MAE 5730 Final Report

1. Introduction

This report consists of derivation, simulation, and data analysis of a triple pendulum system using Lagrange, Newton-Euler, and Differential Algebraic Equation (DAE) methods as well as a 4-bar linkage system using the DAE method. An n-linked pendulum deriver is also developed using the Lagrange method and multivariable chain rule. Periodic solutions of the triple and quadruple pendulum systems were also found using constrained nonlinear minimization (fmincon.m).

2. Pendulum

The system consists of three links, each with their own masses, lengths, and moments of inertias, serially linked together under Newton's gravity (g). The joints are frictionless and there are no applied/non-conservative external forces acting on the system. One of the ends of the first link is fixed in inertial space. The triple pendulum problem is solved using Lagrange, DAE, and Newton-Euler Methods. This section will go over general methodologies. For details, please refer to the deriver files.

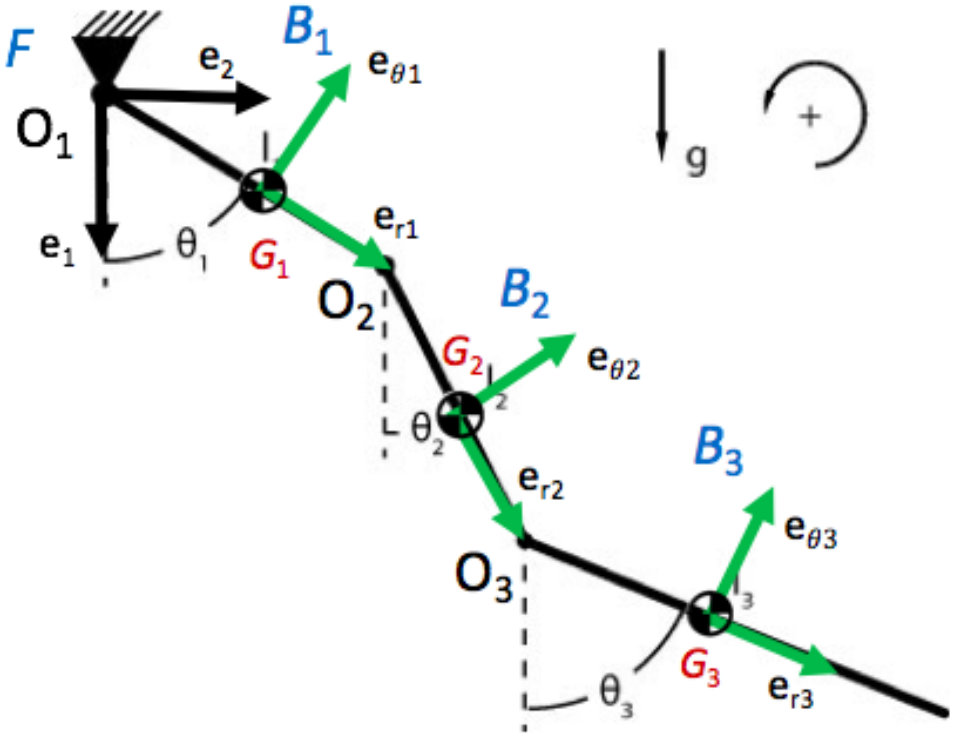

Figure 1. Triple Pendulum Schematic

2.1 Newton-Euler Method

The Newton-Euler method consists of performing angular momentum balance of the whole system about O_1, angular momentum balance of links 2 and 3 about O_2, and angular momentum balance of link 3 about O_3. First, the positions of the center of masses of each link relative to O_1, O_2, and O_3 are defined as follows:

$$r_{G_{1/O_1}} = \frac{l_1}{2} e_{r1}$$
$$r_{G_2/O_2} = \frac{l_2}{2} e_{r2}$$
$$r_{G_2/O_1} = l_1 e_{r1} + r_{G_2/O_2}$$
$$r_{G_3/O_3} = \frac{l_3}{2} e_{r3}$$
$$r_{G_3/O_2} = l_2 e_{r2} + r_{G_3/O_3}$$
$$r_{G_3/O_1} = l_1 e_{r1} + r_{G_3/O_2}$$

To transform from body coordinates into inertial coordinates, simply use a rotation matrix, which is a function of the angle of the respective link relative to the downward vertical. We can define angular velocity and angular acceleration of the links with respect to the inertia frame as:

$$\omega_{\beta_i/F} = \dot{\theta}_i e_3; \alpha_{\beta_i/F} = \ddot{\theta}_i e_3$$

They are intrinsic properties of each link, since each link is regarded as a rigid body. With these definitions, we can compute the accelerations of the center of mass of a link relative to the joints using the 5-term acceleration formula. Since the center of mass is fixed to a rigid body, the body derivatives will tend to 0. For example, the acceleration of the center of mass of the first link is:

$$a_{G_1/F} = \alpha_{\beta_1/F} \times r_{G_{1/O_1}} + \omega_{\beta_1/F} \times \omega_{1/F} \times r_{G_{1/O_1}}$$

The other accelerations can be calculated all in inertial frame the same way and using the separation principle of vectors, namely:

$$a_{G_2/F} = a_{O_2/F} + a_{G_2/O_2}$$
$$a_{G_3/F} = a_{O_2/F} + a_{O_3/O_2} + a_{G_3/O_3}$$

