MAE 5730: Intermediate Dynamics and Vibrations
Final Project Report

Brian Lui (bl569)

Decemeber 9, 2018

1 Introduction

This report discusses the process, results, and reflections from animating a triple pendulum and a four bar linkage. The derivations can be done by hand if done very carefully, but it is not recommended because of the number of terms involved and the inability to generalize the scenario.

There were several very useful MATLAB functions that were used to do this project. These functions are

- simplify(A)- simplifies A (which is full of symbolic terms)
- \- Can use to solve Ax=b. x = A \b
- equationsToMatrix (eqns, unknowns) - This takes in a list of equations (eqns) and a list of the unknowns and outputs the corresponding A and b matrix. The A and b matrix satisfies Ax = b, in which x is eqns.
- matlabFunction(A, 'file', 'fileName')- This outputs a function that derives A symbolically. The output of this is used in ode45.

The derive files generate two files that are used within the ode45 function in the plotting script. These files are rhsStuffMassMatrixZ.m and rhsStuffbVectorZ.m, in which Z is amb, Lagrange, or DAE based on which method is used to derive the equations of motion. These files generate the equations and unknowns will be used to build a matrix of the form Ax = b, in which x is the unknowns, and A is the mass matrix. The unknowns can be solved by using the \function in Matlab (A \b).

2 Triple Pendulum

The triple pendulum was derived three different ways: Using Newton Euler (NE), Lagrange equations, and differential algebraic equations (DAE). Each method was done symbolically in MATLAB. They were checked for accuracy by comparing the output with each other. The lengths of the triple pendulum are given by Figure 1, and the coordinate frames used in the triple pendulum are given by Figure 2.

2.1 Newton-Euler

2.1.1 Free Body Diagram

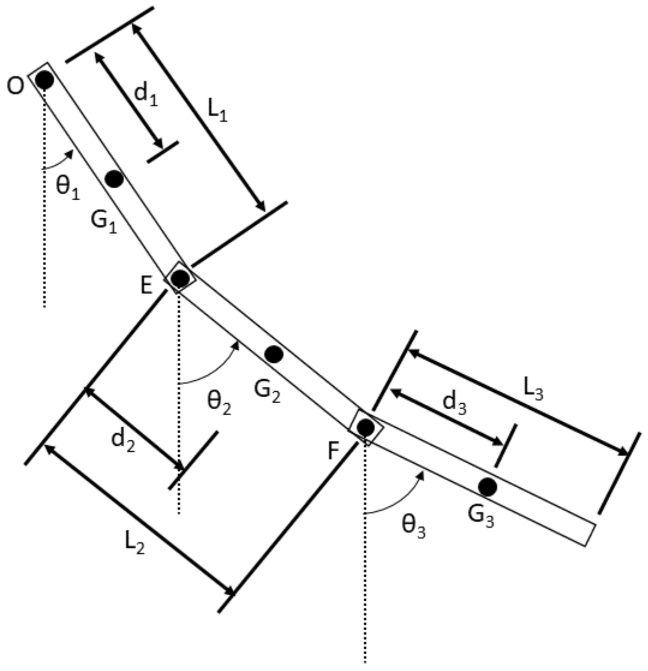

Figure 1: The relevant points and lengths of the bars of the triple pendulum.

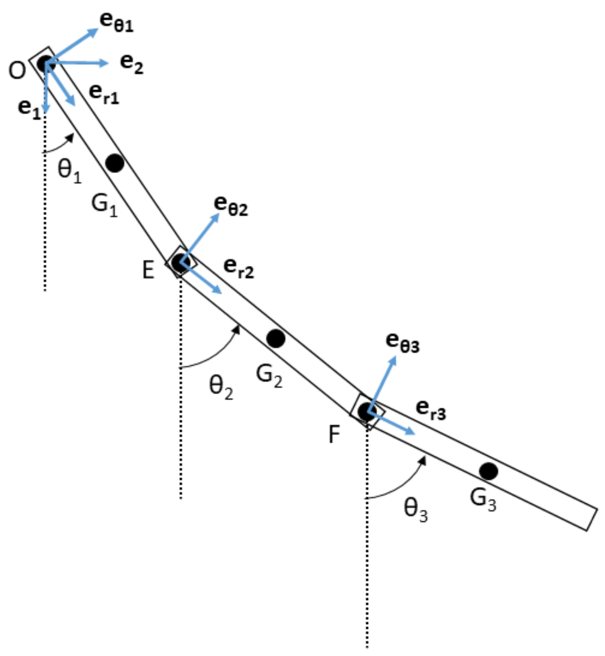

Figure 2: The coordinate frames used in the triple pendulum.

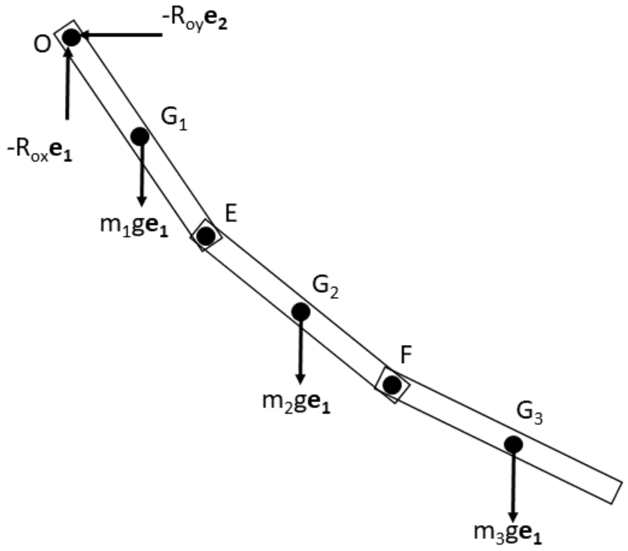

Figure 3: Forces on all three bars of the triple pendulum.

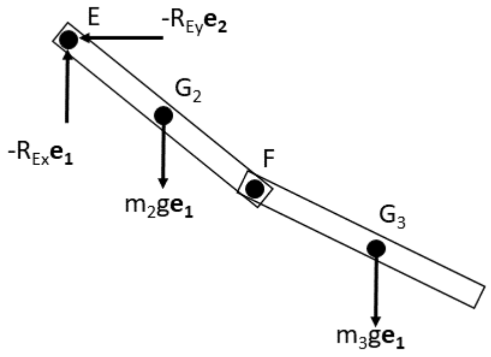

Figure 4: Forces on the bottom two bars of the triple pendulum.

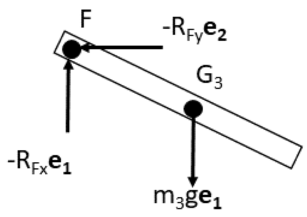

Figure 5: Forces on the last bar of the triple pendulum.

2.1.2 Derivation

Angular momentum balance ($\sum \mathbf{M} = \dot{\mathbf{H}}$) was taken with respect to points O, E, and F in the FBD given by Figures 3, 4, and 5 respectively. AMB was taken about these points so that the reaction force can be ignored; this derivation is a minimal coordinates derivation.

Angular momentum balance about O:

$$\sum \mathbf{M_{/O}} = \dot{\mathbf{H}}_{\mathbf{/O}}$$

$$\sum_{i=1}^{3} \mathbf{r_{Gi/O}} \times mg\mathbf{e_1} = \sum_{i=1}^{3} m_i \mathbf{r_{Gi/O}} \times \mathbf{a_{Gi/O}} + I_i^G \ddot{\theta}_i \mathbf{e_3}$$

Angular momentum balance about E:

$$\sum \mathbf{M_{/E}} = \dot{\mathbf{H}}_{\mathbf{/E}}$$

$$\sum_{i=2}^{3} \mathbf{r_{Gi/E}} \times mg\mathbf{e_1} = \sum_{i=2}^{3} m_i \mathbf{r_{Gi/E}} \times \mathbf{a_{Gi/O}} + I_i^G \ddot{\theta}_i \mathbf{e_3}$$

Angular momentum balance about F:

$$\sum \mathbf{M_{/F}} = \dot{\mathbf{H}}_{\mathbf{/F}}$$

$$\mathbf{r_{G3/F}} \times mg\mathbf{e_1} = m_3 \mathbf{r_{G3/F}} \times \mathbf{a_{G3/O}} + I_3^G \ddot{\theta}_3 \mathbf{e_3}$$

2.2 Lagrange

The free bodies diagrams used are the same as in NE (Figures 3, 4, 5). However, note that the derivation with Lagrange does not actually require the free body diagram.

2.2.1 Derivation

The Lagragian is defined as:

$$L = \sum_{i=1}^{3} E_K i - \sum_{i=1}^{3} E_P i$$

in which $E_K i$ and $E_P i$ is kinetic and potential energy of each bar. For each bar, the kinetic energy is defined as

$$E_{Ki} = \frac{1}{2} m |\mathbf{v_{Gi/O}}|^2 + \frac{1}{2} I_i \theta_i^2$$

and the kinetic energy is defined as

$$E_{Pi} = mgh_{Gi/O}$$

in which $h_{Gi/O}$ is the distance of bar i's center of mass relative to a fixed point (such as O).

The Lagrange equations is then given by

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

The three equations that result from the Lagrange equation is used to find $\ddot{\theta}_1$, $\ddot{\theta}_2$, and $\ddot{\theta}_3$.

