2,:),tnow)],...
        'Ydata',[0 interp1(t,-rP{1}(1,:),tnow)] )
    for i = 2:p.n
        set(link{i}, 'Xdata', [interp1(t,rP{i-1}(2,:),tnow),interp1(t,
           rP{i}(2,:),tnow)],...
            'Ydata', [interp1(t,-rP{i-1}(1,:),tnow), interp1(t,-rP{i
               }(1,:),tnow)])
    end

    drawnow
    tnow = toc;
    %     pause(5e-3)
end

if p.n == 3
    hold on
    axis equal
    plot(rP{1}(2,:),-rP{1}(1,:),'k','LineWidth',1)
    plot(rP{2}(2,:),-rP{2}(1,:),'r','LineWidth',1)
    plot(rP{3}(2,:),-rP{3}(1,:),'g','LineWidth',1)
    title('Motion of End Points of 3 Links')
    leg = legend('Link 1','Link 2','Link 3');
    set(leg,'Interpreter','latex')
    set(findall(gcf,'-property','FontSize'),'FontSize',14)
    xlabel('x (m)')
    ylabel('y (m)')
    grid on
    box on
else
    hold on
    axis equal
    for i = 1:p.n
        plot(rP{i}(2,:),-rP{i}(1,:),'k','LineWidth',1)
```

```
    end
    title(['Motion of End Points of ',num2str(p.n),' Links'])
    set(findall(gcf,'-property','FontSize'),'FontSize',14)
    xlabel('x (m)')
    ylabel('y (m)')
    grid on
    box on
end
%% OLD ANIMATION
% for k = 1:length(t)
%     set(link{1}, 'Xdata',[0 rP{1}(2,k)],...
%         'Ydata',[0 -rP{1}(1,k)] )
%     for i = 2:p.n
%         set(link{i}, 'Xdata', [rP{i-1}(2,k), rP{i}(2,k)],...
%             'Ydata', [-rP{i-1}(1,k), -rP{i}(1,k)])
%     end
%
%     drawnow
% %     pause(5e-3)
%
% end
hold off









end
```