Next, angular momentum balance is conducted over O_1:

$$\sum M_{/O_1}^{(ext)} = \sum_{i=1}^{3} r_{G_i/O_1} \times m_i g e_1 = \sum_{i=1}^{3} I_{G_i}\, \alpha_{i/F} + r_{G_i/O_1} \times m_i\, a_{G_i/F}$$

Next, angular momentum balance is conducted over O_2 for links 2 and 3:

$$\sum M_{/O_2}^{(ext)} = \sum_{i=2}^{3} r_{G_i/O_2} \times m_i g e_1 = \sum_{i=2}^{3} I_{G_i}\, \alpha_{i/F} + r_{G_i/O_2} \times m_i\, a_{G_i/F}$$

Last but not least, angular momentum balance is conducted over O_3 for link 3:

$$M_{/O_3}^{(ext)} = r_{G_3/O_3} \times m_3 g e_1 = I_{G_3}\, \alpha_{3/F} + r_{3/O_3} \times m_3\, a_{3/F}$$

Note the acceleration terms used are accelerations with respect to the origin in inertial frame. Using the MATLAB jacobian method, $\ddot{\theta}_1, \ddot{\theta}_2, and\ \ddot{\theta}_3$ are solved symbolically. The expression is then simplified, converted into strings, and used to create a RHS file for ode45.

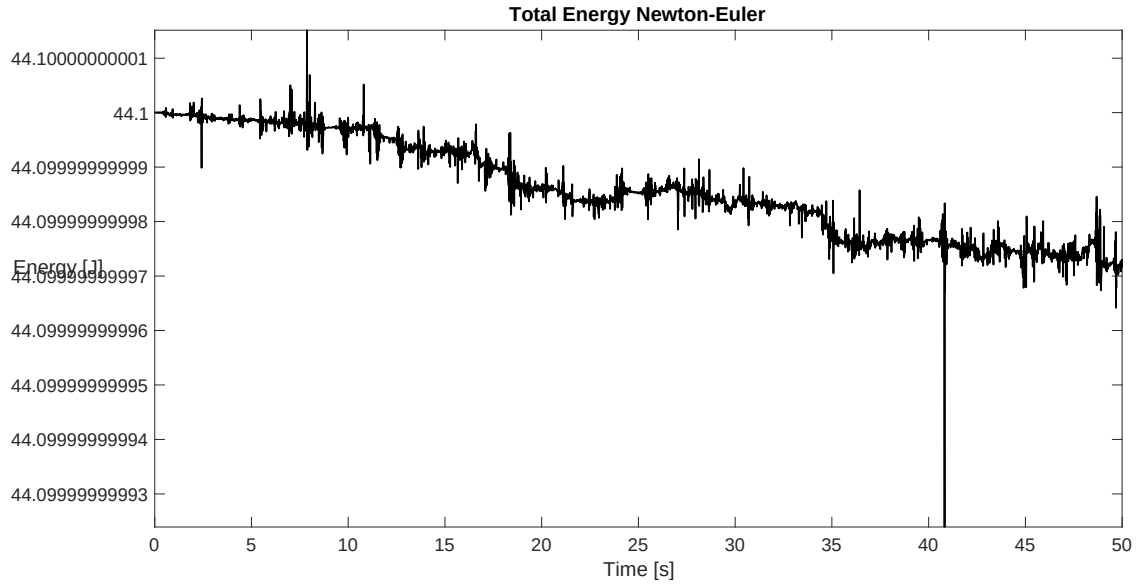

Figure 2. Newton-Euler method triple pendulum energy conservation

The tolerances for all three methods are ran at 1e-13. The total energy of the system is defined as:

$$E_{tot} = KE_{tot} + PE_{tot} = \sum_{i=1}^{3} \frac{1}{2} m_i\, v_{G_i/F} \bullet\ v_{G_i/F} + \frac{1}{2} I_i \dot{\theta}_i^2 + m_i g (r_{G_i} \bullet e_1 - \sum_{j=1}^{i-1} l_j - \frac{l_i}{2})$$

As shown, the total energy of the system is conserved pretty well. Please refer to the READ_ME.txt file to run simulations. Please also refer to the animation video folder for pre-recorded animations.

2.2 Differential Algebraic Equations

For the DAE method, two linear momentum balances and one angular momentum balance about the center of mass are done for each link. Then two constraint equations are formed for each joint, ensuring the difference in accelerations between each overlapping joint is zero. The free body diagram of the triple pendulum is shown below:

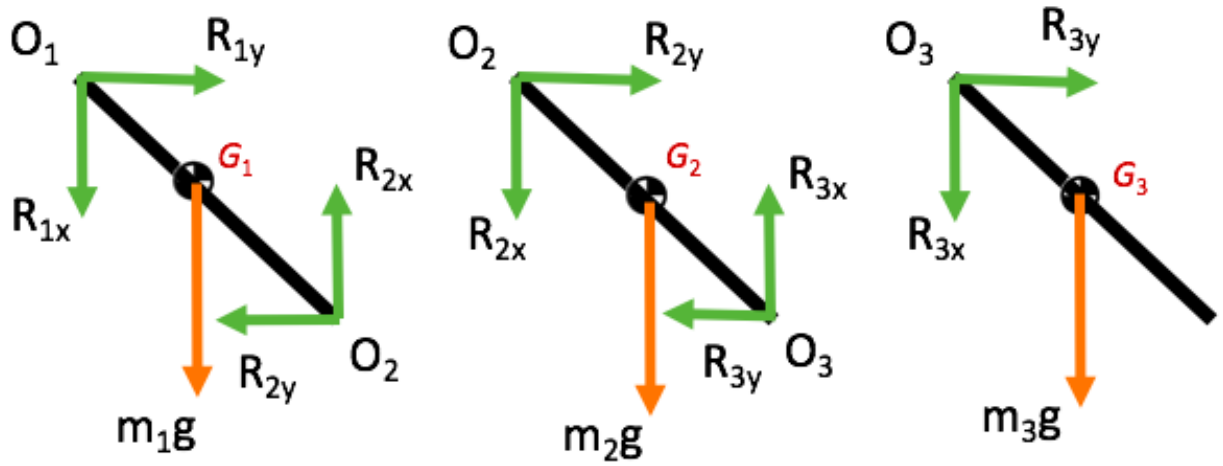

Figure 3. Triple pendulum free body diagrams

The position vectors are obtained the same way as Newton-Euler method. The equations of motion for Link 1 are:

$$F_{1x} = R_{1x} + m_1 g e_1 - R2_x$$
$$F_{1y} = R_{1y} - R_{2y}$$
$$M_{G_1} = \frac{l_1}{2} e_{r1} \times \begin{bmatrix} -R_{2x} \\ -R_{2y} \\ 0 \end{bmatrix}_F - \frac{l_1}{2} e_{r1} \times \begin{bmatrix} R_{1x} \\ R_{1y} \\ 0 \end{bmatrix}_F = I_{G_1} \alpha_{\beta_1/F}$$

Equations of motion for the other two links can be derived the same way, resulting in 9 total equations so far. Next, the constraint equations are derived. The goal is to express the location of the joints in terms of locations of center of masses and relative positions of the joints with respect to the center of masses, take two time derivatives, and set overlapping joint accelerations equal. For example, O_1 constraints can be derived this way:

$$\ddot{x}_{G_1} = \frac{l_1}{2} \left(-\ddot{\theta}_1 \sin(\theta_1) - \dot{\theta}_1^2 \cos(\theta_1) \right)$$
$$\ddot{y}_{G_1} = \frac{l_1}{2} \left(\ddot{\theta}_1 \cos(\theta_1) - \dot{\theta}_1^2 \sin(\theta_1) \right)$$

The other two joint constraints can be derived the same way. This brings the total number of equations for the DAE method to 15 equations. Using the jacobian function, the center of mass

accelerations and angular accelerations were solved symbolically, which were fed into a RHS file for ode45.

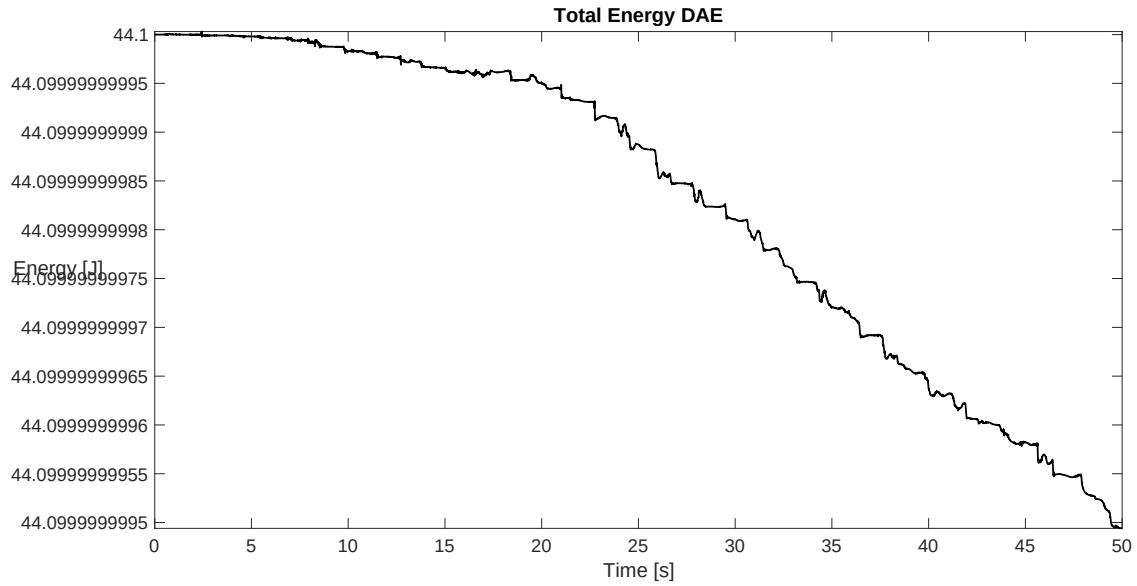

Figure 4. DAE method triple pendulum energy conservation

As shown, the energy conservation for the DAE method is pretty well.