2.3 DAE

2.3.1 Free Body Diagram

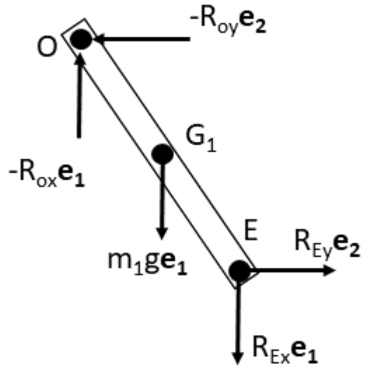

Figure 6: Forces on the first arm of the triple pendulum, DAE.

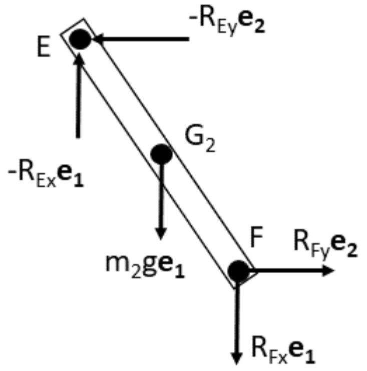

Figure 7: Forces on the second arm of the triple pendulum, DAE.

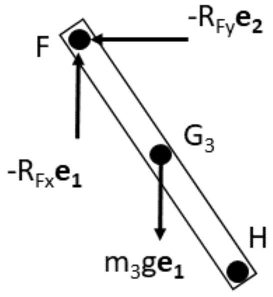

Figure 8: Forces on the third arm of the triple pendulum, DAE.

2.3.2 Derivation

This method is a maximal coordinate method; it requires writing all the possible independent equations to solve for all the unknowns in the problem. As defined, there are 15 unknowns in the system: $\ddot{x}_1$, $\ddot{y}_1$, $\ddot{\theta}_1$, $\ddot{x}_2$, $\ddot{y}_2$, $\ddot{\theta}_2$, $\ddot{x}_3$, $\ddot{y}_3$, $\ddot{\theta}_3$, R_{Ox}, R_{Oy}, R_{Ex}, R_{Ey}, R_{Fx}, and R_{Fy}. For i=1,2,3, $\ddot{x}_i$ and $\ddot{y}_i$ is the acceleration of point G_i on the FBDs; they are the accelerations of the center of mass of each bar. $\mathbf{R_O}$, $\mathbf{R_E}$, and $\mathbf{R_F}$ are the reaction force at the hinges of the triple pendulum, and their components are R_{Ox}, R_{Oy}, R_{Ex}, R_{Ey}, R_{Fx}, and R_{Fy} respectively.

The relevant equations to get a full rank matrix are as followed:

- LMB- 6 equations, 2 from each direction for each bar
- AMB- 3 equations, 1 for each bar
- Constraint equations- 6 equations, 2 each at points O,E,and F.

Linear Momentum Balance
On the FBD from Figure 6 (Bar OE):

$$\sum \mathbf{F} = m\mathbf{a_{G1/O}}$$

$$m_1 g\hat{\mathbf{e_1}} - R_{OX}\hat{\mathbf{e_1}} - R_{OY}\hat{\mathbf{e_2}} + R_{EX}\hat{\mathbf{e_1}} + R_{EY}\hat{\mathbf{e_2}} = m_1\mathbf{a_{G1/O}}$$

On the FBD from Figure 7 (Bar EF):

$$\sum \mathbf{F} = m_2\mathbf{a_{G2/O}} \tag{1}$$

$$m_2 g\hat{\mathbf{e_1}} - R_{EX}\hat{\mathbf{e_1}} - R_{EY}\hat{\mathbf{e_2}} + R_{FX}\hat{\mathbf{e_1}} + R_{FY}\hat{\mathbf{e_2}} = m_2\mathbf{a_{G2/O}}$$

On the FBD from Figure 8 (Bar FH):

$$\sum \mathbf{F} = m_3\mathbf{a_{G3/O}} \tag{2}$$

$$m_3 g\hat{\mathbf{e_1}} - R_{FX}\hat{\mathbf{e_1}} - R_{FY}\hat{\mathbf{e_2}} = m_3\mathbf{a_{G3/O}}$$

Angular Momentum Balance
Angular momentum balance ($\sum \mathbf{M} = \dot{\mathbf{H}}$) was taken with respect to points G1, G2, and G3 in the FBD given by Figures 6, 7, and 8 respectively.

Angular momentum balance about O:

$$\sum \mathbf{M_{/G1}} = \dot{\mathbf{H}}_{\mathbf{/G1}}$$

$$\mathbf{r_{O/G1}} \times -\mathbf{R_O} + \mathbf{r_{E/G1}} \times \mathbf{R_E} = I_1^G \ddot{\theta}_1 \mathbf{e_3}$$

Angular momentum balance about G2:

$$\sum \mathbf{M}_{/\mathbf{G2}} = \dot{\mathbf{H}}_{/\mathbf{G2}}$$

$$\mathbf{r}_{\mathbf{E/G2}} \times -\mathbf{R}_{\mathbf{E}} + \mathbf{r}_{\mathbf{F/G2}} \times \mathbf{R}_{\mathbf{F}} = I_2^G \ddot{\theta}_2 \mathbf{e_3}$$

Angular momentum balance about G3:

$$\sum \mathbf{M}_{/\mathbf{G3}} = \dot{\mathbf{H}}_{/\mathbf{G3}}$$

$$\mathbf{r}_{\mathbf{F/G3}} \times -\mathbf{R}_{\mathbf{F}} = I_1^G \ddot{\theta}_3 \mathbf{e_3}$$

Constraint Equations
The constraint equations are that the ends of the bars are constrained or are equal to the acceleration of the bar it is attached to. Specifically, the acceleration of point O is equal to 0, the acceleration of point E is equal to the acceleration in bar OE's frame as well as the acceleration in bar EF's frame, and the acceleration of point F is equal to the acceleration in bar EF's frame as well as the acceleration in bar FH's frame.

Point O:

$$\mathbf{a_O} = \mathbf{a_O}$$

$$\ddot{x_1}\mathbf{e_3} + \mathbf{a_{O/G1}} = \mathbf{0} \tag{3}$$

Point E:

$$\mathbf{a_E} = \mathbf{a_E}$$

$$\ddot{x_1}\mathbf{e_3} + \mathbf{a_{E/G1}} = \ddot{x_2}\mathbf{e_3} - \mathbf{a_{E/G2}} \tag{4}$$

Point F:

$$\mathbf{a_F} = \mathbf{a_F}$$

$$\ddot{x_2}\mathbf{e_3} + \mathbf{a_{F/G2}} = \ddot{x_3}\mathbf{e_3} - \mathbf{a_{F/G3}} \tag{5}$$

2.4 Sample Output

Three different initial conditions of the triple pendulum is plotted here:

Case 1: $\theta_1 = \frac{\pi}{2}$, $\theta_2 = \pi$, $\theta_3 = \pi$
Case 2: $\theta_1 = \frac{\pi}{4}$, $\theta_2 = \frac{\pi}{4}$, $\theta_3 = 3\pi 4$
Case 3: $\theta_1 = \frac{\pi}{4}$, $\theta_2 = \frac{\pi}{3}$, $\theta_3 = \pi 3$

The plots of the end of the triple pendulum from these three conditions can be seen in Figures 9, 10, and 11. It shows that the system is chaotic: slight differences in the initial condition results in very different behavior of the triple pendulum.

To check for accuracy, the x position from each simulation is compared with each other as a function of time. The difference between the three methods is very small and is likely only because of numerical integration error, and can be seen in Figures 12 and 13.

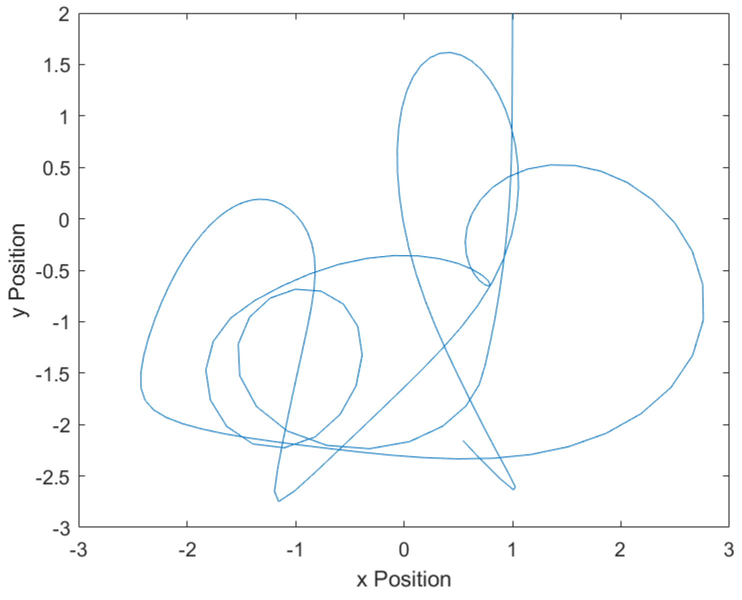

Figure 9: The end of the triple pendulum (H) for $\theta_1 = \frac{\pi}{2}$, $\theta_2 = \pi$, $\theta_3 = \pi$

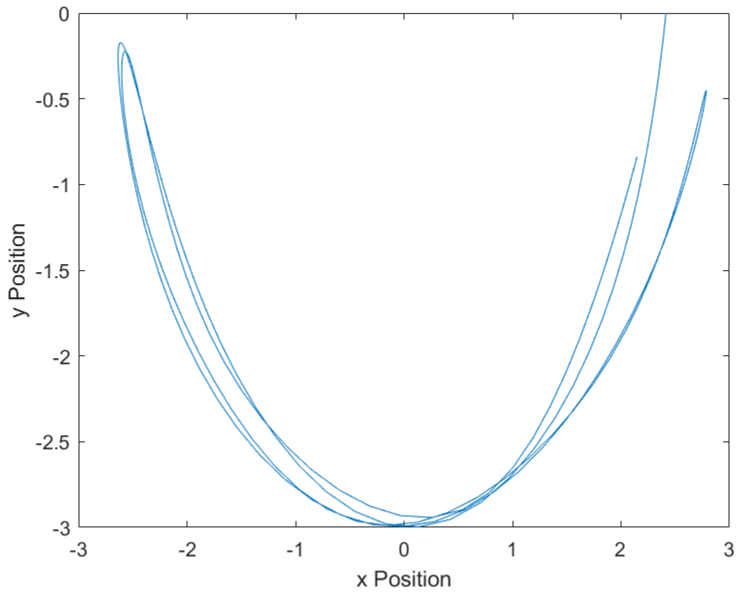

Figure 10: The end of the triple pendulum (H) for $\theta_1 = \frac{\pi}{4}$, $\theta_2 = \frac{\pi}{2}$, $\theta_3 = \frac{3\pi}{4}$

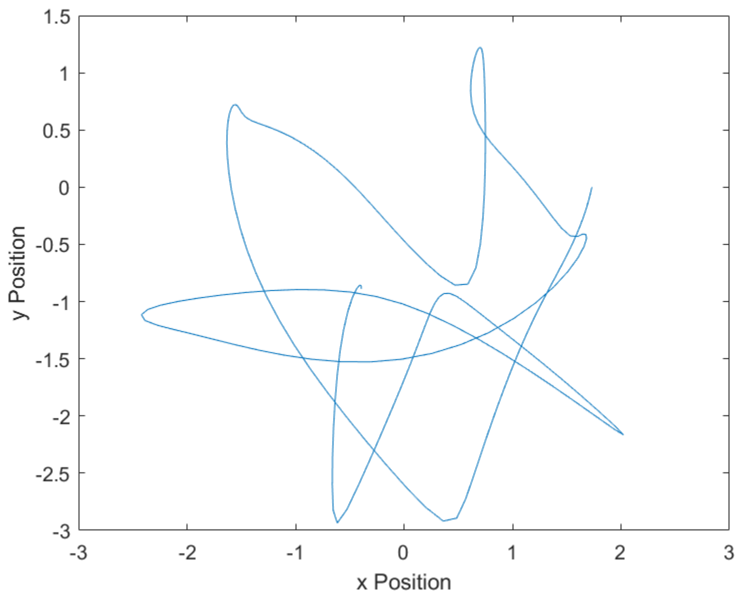

Figure 11: The end of the triple pendulum (H) for $\theta_1 = \pi$, $\theta_2 = \frac{\pi}{3}$, $\theta_3 = \frac{\pi}{3}$

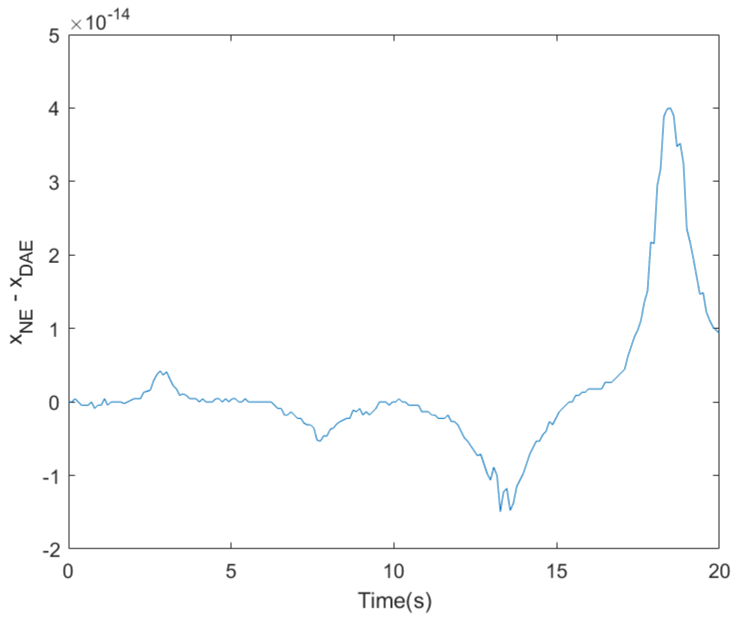

Figure 12: The difference in the x position between the Newton-Euler and DAE method

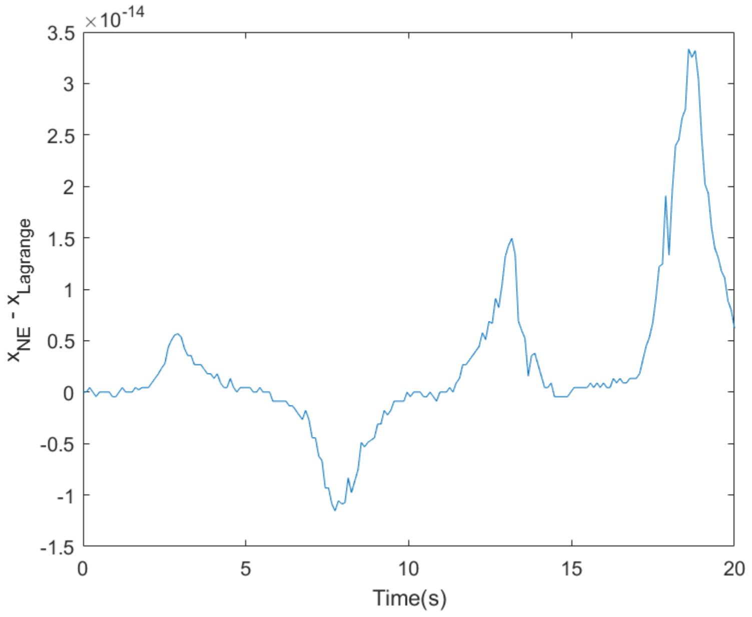

Figure 13: The difference in the x position between the Newton-Euler and Lagrange

3 Four Bar Linkage

The four bar linkage is essentially a triple pendulum with an additional constraint force restricting the motion of the other end of the triple pendulum. This can be seen in Figure 14. This problem was solved using the DAE method: additional constraint forces were added to account for point H being stationary. There are now two additional unknowns on the system, $\mathbf{R_H} = R_{HX}\hat{\mathbf{e_1}}$ and $R_{HY}\hat{\mathbf{e_2}}$. Now, there are 17 equations and 17 unknowns in the matrix to solve. The two additional equations required to make the mass matrix rank sufficient come from the additional stationary constraint at H.

3.1 Free Body Diagrams

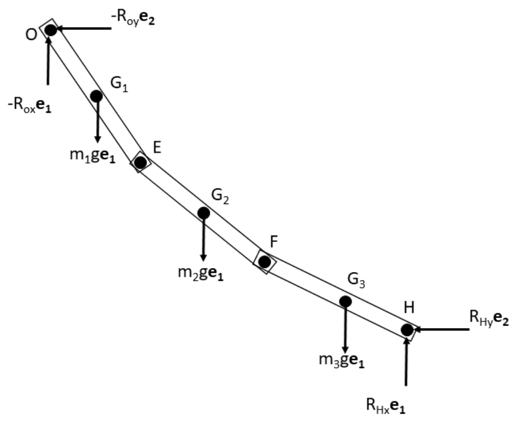

Figure 14: Forces on the 4-bar linkage.

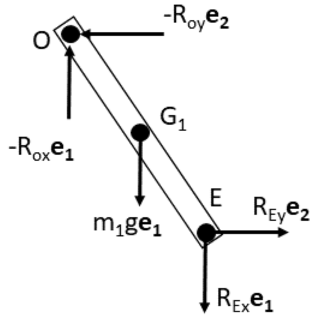

Figure 15: Forces on the 4-bar linkage.

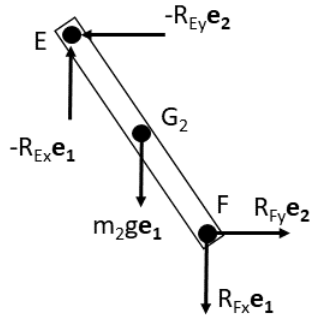

Figure 16: Forces on the 4-bar linkage.

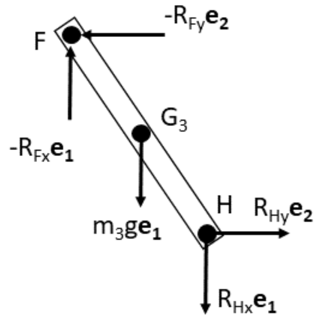

Figure 17: Forces on the 4-bar linkage.