2.3 Lagrange Method

I was able to create an n-linked pendulum deriver using the Lagrange method. The basic formulation is:

$$\frac{d}{dt}\frac{\delta L}{\delta \dot{q}_i} - \frac{\delta L}{\delta q_i} = Q_i = 0$$

where q is the set of minimal coordinates (in this case θ) capable of describing the system and L is the Lagrangian, defined as:

$$L = KE_{tot} - PE_{tot}$$

The total kinetic energy and potential energy of the system are calculated as follows:

$$KE_{tot} = \sum_{i=1}^{n} \frac{1}{2} m_i \, v_{G_i/F} \bullet \, v_{G_i/F} + \frac{1}{2} I_i \dot{\theta}_i^2$$

$$PE_{tot} = \sum_{i=1}^{n} m_i g (r_{G_i} \bullet e_1 - \sum_{j=1}^{i-1} l_j - \frac{l_i}{2})$$

Using MATLAB symbolic differentiation toolbox, $\frac{\delta L}{\delta q_i}$ and $\frac{\delta L}{\delta \dot{q}_i}$ can be calculated easily. The challenge is to calculate $\frac{d}{dt}\frac{\delta L}{\delta \dot{q}_i}$ symbolically, which can be performed using multivariable chain rule:

$$\text{Let } \frac{\delta L}{\delta \dot{q}_i} = f(q_i, \dot{q}_i)$$
$$\frac{d}{dt} f(q_i, \dot{q}_i) = \frac{\delta f(q_i, \dot{q}_i)}{\delta q_i}\dot{q}_i + \frac{\delta f(q_i, \dot{q}_i)}{\delta \dot{q}_i}\ddot{q}_i$$

From this, we can solve for matrix A and vector b using jacobian method such that:

$$A\ddot{\boldsymbol{q}} = b$$

For large n, it would be too computationally intensive to invert the matrix A to solve for angular accelerations. As a remedy, I transported symbolic A and b directly into the RHS file and invert A inside RHS. While the RHS file can be large, inverting a *numeric* A matrix for large n is much easier than a symbolic one. This demonstrates also the disadvantage of the DAE method. Since the reaction forces must be solved symbolically along with the accelerations, the DAE method becomes computationally intractable for large n. I was able to generate a 100-linked pendulum animation. Please refer to the animation video's folder for the actual animation, as it will take too long to simulate the 100-linked pendulum. For reference, it took more than 5 hours to derive the 100-linked pendulum RHS file.

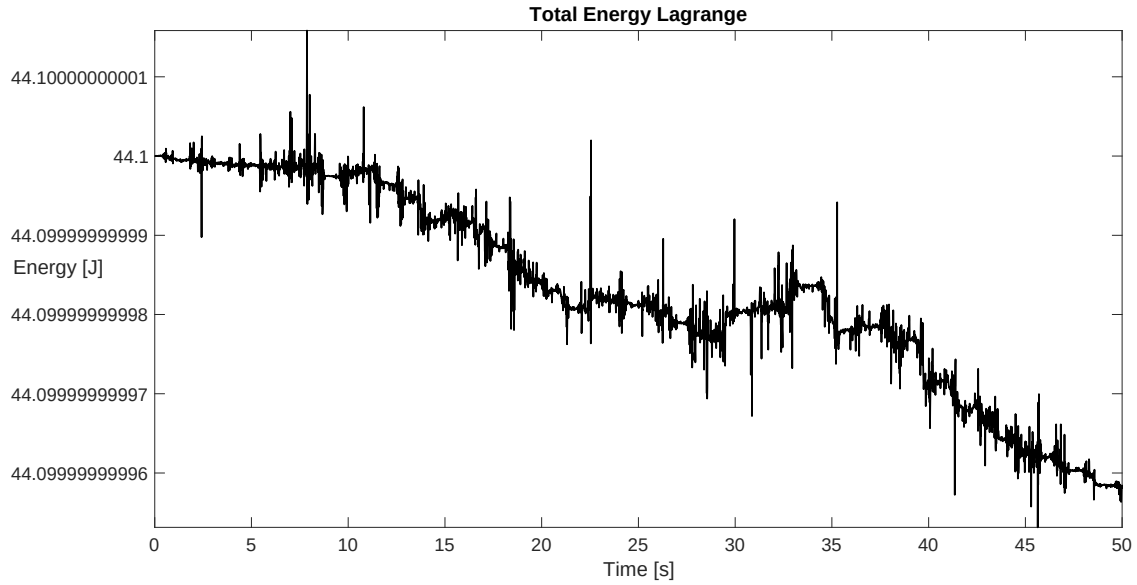

Figure 5. Lagrange method triple pendulum energy conservation

As shown, the Lagrange method conserved total energy pretty well.

2.4 Data Analysis

This section compares the solution of the triple pendulum among the three different methods with fixed initial conditions and solver tolerances.