3.2 Derivation

Linear Momentum Balance
The linear momentum balance is equivalent for bars OE and EF. It has an additional constraint force at H in bar FH.

On the FBD from Figure 15 (Bar OE):

$$\sum \mathbf{F} = m\mathbf{a_{G1/O}}$$

$$m_1 g\hat{\mathbf{e_1}} - R_{OX}\hat{\mathbf{e_1}} - R_{OY}\hat{\mathbf{e_2}} + R_{EX}\hat{\mathbf{e_1}} + R_{EY}\hat{\mathbf{e_2}} = m_1\mathbf{a_{G1/O}}$$

On the FBD from Figure 16 (Bar EF):

$$\sum \mathbf{F} = m_2\mathbf{a_{G2/O}} \tag{6}$$

$$m_2 g\hat{\mathbf{e_1}} - R_{EX}\hat{\mathbf{e_1}} - R_{EY}\hat{\mathbf{e_2}} + R_{FX}\hat{\mathbf{e_1}} + R_{FY}\hat{\mathbf{e_2}} = m_2\mathbf{a_{G2/O}}$$

On the FBD from Figure 17 (Bar FH):

$$\sum \mathbf{F} = m_3\mathbf{a_{G3/O}} \tag{7}$$

$$m_3 g\hat{\mathbf{e_1}} - R_{FX}\hat{\mathbf{e_1}} - R_{FY}\hat{\mathbf{e_2}} + R_{HX}\hat{\mathbf{e_1}} + R_{HY}\hat{\mathbf{e_2}} = m_3\mathbf{a_{G3/O}}$$

Angular Momentum Balance
Angular momentum balance ($\sum \mathbf{M} = \dot{\mathbf{H}}$) was taken with respect to points G1, G2, and G3 in the FBD given by Figures 15, 16, and 17 respectively. The difference between these equations and the triple pendulum is the additional constraint force in the angular momentum about G3.

Angular momentum balance about O:

$$\sum \mathbf{M_{/G1}} = \dot{\mathbf{H}}_{\mathbf{/G1}}$$

$$\mathbf{r_{O/G1}} \times -\mathbf{R_O} + \mathbf{r_{E/G1}} \times \mathbf{R_E} = I_1^G\ddot{\theta}_1\mathbf{e_3}$$

Angular momentum balance about G2:

$$\sum \mathbf{M_{/G2}} = \dot{\mathbf{H}}_{\mathbf{/G2}}$$

$$\mathbf{r_{E/G2}} \times -\mathbf{R_E} + \mathbf{r_{F/G2}} \times \mathbf{R_F} = I_2^G\ddot{\theta}_2\mathbf{e_3}$$

Angular momentum balance about G3:

$$\sum \mathbf{M_{/G3}} = \dot{\mathbf{H}}_{\mathbf{/G3}}$$

$$\mathbf{r_{F/G3}} \times -\mathbf{R_F} + \mathbf{r_{H/G3}} \times -\mathbf{R_H} = I_1^G\ddot{\theta}_3\mathbf{e_3}$$

Constraint Equations
The constraint equations are the same as that of the triple pendulum, except with the additional constrained point H.

Point O:

$$\mathbf{a_O} = \mathbf{a_O}$$

$$\ddot{x}_1\mathbf{e_3} + \mathbf{a_{O/G1}} = \mathbf{0} \tag{8}$$

Point E:

$$\mathbf{a_E} = \mathbf{a_E}$$

$$\ddot{x}_1\mathbf{e_3} + \mathbf{a_{E/G1}} = \ddot{x}_2\mathbf{e_3} - \mathbf{a_{E/G2}} \tag{9}$$

Point F:

$$\mathbf{a_F} = \mathbf{a_F}$$
$$\ddot{x_2}\mathbf{e_3} + \mathbf{a_{F/G2}} = \ddot{x_3}\mathbf{e_3} - \mathbf{a_{F/G3}} \tag{10}$$

Point H:

$$\mathbf{a_H} = \mathbf{a_H}$$
$$\ddot{x_3}\mathbf{e_3} + \mathbf{a_{H/G3}} = \mathbf{0} \tag{11}$$

4 Concluding Remarks

4.1 Checking the Animation

4.1.1 Looking at the Animation

The animation for known simplifications at specific initial conditions can be checked. For example, this involves looking at the cases in which the triple pendulum simplifies down to a double pendulum or a single pendulum, such as when L_2=L_3=d_2=d_3=0. Another example is for stationary cases of the four bar linkage when all the bars are in stable equilibrium, such as when all the bars are all hanging down, ie. θ_1=θ_2=θ_3=0.

4.1.2 Solving for Equations of Motion using Multiple Methods

This check was done for the triple pendulum by calculating the equations of motion and seeing how similar the motion looks. The graphs will not be exactly the same: there are some minor differences because of numerical integration error, but it should be on an orders of magnitude smaller than any reasonable errors in the simulation. Although possible, it is unlikely that the equations of motions derived from three different methods that result in the same animation are all wrong.

4.1.3 Energy Conservation

The energy at every timestep can be calculated and compared to see if it is conserved. One way of doing this is to calculate the total energy at every timestep. The total energy is given by

$$E = KE + PE$$

in which E is the total energy of the system, KE is the total kinetic energy of the system, and PE is the total potential energy of the system. For a triple pendulum and a four bar linkage, the KE of the system is given by

$$KE = \sum_{i=1}^{3} \frac{1}{2} m_i v_{Gi/O}^2 + \frac{1}{2} I_i \dot{\theta_i}^2$$

in which m_i is the mass of bar i, $v_{Gi/O}$ is the velocity of the center of mass of bar i, I_i is the moment of inertia of bari i, and $\dot{\theta_i}$ is the angle bar i makes with the vertical. Another way to think about this is the net kinetic energy is equal to the kinetic energy of each bar of the pendulum.

In a similar fashion, the PE of the system can be calculated, and is given by

$$PE = \sum_{i=1}^{3} m_i g h_{Gi}$$

in which h_{Gi} is the height of bar i's center of mass relative to a given reference point (such as point O).

One way of calculating the energy at every timestep would be to include the position (x_1, y_1, x_2, y_2, x_3, y_3) and the velocity of the center of masses ($\dot{x_1}$, $\dot{y_1}$, $\dot{x_2}$, $\dot{y_2}$, $\dot{x_3}$, $\dot{y_3}$) in the state that is passed into ode45. Doing so would allow all the relevant values to be outputted and the total energy can be calculated. For both the triple pendulum and the four bar linkage, energy should remain at a relatively constant level. It will not be exactly constant because of numerical errors that accumulate.

4.2 Miscellaneous

· Debugging the MATLAB was dreadful. Especially when the errors are always something minor, such as replacing a y for an x, or a 3 for a 2. Very hard to debug, and requires going through every line. There exists a symmetry to the linear momentum balance, angular momentum balance, and constraint equations, so a better way of deriving the equations of motion would be to automate it. This would also enable n-link pendulums to be generated more easily.

5 MATLAB Code

5.1 Deriving the triple pendulum EOM using NE: tPendDeriveEOMamb.m

```
%This derives the triple pendulum EOM for the double pendulum. This does it using NE- AMB.

syms d1 d2 d3 L1 L2 m1 I1 m2 I2 m3 I3 g real %parameters (Note d one, not d L)
syms theta1 theta2 theta3 theta1dot theta2dot theta3dot real %states
syms theta1doubledot theta2doubledot theta3doubledot real %states
syms e1 e2 e3 er1 etheta1 er2 etheta2 er3 etheta3 real %unit vectors
syms rG1relO aG1relO                           %relevant vectors
syms rG2relO aG2relO rG2relE aG2relE rErelO aErelO %relevant vectors
syms rG3relO aG3relO rG3relF aG3relF rFrelE aFrelE real %relveant vectors
syms rG3relE aG3relE real
syms MrelO MrelE MrelF H1relODot H2relODot H3relODot real

%defining unit vectors
e1 = [1 0 0]';
e2 = [0 1 0]';
e3 = cross(e1,e2);

er1 = cos(theta1)*e1 + sin(theta1)*e2;
etheta1 = cross(e3, er1);
er2 = cos(theta2)*e1 + sin(theta2)*e2;
etheta2 = cross(e3, er2);
er3 = cos(theta3)*e1 + sin(theta3)*e2;
etheta3 = cross(e3, er3);


%defining vectors of interest
%For the first bar
rG1relO = d1*er1;
aG1relO = d1*theta1doubledot *etheta1 - d1*theta1dot^2 * er1;

%For the second bar (forearm)
rG2relE = d2*er2;
rErelO = L1*er1;
rG2relO = rG2relE + rErelO;

aG2relE = d2*theta2doubledot*etheta2 - d2*theta2dot^2 * er2;
aErelO = L1*theta1doubledot*etheta1 - L1*theta1dot^2 * er1;
aG2relO = aG2relE + aErelO;

%For the third bar
rG3relF = d3*er3;
rFrelG3 = -rG3relF;

rFrelE = L2*er2;
rFrelO = rFrelE + rErelO;

rG3relE = rG3relF + rFrelE;
rG3relO = rG3relF + rFrelE + rErelO;

rFrelG2 = rFrelO - rG2relO;

aG3relF = d3*theta3doubledot*etheta3 - d3*theta3dot^2 * er3;
aFrelE = L2*theta2doubledot*etheta2 - L2*theta2dot^2 * er2;
aG3relE = aG3relF + aFrelE;
aG3relO = aG3relF + aFrelE + aErelO;
```

```
%defining forces acting on system
%don't define reaction forces because we ignore them when we do AMB
gravityForce1 = m1*g*e1;
gravityForce2 = m2*g*e1;
gravityForce3 = m3*g*e1;

%The stuff that goes into the angular momentum balance
%AMB about point O
H1relODot = cross(rG1relO, m1*aG1relO) + I1*theta1doubledot*e3;
H2relODot = cross(rG2relO, m2*aG2relO) + I2*theta2doubledot*e3;
H3relODot = cross(rG3relO, m3*aG3relO) + I3*theta3doubledot*e3;
HrelODot = H1relODot + H2relODot + H3relODot;

%AMB about point E
H2relEDot = cross(rG2relE, m2*aG2relO) + I2*theta2doubledot*e3;
H3relEDot = cross(rG3relE, m3*aG3relO) + I3*theta3doubledot*e3;
HrelEDot = H2relEDot + H3relEDot;

%AMB about point F
HrelFDot = cross(rG3relF, m2*aG3relO) + I3*theta3doubledot*e3;

%The net moments of each of these cases
MrelO = cross(rG1relO, gravityForce1) + cross(rG2relO, gravityForce2) +...
    cross(rG3relO, gravityForce3);
MrelE = cross(rG2relE, gravityForce2) + cross(rG3relE, gravityForce3);
MrelF = cross(rG3relF, gravityForce3);


%equations to solve
%These equations should equal 0 at all times for system
%eqn1 = dot(MrelO - HrelODot, e3) %I think this also works?
eqn1 = MrelO - HrelODot;
eqn1 = eqn1(3);
eqn2 = MrelE - HrelEDot;
eqn2 = eqn2(3);
eqn3 = MrelF - HrelFDot;
eqn3 = eqn3(3);
eqns = [eqn1; eqn2; eqn3];

thetaDoubleDots = [theta1doubledot, theta2doubledot, theta3doubledot];
[A,b] = equationsToMatrix(eqns, thetaDoubleDots);
A = simplify(A)
b = simplify(b)


matlabFunction(A, 'file', 'rhsStuffmassMatrixAMB');
matlabFunction(b, 'file', 'rhsStuffbVectorAMB');
```

5.2 Deriving the triple pendulum EOM using Lagrange: PendDeriveEOMLagrange.m

```
%This derives the EOM for the double pendulum. This does it using Lagrange
%equations.

%Need the real or else Matlab does some funky stuff with complex conjugates
syms d1 d2 d3 L1 L2 m1 I1 m2 I2 m3 I3 g t real %parameters (Note d one, not d L)
syms theta1 theta2 theta3 theta1dot theta2dot theta3dot real %states
syms theta1doubledot theta2doubledot theta3doubledot real %states
syms e1 e2 e3 er1 etheta1 er2 etheta2 er3 etheta3 real %unit vectors
syms rG1relO aG1relO                           %relevant vectors
syms rG2relO aG2relO rG2relE aG2relE rErelO aErelO %relevant vectors
syms rG3relO aG3relO rG3relF aG3relF rFrelE aFrelE real %relveant vectors
syms rG3relE aG3relE real
syms MrelO MrelE MrelF H1relODot H2relODot H3relODot real

%defining unit vectors
e1 = [1 0 0]';
e2 = [0 1 0]';
e3 = cross(e1,e2);

er1 = cos(theta1)*e1 + sin(theta1)*e2;
etheta1 = cross(e3, er1);
er2 = cos(theta2)*e1 + sin(theta2)*e2;
etheta2 = cross(e3, er2);
er3 = cos(theta3)*e1 + sin(theta3)*e2;
etheta3 = cross(e3, er3);

%defining position and velocity vectors of interest
rG1relO = d1*er1;               %first link

%second link
rErelO  = L1*er1;
rG2relE = d2*er2;
rG2relO = rErelO + rG2relE;

%third link
rFrelE = L2*er2;
rG3relF = d3*er3;
rG3relO = rErelO + rFrelE + rG3relF;

vG1relO = d1*theta1dot*etheta1; %first link
%second link
vErelO = L1*theta1dot*etheta1;
vG2relE = d2*theta2dot*etheta2;
vG2relO = vErelO + vG2relE;
%third link
vFrelE = L2*theta2dot*etheta2;
vG3relF = d3*theta3dot*etheta3;
vG3relO = vErelO + vFrelE + vG3relF;


%Kinetic Energy
Ek1 = 1/2*m1*dot(vG1relO, vG1relO) + 1/2*I1*theta1dot^2;
Ek2 = 1/2*m2*dot(vG2relO, vG2relO) + 1/2*I2*theta2dot^2;
Ek3 = 1/2*m3*dot(vG3relO, vG3relO) + 1/2*I3*theta3dot^2;
Ek = Ek1 + Ek2 + Ek3;

%Potential Energy
Ep1 = -m1*g*rG1relO;  %Negative because 0 level at origin O
Ep2 = -m2*g*rG2relO;
Ep3 = -m3*g*rG3relO;
Ep = Ep1(1) + Ep2(1) + Ep3(1);

%Setting up the Lagrange Equation
L  = Ek - Ep;

thetas = [theta1, theta2, theta3];
thetaDots = [theta1dot, theta2dot, theta3dot];
thetaDoubleDots = [theta1doubledot, theta2doubledot, theta3doubledot];
```

```
%d/dt(dL/dxDot) - dL/dx = 0

dLdthetaDots = jacobian(L, thetaDots);
%the first three terms are the left term of the equation (chain rule)
eqns =  jacobian(dLdthetaDots,thetas)*thetaDots' ...
      + jacobian(dLdthetaDots,thetaDots)*thetaDoubleDots' ...
      + jacobian(dLdthetaDots,t)           ...
      - jacobian(L,thetas)';

[M,b] = equationsToMatrix(eqns,thetaDoubleDots);
M_L = simplify(M);
b_L = simplify(b);

matlabFunction(M_L, 'file', 'rhsStuffmassMatrixLagrange');
matlabFunction(b_L, 'file', 'rhsstuffbVectorLagrange');
```