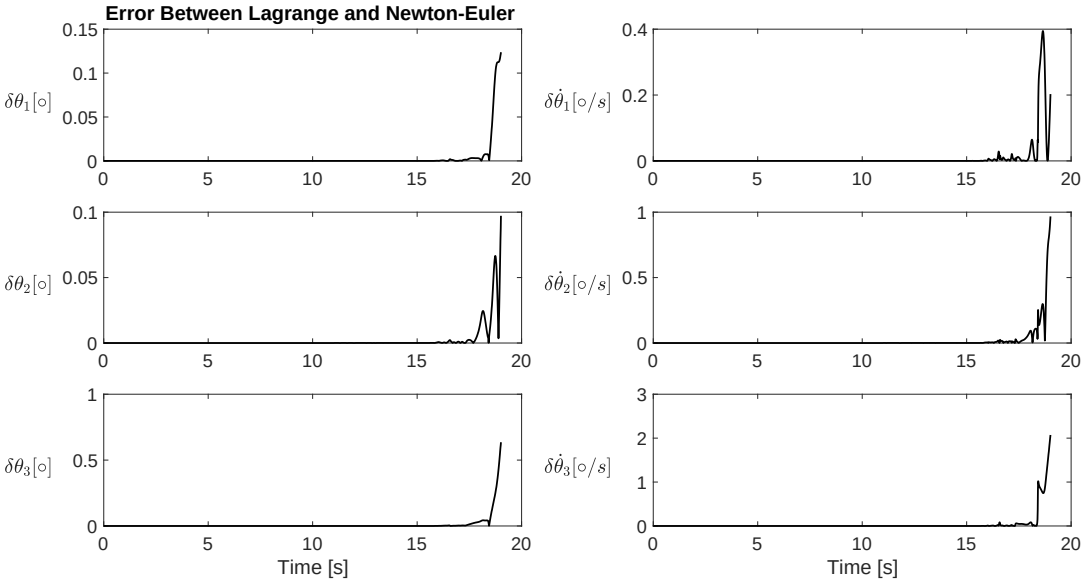

Figure 6. Absolute error between Lagrange and Newton-Euler methods

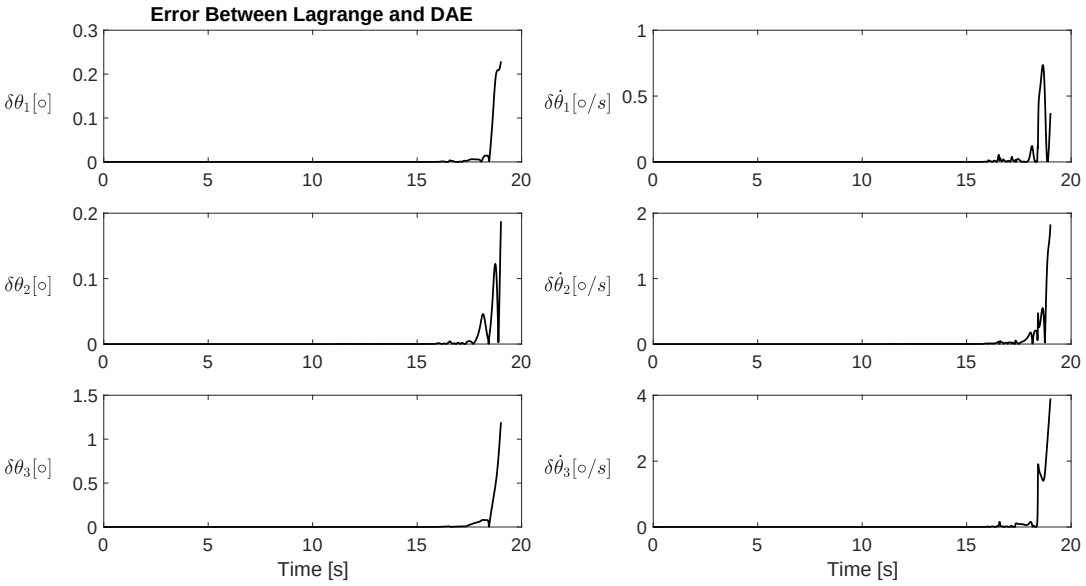

Figure 7. Absolute error between Lagrange and DAE methods

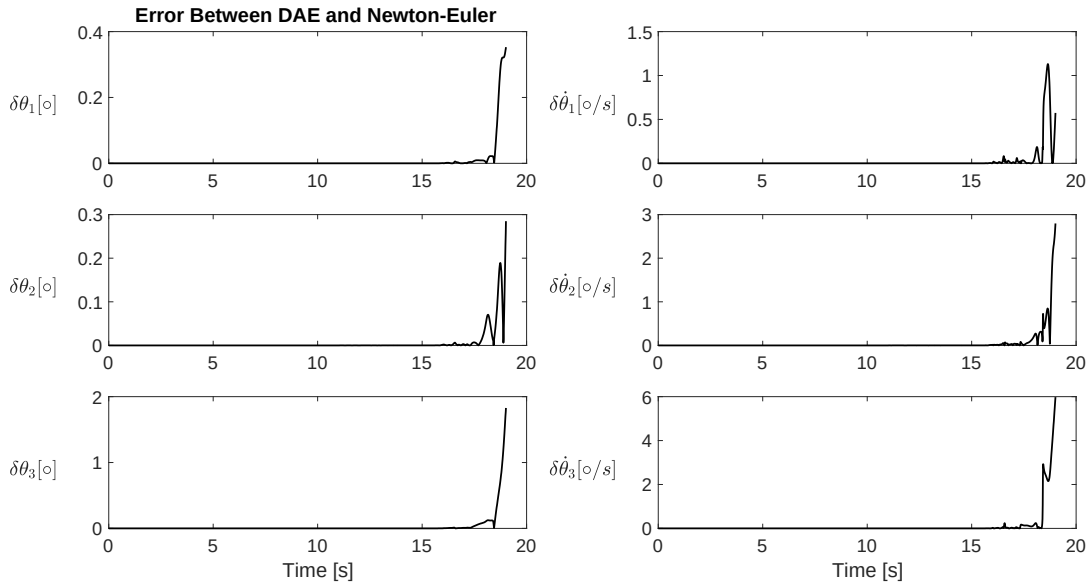

Figure 8. Absolute error between DAE and Newton-Euler methods

As shown, the solutions begin to diverge after 17 seconds. Since the triple pendulum Is highly chaotic, any numerical error initially will cause the solution to diverge greatly after some time. This is shown by increasing the integration tolerance, which causes the "band of convergence" to decrease. It seems the Newton-Euler method and Lagrange method are the most similar, while the DAE method and Newton-Euler method have the largest difference. However, all three methods converge quite well from 0 to 15 seconds.

3. Four-Bar Linkage

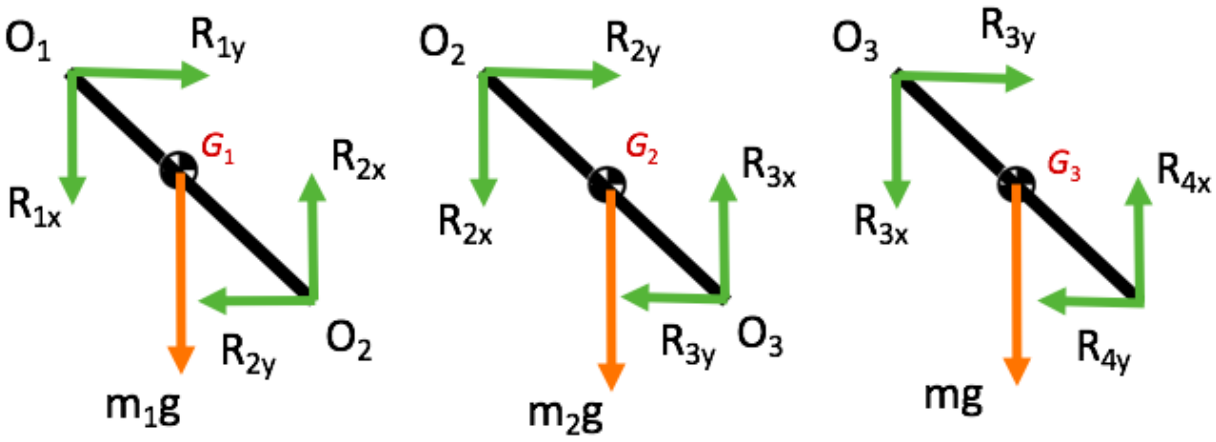

Figure 9. 4-Bar linkage free body diagrams

The four-bar linkage RHS is derived using the DAE method. It is a very natural extension to the DAE deriver used for the triple pendulum. The goal is to apply an acceleration constraint at the end of the triple pendulum by adding two reaction forces. Since there are two more variables, two more constraint equations need to be generated:

$$\ddot{x}_{G_3} = -\frac{l_3}{2}\left(-\ddot{\theta}_3 \sin(\theta_3) - \dot{\theta}_3^2 \cos(\theta_3)\right)$$
$$\ddot{y}_{G_3} = -\frac{l_3}{2}\left(\ddot{\theta}_3 \cos(\theta_3) - \dot{\theta}_3^2 \sin(\theta_3)\right)$$

The position and angular accelerations can then be solved using the jacobian command and imported into an RHS file.

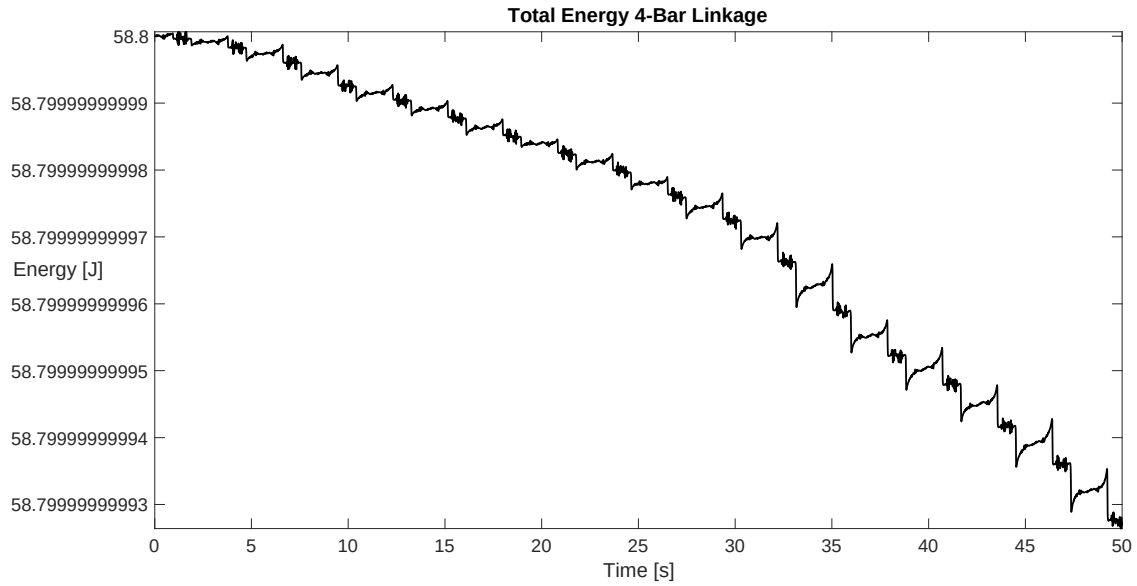

Figure 10. DAE method 4-Bar linkage conservation of energy

As shown, the 4-Bar linkage simulating using DAE method conserves energy pretty well.