5.3 Deriving the triple pendulum EOM using DAE: tPendDeriveEOMDAE.m

```
%This derives the EOM for the triple pendulum. This does it using DAE.

syms d1 d2 d3 L1 L2 L3 m1 I1 m2 I2 m3 I3 g real %parameters (Note d one, not d L)
syms theta1 theta2 theta3 theta1dot theta2dot theta3dot real %states
syms theta1doubledot theta2doubledot theta3doubledot real %states
syms x1 x1dot x1doubledot y1 y1dot y1doubledot real %states
syms x2 x2dot x2doubledot y2 y2dot y2doubledot real %states
syms x3 x3dot x3doubledot y3 y3dot y3doubledot real %states
syms e1 e2 e3 er1 etheta1 er2 etheta2 er3 etheta3 real %unit vectors
syms rG1relO aG1relO real                          %relevant vectors
syms rG2relO aG2relO rG2relE aG2relE rErelO aErelO real %relevant vectors
syms rG3relO aG3relO rG3relF aG3relF rFrelE aFrelE real %relveant vectors
syms MrelG1 MrelG2 MrelG3 real
syms ROx ROy REx REy RFx RFy real                  %Reaction forces

%15 unknowns, 15 equations (9 from LMB/AMB, 6 from constraint)
unknowns = [x1doubledot, y1doubledot, theta1doubledot, ...
   x2doubledot, y2doubledot, theta2doubledot, ...
   x3doubledot, y3doubledot, theta3doubledot, ...
   ROx, ROy, REx, REy, RFx, RFy];

%defining unit vectors
e1 = [1 0 0]';
e2 = [0 1 0]';
e3 = cross(e1,e2);

er1 = cos(theta1)*e1 + sin(theta1)*e2;
etheta1 = cross(e3, er1);
er2 = cos(theta2)*e1 + sin(theta2)*e2;
etheta2 = cross(e3, er2);
er3 = cos(theta3)*e1 + sin(theta3)*e2;
etheta3 = cross(e3, er3);

%defining vectors of interest
%For the first bar
rG1relO = d1*er1;
rOrelG1 = -rG1relO;
aG1relO = d1*theta1doubledot *etheta1 - d1*theta1dot^2 * er1;

%For the second bar (forearm)
rG2relE = d2*er2;
```

```
rErelG2 = -rG2relE;
rErelO = L1*er1;
rG2relO = rG2relE + rErelO;

aG2relE = d2*theta2doubledot*etheta2 - d2*theta2dot^2 * er2;
aErelO = L1*theta1doubledot*etheta1 - L1*theta1dot^2 * er1;
aG2relO = aG2relE + aErelO;

rErelG1 = rErelO - rG1relO;
%aErelG1 = aErelO - aG1relO;  %Added this

%For the third bar
rG3relF = d3*er3;
rFrelG3 = -rG3relF;

rFrelE = L2*er2;
rFrelO = rFrelE + rErelO;

rG3relE = rG3relF + rFrelE;
rG3relO = rG3relF + rFrelE + rErelO;

rFrelG2 = rFrelO - rG2relO;
%added these here too
%rHrelF = L3*er3;
%rHrelO = rHrelF + rFrelO;
%rHrelG3 = rHrelO - rG3relO;

aG3relF = d3*theta3doubledot*etheta3 - d3*theta3dot^2 * er3;
%aG3relF = cross(theta3doubledot*e3, rG3relF) - theta3dot^2*d3*er3
aFrelE = L2*theta2doubledot*etheta2 - L2*theta2dot^2 * er2;
aG3relE = aG3relF + aFrelE;
aG3relO = aG3relF + aFrelE + aErelO;
%aFrelG2 = aFrelE - aG2relE;     %added this
%aHrelF = L3*theta3doubledot*etheta3 - L3*theta3dot^2 * er3; %added this
%aHrelG3 = aHrelF - aG3relF;     %added this


%defining forces acting on system
gravityForce1 = m1*g*e1;
gravityForce2 = m2*g*e1;
gravityForce3 = m3*g*e1;

%Reaction forces
reactO = ROx*e1 + ROy*e2;
reactE = REx*e1 + REy*e2;
reactF = RFx*e1 + RFy*e2;

%Forces on Links
forcesOnFirstLink = -reactO + reactE + gravityForce1;
forcesOnSecondLink = -reactE + reactF + gravityForce2;
forcesOnThirdLink = -reactF + gravityForce3;

%Linear Momentum Balance Equations
%First 2 equations (actually more like 6 equations)
%lmb1 = forcesOnFirstLink - m1*aG1relO;
lmb1x = forcesOnFirstLink(1) - m1*x1doubledot; %LMB1 in the x direction
lmb1y = forcesOnFirstLink(2) - m1*y1doubledot; %LMB1 in the y direction
%lmb2 = forcesOnSecondLink - m2*aG2relO;
lmb2x = forcesOnSecondLink(1) - m2*x2doubledot; %LMB2 in the x direction
lmb2y = forcesOnSecondLink(2) - m2*y2doubledot; %LMB2 in the y direction
%lmb3 = forcesonThirdLink - m3*aG3relO;
lmb3x = forcesOnThirdLink(1) - m3*x3doubledot;
```

```
lmb3y = forcesOnThirdLink(2) - m3*y3doubledot;

%Angular Momentum Balance Equations
%HrelODot = cross(rG1relO, m1*aG1relO) + I1*theta1doubledot*e3;
%MrelO = cross(rG1relO, gravityForce1) + cross(rG2relO, gravityForce2);
%
HrelG1Dot = I1*theta1doubledot*e3;
MrelG1 = cross(rOrelG1, -reactO) + cross(rErelG1, reactE);

HrelG2Dot = I2*theta2doubledot*e3;
MrelG2 = cross(rErelG2, -reactE) + cross(rFrelG2, reactF);

HrelG3Dot = I3*theta3doubledot*e3;
MrelG3 = cross(rFrelG3, -reactF);

%amb1 = dot(HrelODot - MrelO, e3)
amb1 = dot(HrelG1Dot - MrelG1, e3);
amb2 = dot(HrelG2Dot - MrelG2, e3);
amb3 = dot(HrelG3Dot - MrelG3, e3);
%

%{
%The stuff that goes into the angular momentum balance
%AMB about point O, ie. these equations also work.
HrelODot = cross(rG1relO, m1*aG1relO) + I1*theta1doubledot*e3;
%AMB about point E
HrelEDot = cross(rG2relE, m2*aG2relO) + I2*theta2doubledot*e3;
%AMB about point F
HrelFDot = cross(rG3relF, m2*aG3relO) + I3*theta3doubledot*e3;

MrelO = cross(rG1relO, gravityForce1) + cross(rErelO, reactE);
MrelE = cross(rG2relE, gravityForce2) + cross(rFrelE, reactF);
MrelF = cross(rG3relF, gravityForce3);

amb1 = dot(HrelODot - MrelO, e3);
amb2 = dot(HrelEDot - MrelE, e3);
amb3 = dot(HrelFDot - MrelF, e3);
%}

%Constraint Equations
constraint1x = x1doubledot - aG1relO(1);
constraint1y = y1doubledot - aG1relO(2);

constraint2x = x2doubledot - aG2relO(1);
constraint2y = y2doubledot - aG2relO(2);

constraint3x = x3doubledot - aG3relO(1);
constraint3y = y3doubledot - aG3relO(2);

%{
%These also work
constraint1x = x1doubledot - aG1relO(1)
constraint1y = y1doubledot - aG1relO(2);

constraint2x = x2doubledot - aG2relE(1) - (x1doubledot + aErelG1(1));
constraint2y = y2doubledot - aG2relE(2) - (y1doubledot + aErelG1(2));

constraint3x = x3doubledot - aG3relF(1) - (x2doubledot + aFrelG2(1))
constraint3y = y3doubledot - aG3relF(2) - (y2doubledot + aFrelG2(2));
%}

%Putting all the equations into a matrix 15 equations
```

```
eqns = [lmb1x, lmb1y, lmb2x, lmb2y, lmb3x, lmb3y,...
    amb1, amb2, amb3,...
    constraint1x, constraint1y,...
    constraint2x, constraint2y,...
    constraint3x, constraint3y];

[A,b] = equationsToMatrix(eqns, unknowns);
A = simplify(A)
b = simplify(b)

matlabFunction(A, 'file', 'rhsStuffmassMatrixDAE');
matlabFunction(b, 'file', 'rhsStuffbVectorDAE');
```