4. Periodic Solutions

Periodic solutions of the triple and quadruple pendulum were found using fmincon.m, which attempts to solve the following optimization problem:

$$\min_z \|z(1:6) - f(z(1:6), z(7))\|$$
$$s.t. \quad x_0 = z(1:6)$$
$$Period = z(7)$$

where the function f(x_0, T) simulates the triple pendulum with initial conditions of x_0 and outputs the state at time T. This objective function is the error norm between the initial state and the state at time T. If the cost is low enough, it is possible to find a periodic solution. I found it necessary to use at least 1e-12 for both integration and solver tolerances.

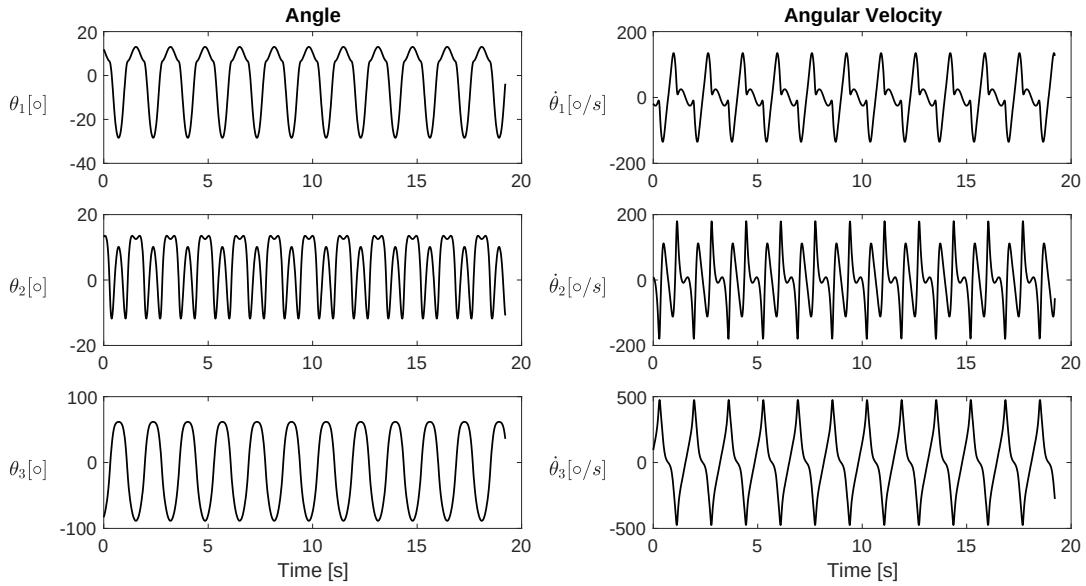

Figure 11. Triple pendulum periodic solution

The above solution was generated with the following initial conditions (units are in rad and rad/s):

$$x_0 = \begin{bmatrix} \theta_{1_0} \\ \theta_{2_0} \\ \theta_{3_0} \\ \dot{\theta}_{1_0} \\ \dot{\theta}_{2_0} \\ \dot{\theta}_{3_0} \end{bmatrix} = \begin{bmatrix} 0.207821287890159 \\ 0.230046723844060 \\ -1.448527525250897 \\ -0.323636324707770 \\ 0.146094757260416 \\ 1.655339043727058 \end{bmatrix}; l = 1; m = 1; I_G = \frac{1}{12}; g = 9.8$$

To see the animation, please look at "Lagrange_Pendulum_3_Periodic" in the animation videos folder. Additionally, I found a triple pendulum in a normal mode while swinging back and forth like a single pendulum, using the following initial conditions. Please refer to "Lagrange_Pendulum_3_2_Periodic" if interested in the animation:

$$\begin{bmatrix} 0.079855812687088 \\ 0.518084913559499 \\ 0.284825068342415 \\ -1.489804015361900 \\ 0.110953686219146 \\ -2.925278615817591 \end{bmatrix}; l = 1; m = 1; I_G = \frac{1}{12}; g = 9.8$$