5.4 Plotting the triple pendulum

```
clear all;
close all;

%Triple pendulum. Hinges at origin O, then elbow E, then elbow F.
%Neglect all friction and assume there are no joint motors.

%Parameters
p.L1 = 1; p.L2 = 1; p.L3 = 1;
p.d1 = 0.5; p.d2 = 0.5; p.d3 = 0.5;
p.m1 = 1; p.m2 = 1; p.m3 = 1;
%p.I1 = 1/12*p.m1*p.d1^2; p.I2 = 1/12*p.m2*p.d2^2;
p.g = 1; p.I1 = 0.2; p.I2 = 0.3; p.I3 = 0.4;

%setting up tspan
dur = 20;
npoints = 200;
tspan = linspace(0, dur, npoints);

%initial condition
%z0 = [pi/2; pi; pi; 0; 0; 0];
%z0 = [pi/4; pi/2; 3*pi/4;0;0;0];
%z0 = [pi/2; pi; pi;0;0;0];
z0 = [pi; pi/3; pi/3;0;0;0];

%Setting tolerance
options = odeset('RelTol', 1e-6, 'AbsTol', 1e-6);

%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Solving using the NE Method
fAMB = @(t,z) rhsAMB(z,p);
[tArrayAMB, zArrayAMB] = ode45(fAMB, tspan, z0, options);

theta1ArrayAMB = zArrayAMB(:,1);
theta2ArrayAMB = zArrayAMB(:,2);
theta3ArrayAMB = zArrayAMB(:,3);

%Getting the points of interest
xEArrayAMB = p.L1.*sin(theta1ArrayAMB);
yEArrayAMB = -(p.L1.*cos(theta1ArrayAMB));

xFArrayAMB = p.L1.*sin(theta1ArrayAMB) + p.L2.*sin(theta2ArrayAMB);
yFArrayAMB = -(p.L1.*cos(theta1ArrayAMB) + p.L2.*cos(theta2ArrayAMB));
```

```
xEndArrayAMB = xFArrayAMB + p.L3.*sin(theta3ArrayAMB);
yEndArrayAMB = yFArrayAMB - p.L3.*cos(theta3ArrayAMB);

xOAMB = zeros(length(tArrayAMB), 1);
yOAMB = zeros(length(tArrayAMB), 1);

%The thing to actually plot
xPlotAMB = [xOAMB, xEArrayAMB, xFArrayAMB, xEndArrayAMB];
yPlotAMB = [yOAMB, yEArrayAMB, yFArrayAMB, yEndArrayAMB];

%
figure(1)
for i = 1:length(tArrayAMB)
    plot(xPlotAMB(i,:),yPlotAMB(i,:),'LineWidth', 5)
    axis equal
    axis ([-6 6 -6 6]);
    title('Animation of Triple Pendulum-AMB')
    xlabel('x Position')
    ylabel('y Position')
    pause(0.00001);
    shg;
end
%
%
figure(2);    %just the path of it
plot(xEndArrayAMB, yEndArrayAMB);
%title('Position of the end of 3rd link over time-AMB')
xlabel('x Position')
ylabel('y Position')

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

%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Solving using the DAE Method
fDAE = @(t,z) rhsDAE(z,p);
[tArrayDAE, zArrayDAE] = ode45(fDAE, tspan, z0, options);

theta1ArrayDAE = zArrayDAE(:,1);
theta2ArrayDAE = zArrayDAE(:,2);
theta3ArrayDAE = zArrayDAE(:,3);

%Getting the points of interest
xEArrayDAE = p.L1.*sin(theta1ArrayDAE);
yEArrayDAE = -(p.L1.*cos(theta1ArrayDAE));

%xFArrayDAE = p.L1.*sin(theta1ArrayDAE) + p.L2.*sin(theta2ArrayDAE);
%yFArrayDAE = -(p.L1.*cos(theta1ArrayDAE) + p.L2.*cos(theta2ArrayDAE));
xFArrayDAE = xEArrayDAE + p.L2.*sin(theta2ArrayDAE);
yFArrayDAE = yEArrayDAE - p.L2.*cos(theta2ArrayDAE);

xEndArrayDAE = xFArrayDAE + p.L3.*sin(theta3ArrayDAE);
yEndArrayDAE = yFArrayDAE - p.L3.*cos(theta3ArrayDAE);

xODAE = zeros(length(tArrayDAE), 1);
yODAE = zeros(length(tArrayDAE), 1);


%The thing to actually plot
xPlotDAE = [xODAE, xEArrayDAE, xFArrayDAE, xEndArrayDAE];
yPlotDAE = [yODAE, yEArrayDAE, yFArrayDAE, yEndArrayDAE];
```

```
%{
figure(3)
for i = 1:length(tArrayDAE)
    plot(xPlotDAE(i,:),yPlotDAE(i,:),'LineWidth', 5)
    axis equal
    axis ([-6 6 -6 6]);
    title('Animation of Triple Pendulum-DAE')
    xlabel('x Position')
    ylabel('y Position')
    pause(0.00001);
    shg;
end
%}
figure(4);    %just the path of it
plot(xEndArrayDAE, yEndArrayDAE);
title('Position of the end of 3rd link over time-DAE')
xlabel('x Position')
ylabel('y Position')
%

%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Solving using the Lagrange Method
fLagrange = @(t,z) rhsLagrange(z,p);
[tArrayLagrange, zArrayLagrange] = ode45(fLagrange, tspan, z0, options);

theta1ArrayLagrange = zArrayLagrange(:,1);
theta2ArrayLagrange = zArrayLagrange(:,2);
theta3ArrayLagrange = zArrayLagrange(:,3);

%Getting the points of interest
xFArrayLagrange = p.L1.*sin(theta1ArrayLagrange) + p.L2.*sin(theta2ArrayLagrange);
yFArrayLagrange = -(p.L1.*cos(theta1ArrayLagrange) + p.L2.*cos(theta2ArrayLagrange));

xEArrayLagrange = p.L1.*sin(theta1ArrayLagrange);
yEArrayLagrange = -(p.L1.*cos(theta1ArrayLagrange));

xEndArrayLagrange = xFArrayLagrange + p.L3.*sin(theta3ArrayLagrange);
yEndArrayLagrange = yFArrayLagrange - p.L3.*cos(theta3ArrayLagrange);


xOLagrange = zeros(length(tArrayLagrange), 1);
yOLagrange = zeros(length(tArrayLagrange), 1);


%The thing to actually plot
xPlotLagrange = [xOLagrange, xEArrayLagrange, xFArrayLagrange, xEndArrayLagrange];
yPlotLagrange = [yOLagrange, yEArrayLagrange, yFArrayLagrange, yEndArrayLagrange];
%{
figure(5)
for i = 1:length(tArrayLagrange)
    plot(xPlotLagrange(i,:),yPlotLagrange(i,:),'LineWidth', 5)
    axis equal
    axis ([-6 6 -6 6]);
    title('Animation of Triple Pendulum-Lagrange')
    xlabel('x Position')
    ylabel('y Position')
    pause(0.00001);
    shg;
end
%}
```

```
figure(6);    %just the path of it
plot(xEndArrayLagrange, yEndArrayLagrange);
title('Position of the end of 3rd link over time-Lagrange')
xlabel('x Position')
ylabel('y Position')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%}

%
figure(7);
plot(tArrayLagrange, xEndArrayAMB- xEndArrayLagrange)
xlabel('Time(s)');
ylabel('x_{NE} - x_{Lagrange}');
%
%
figure(8);
plot(tArrayAMB, xEndArrayAMB - xEndArrayDAE);
xlabel('Time(s)');
ylabel('x_{NE} - x_{DAE}');
%


function zDot = rhsAMB(z,p)
    %unpacks all the parameters into rhs function using the NE method
    names = fieldnames(p);
    for i = 1:length(names)
        eval([names{i} '= p.' names{i} ';']);
    end

    %unpack the state
    theta1 = z(1); theta2 = z(2); theta3 = z(3);
    theta1dot = z(4); theta2dot = z(5); theta3dot = z(6);
    %W = z(7);  %For energy check


    %getting the EOM from AMB, ie. the functions generated from
    %dPendDeriveEOMamb file
    A = rhsStuffmassMatrixAMB(I1,I2,I3,L1,L2,d1,d2,d3,m1,m2,m3,theta1,theta2,theta3);
    b = rhsStuffbVectorAMB(L1,L2,d1,d2,d3,g,m1,m2,m3,theta1,theta2,theta3,theta1dot,theta2dot,theta3dot);
    q = A\b;
    theta1doubledot = q(1);
    theta2doubledot = q(2);
    theta3doubledot = q(3);

    %Energy Check
    %Wdot = P;

    zDot = [theta1dot, theta2dot, theta3dot,...
        theta1doubledot, theta2doubledot theta3doubledot]';
end

function zDot = rhsDAE(z,p)
    %unpacks all the parameters into rhs function using the DAE method
    names = fieldnames(p);
    for i = 1:length(names)
        eval([names{i} '= p.' names{i} ';']);
    end

    %unpack the state
    theta1 = z(1); theta2 = z(2); theta3 = z(3);
    theta1dot = z(4); theta2dot = z(5); theta3dot = z(6);
    %getting the EOM from AMB, ie. dPendDeriveEOMdae file
    A = rhsStuffmassMatrixDAE(I1,I2,I3,L1,L2,d1,d2,d3,m1,m2,m3,theta1,theta2,theta3);
    b = rhsStuffbVectorDAE(L1,L2,d1,d2,d3,g,m1,m2,m3,theta1,theta2,theta3,theta1dot,theta2dot,theta3dot);

    q = A\b;

    theta1doubledot =q(3);
    theta2doubledot =q(6);
    theta3doubledot =q(9);
    zDot = [theta1dot, theta2dot, theta3dot,...
        theta1doubledot, theta2doubledot theta3doubledot]';
end


function zDot = rhsLagrange(z,p)
    %unpacks all the parameters into rhs function using the Lagrange method
    names = fieldnames(p);
    for i = 1:length(names)
        eval([names{i} '= p.' names{i} ';']);
    end

    %unpack the state
    theta1 = z(1); theta2 = z(2); theta3 = z(3);
    theta1dot = z(4); theta2dot = z(5); theta3dot = z(6);


    %getting the EOM from AMB, ie. tPendDeriveEOMLagrange file
    M_L = rhsStuffmassMatrixLagrange(I1,I2,I3,L1,L2,d1,d2,d3,m1,m2,m3,theta1,theta2,theta3);
    b_L =
        rhsstuffbVectorLagrange(L1,L2,d1,d2,d3,g,m1,m2,m3,theta1,theta2,theta3,theta1dot,theta2dot,theta3dot);

    q = M_L\b_L;
    theta1doubledot =q(1);
    theta2doubledot =q(2);
    theta3doubledot =q(3);

    zDot = [theta1dot, theta2dot, theta3dot,...
        theta1doubledot, theta2doubledot theta3doubledot]';
end
```