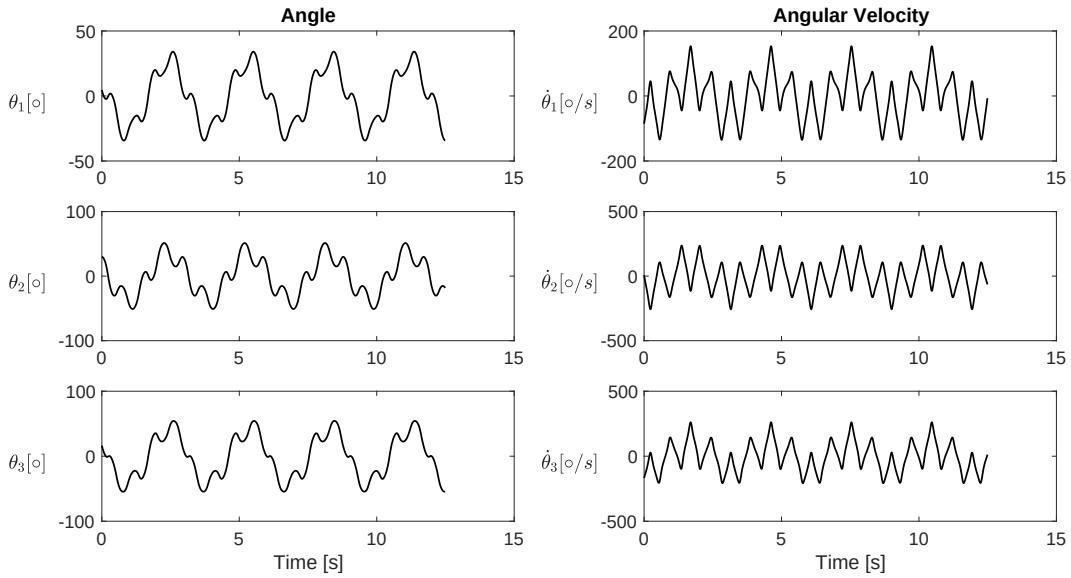

Figure 12. Triple pendulum like a single pendulum while in normal mode

I was also able to find two normal mode solutions of a quadruple pendulum using the following initial conditions:

$$\begin{bmatrix} -0.029111726332266 \\ -0.016045163432762 \\ 0.022495297315480 \\ 0.110974733413308 \\ 0.370089523425431 \\ 0.214148128979260 \\ -0.301259983532510 \\ -1.411248973186850 \end{bmatrix}; \begin{bmatrix} -0.005486776206907 \\ 0.002813551901407 \\ 0.010723510070385 \\ -0.015650651510557 \\ 0.279933154790084 \\ -0.143699045708179 \\ -0.546710497761539 \\ 0.798011124562596 \end{bmatrix}; l = 1; m = 1; I_G = \frac{1}{12}; g = 9.8$$

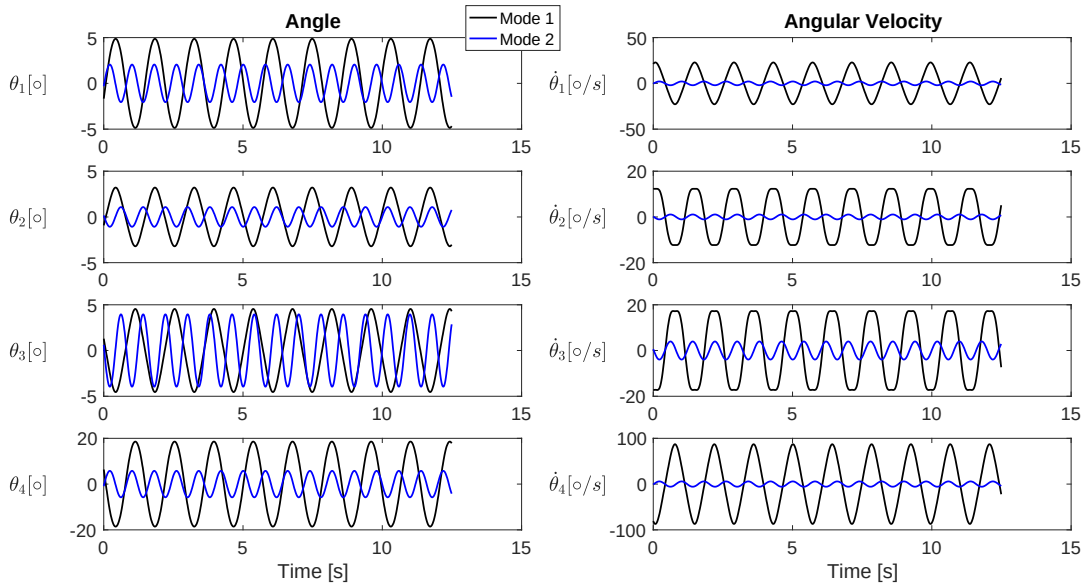

Figure 13. Quadruple pendulum modes

5. Conclusion

Equations of motion for the triple pendulum were derived and simulated using DAE, Newton-Euler, and Lagrange. The 4-bar linkage was derived and simulated using DAE. An n-linked pendulum deriver was developed using Lagrange. Periodic solutions of triple and quadruple pendulums were found using fmincon.

6. MATLAB Functions

• animatePendulum(): Produces animation for Newton-Euler and Lagrange generated data.
• animatePendulumDAE(): Produces animation for DAE generated data only
• createFunction4BarLinkage(): Derives RHS for 4-Bar Linkage using DAE.
• createFunctionPendulum3_DAE(): Derives RHS for triple pendulum using DAE.
• createFunctionPendulum3_NE(): Derives RHS for triple pendulum using Newton-Euler.
• createFunctionPendulumN_Lagrange(): Derives RHS for n-linked pendulum using Lagrange.
• energyPendulum(): calculates the potential and kinetic energy of the whole system
• errorNorm(): objective function used to find periodic solutions.
• **MAE5730_Final_Project()**: the main file containing code to run simulations. Refer to READ_ME.txt for instructions on how to use it.
• thetas2pos(): calculates the end positions of each link based on the angles