5.5 Deriving the four bar linkage EOM using DAE- fourBarDerive.m

```
%This derives the EOM for the four bar linkage. This does it using DAE.

syms d1 d2 d3 L1 L2 L3 m1 I1 m2 I2 m3 I3 g real %parameters (Note d one, not d L)
syms theta1 theta2 theta3 theta1dot theta2dot theta3dot real %states
syms theta1doubledot theta2doubledot theta3doubledot real %states
syms x1 x1dot x1doubledot y1 y1dot y1doubledot real %states
syms x2 x2dot x2doubledot y2 y2dot y2doubledot real %states
syms x3 x3dot x3doubledot y3 y3dot y3doubledot real %states
syms e1 e2 e3 er1 etheta1 er2 etheta2 er3 etheta3 real %unit vectors
syms rG1relO aG1relO real                          %relevant vectors
syms rG2relO aG2relO rG2relE aG2relE rErelO aErelO real %relevant vectors
syms rG3relO aG3relO rG3relF aG3relF rFrelE aFrelE real %relevant vectors
syms MrelG1 MrelG2 MrelG3 real
syms ROx ROy REx REy RFx RFy RHx RHy real          %Reaction forces

%17 unknowns, 17 equations (9 from LMB/AMB, 8 from constraint)
```

```matlab
unknowns = [x1doubledot, y1doubledot, theta1doubledot, ...
    x2doubledot, y2doubledot, theta2doubledot, ...
    x3doubledot, y3doubledot, theta3doubledot, ...
    ROx, ROy, REx, REy, RFx, RFy, RHx, RHy];

%defining unit vectors
e1 = [1 0 0]';
e2 = [0 1 0]';
e3 = cross(e1,e2);

er1 = cos(theta1)*e1 + sin(theta1)*e2;
etheta1 = cross(e3, er1);
er2 = cos(theta2)*e1 + sin(theta2)*e2;
etheta2 = cross(e3, er2);
er3 = cos(theta3)*e1 + sin(theta3)*e2;
etheta3 = cross(e3, er3);

%defining vectors of interest
%For the first bar
rG1relO = d1*er1;
rOrelG1 = -rG1relO;
aG1relO = d1*theta1doubledot *etheta1 - d1*theta1dot^2 * er1;

%For the second bar (forearm)
rG2relE = d2*er2;
rErelG2 = -rG2relE;
rErelO = L1*er1;
rG2relO = rG2relE + rErelO;

aG2relE = d2*theta2doubledot*etheta2 - d2*theta2dot^2 * er2;
aErelO = L1*theta1doubledot*etheta1 - L1*theta1dot^2 * er1;
aG2relO = aG2relE + aErelO;

rErelG1 = rErelO - rG1relO;
aErelG1 = aErelO - aG1relO;

%For the third bar
rG3relF = d3*er3;
rFrelG3 = -rG3relF;

rFrelE = L2*er2;
rFrelO = rFrelE + rErelO;

rG3relE = rG3relF + rFrelE;
rG3relO = rG3relF + rFrelE + rErelO;

rFrelG2 = rFrelO - rG2relO;

rHrelF = L3*er3;
rHrelO = rHrelF + rFrelO;
rHrelG3 = rHrelO - rG3relO;

aG3relF = d3*theta3doubledot*etheta3 - d3*theta3dot^2 * er3;
aFrelE = L2*theta2doubledot*etheta2 - L2*theta2dot^2 * er2;
aFrelG2 = aFrelE - aG2relE;

aG3relE = aG3relF + aFrelE;
aG3relO = aG3relF + aFrelE + aErelO;

aHrelF = L3*theta3doubledot*etheta3 - L3*theta3dot^2 * er3;
aHrelG3 = aHrelF - aG3relF;
```

```
%defining forces acting on system
gravityForce1 = m1*g*e1;
gravityForce2 = m2*g*e1;
gravityForce3 = m3*g*e1;

%Reaction forces
reactO = ROx*e1 + ROy*e2;
reactE = REx*e1 + REy*e2;
reactF = RFx*e1 + RFy*e2;
reactH = RHx*e1 + RHy*e2;

%Forces on Links
forcesOnFirstLink = -reactO + reactE + gravityForce1;
forcesOnSecondLink = -reactE + reactF + gravityForce2;
forcesOnThirdLink = -reactF + reactH + gravityForce3;

%Linear Momentum Balance Equations
%First 2 equations (actually more like 6 equations)
%lmb1 = forcesOnFirstLink - m1*aG1relO;
lmb1x = forcesOnFirstLink(1) - m1*x1doubledot; %LMB1 in the x direction
lmb1y = forcesOnFirstLink(2) - m1*y1doubledot; %LMB1 in the y direction
%lmb2 = forcesOnSecondLink - m2*aG2relO;
lmb2x = forcesOnSecondLink(1) - m2*x2doubledot; %LMB2 in the x direction
lmb2y = forcesOnSecondLink(2) - m2*y2doubledot; %LMB2 in the y direction
%lmb3 = forcesonThirdLink - m3*aG3relO;
lmb3x = forcesOnThirdLink(1) - m3*x3doubledot;
lmb3y = forcesOnThirdLink(2) - m3*y3doubledot;

%Angular Momentum Balance Equations
%HrelODot = cross(rG1relO, m1*aG1relO) + I1*theta1doubledot*e3;
%MrelO = cross(rG1relO, gravityForce1) + cross(rG2relO, gravityForce2);
%
HrelG1Dot = I1*theta1doubledot*e3;
MrelG1 = cross(rOrelG1, -reactO) + cross(rErelG1, reactE);

HrelG2Dot = I2*theta2doubledot*e3;
MrelG2 = cross(rErelG2, -reactE) + cross(rFrelG2, reactF);

HrelG3Dot = I3*theta3doubledot*e3;
MrelG3 = cross(rFrelG3, -reactF) + cross(rHrelG3, reactH);

%amb1 = dot(HrelODot - MrelO, e3)
amb1 = dot(HrelG1Dot - MrelG1, e3);
amb2 = dot(HrelG2Dot - MrelG2, e3);
amb3 = dot(HrelG3Dot - MrelG3, e3);
%


%Constraint Equations
constraint1x = x1doubledot - aG1relO(1);
constraint1y = y1doubledot - aG1relO(2);

constraint2x = x2doubledot - aG2relE(1) - (x1doubledot + aErelG1(1));
constraint2y = y2doubledot - aG2relE(2) - (y1doubledot + aErelG1(2));

constraint3x = x3doubledot - aG3relF(1) - (x2doubledot + aFrelG2(1));
constraint3y = y3doubledot - aG3relF(2) - (y2doubledot + aFrelG2(2));

constraint4x = x3doubledot + aHrelG3(1);
constraint4y = y3doubledot + aHrelG3(2);

%Putting all the equations into a matrix 17 equations
```

```
eqns = [lmb1x, lmb1y, lmb2x, lmb2y, lmb3x, lmb3y,...
    amb1, amb2, amb3,...
    constraint1x, constraint1y,...
    constraint2x, constraint2y,...
    constraint3x, constraint3y,...
    constraint4x, constraint4y];

[A,b] = equationsToMatrix(eqns, unknowns);
A = simplify(A)
b = simplify(b)

matlabFunction(A, 'file', 'fourBar_rhsStuffmassMatrixDAE');
matlabFunction(b, 'file', 'fourBar_rhsStuffbVectorDAE');
```

5.6 Plotting the four bar linkage- fourBarPlot.m

```
clear all
close all

%4 Bar Linkage. Hinges at origin O, then elbow E, then elbow F. Constrained
%at the other end at point H. g=1.
%Neglect all friction and assume there are no joint motors.

%Parameters
p.L1 = 1; p.L2 = 1; p.L3 = 2;
p.d1 = 0.5; p.d2 = 0.5; p.d3 = 0.5;
p.m1 = 1; p.m2 = 1; p.m3 = 1;
%p.I1 = 1/12*p.m1*p.d1^2; p.I2 = 1/12*p.m2*p.d2^2;
p.g = 1; p.I1 = 0.2; p.I2 = 0.3; p.I3 = 0.4;

%setting up tspan
dur = 100;
npoints = 200;
tspan = linspace(0, dur, npoints);

%initial condition
%They must satisfy constraint
%z0 = [pi/4; pi; pi;0;0;0];   %stationary condition
theta1_0 = pi - pi/30;
theta2_0 = pi/2;
theta3_0 = 0;
theta1dot_0 = 0;
theta2dot_0 = 0;
theta3dot_0 = 0;

z0 = [theta1_0; theta2_0; theta3_0; theta1dot_0; theta2dot_0; theta3dot_0];

%Setting tolerance
options = odeset('RelTol', 1e-6, 'AbsTol', 1e-6);


%Solving and plotting
fDAE = @(t,z) rhsDAE(z,p);
[tArrayDAE, zArrayDAE] = ode45(fDAE, tspan, z0, options);

theta1ArrayDAE = zArrayDAE(:,1);
theta2ArrayDAE = zArrayDAE(:,2);
theta3ArrayDAE = zArrayDAE(:,3);
```

```
%Getting the points of interest
xEArrayDAE = p.L1.*sin(theta1ArrayDAE);
yEArrayDAE = -(p.L1.*cos(theta1ArrayDAE));

xFArrayDAE = xEArrayDAE + p.L2.*sin(theta2ArrayDAE);
yFArrayDAE = yEArrayDAE - p.L2.*cos(theta2ArrayDAE);

xEndArrayDAE = xFArrayDAE + p.L3.*sin(theta3ArrayDAE);
yEndArrayDAE = yFArrayDAE - p.L3.*cos(theta3ArrayDAE);

xODAE = zeros(length(tArrayDAE), 1);
yODAE = zeros(length(tArrayDAE), 1);


%The thing to actually plot
xPlotDAE = [xODAE, xEArrayDAE, xFArrayDAE, xEndArrayDAE];
yPlotDAE = [yODAE, yEArrayDAE, yFArrayDAE, yEndArrayDAE];

%
figure(1)
for i = 1:length(tArrayDAE)
   plot(xPlotDAE(i,:),yPlotDAE(i,:),'LineWidth', 5)
   axis equal
   axis ([-4 4 -4 4]);
   title('Animation of Triple Pendulum-DAE')
   xlabel('x Position')
   ylabel('y Position')
   pause(0.00001);
   shg;
end


function zDot = rhsDAE(z,p)
   %unpacks all the parameters into rhs function using the DAE method
   names = fieldnames(p);
   for i = 1:length(names)
      eval([names{i} '= p.' names{i} ';']);
   end

   %unpack the state
   theta1 = z(1); theta2 = z(2); theta3 = z(3);
   theta1dot = z(4); theta2dot = z(5); theta3dot = z(6);

   A = fourBar_rhsStuffmassMatrixDAE(I1,I2,I3,L1,L2,L3,d1,d2,d3,m1,m2,m3,theta1,theta2,theta3);
   b =
       fourBar_rhsStuffbVectorDAE(L1,L2,L3,d1,d2,d3,g,m1,m2,m3,theta1,theta2,theta3,theta1dot,theta2dot,theta3dot);

   q = A\b;


   theta1doubledot =q(3);
   theta2doubledot =q(6);
   theta3doubledot =q(9);


   accelXY = [q(1) q(2) q(4) q(5) q(7) q(8)];
   zDot = [theta1dot, theta2dot, theta3dot,...
      theta1doubledot, theta2doubledot theta3doubledot]';

end
```