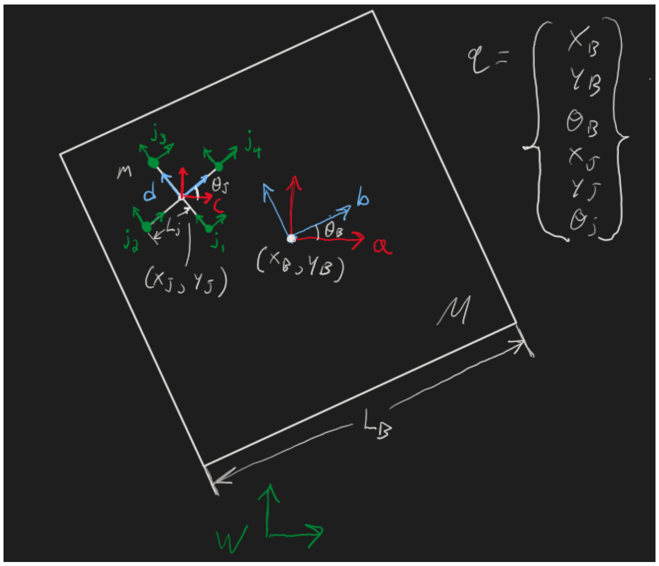
ME 314 Final Project Write-Up

1. I chose to complete the default problem for my final project as described in the drawing below.

2. Coordinate frame drawing and rigid body transformations:




```python
#Rigid body transformations
g_wa = sym.Matrix([
    [1., 0., 0., xB],
    [0., 1., 0., yB],
    [0., 0., 1., 0.],
    [0., 0., 0., 1.]
 ])

g_ab = sym.Matrix([
    [sym.cos(thB), -sym.sin(thB), 0., 0.],
    [sym.sin(thB),  sym.cos(thB), 0., 0.],
    [           0.,            0., 1., 0.],
    [           0.,            0., 0., 1.]
 ])

g_wc = sym.Matrix([
    [1., 0., 0., xJ],
    [0., 1., 0., yJ],
    [0., 0., 1., 0.],
    [0., 0., 0., 1.]
 ])

g_cd = sym.Matrix([
    [sym.cos(thJ), -sym.sin(thJ), 0., 0.],
    [sym.sin(thJ),  sym.cos(thJ), 0., 0.],
    [           0.,            0., 1., 0.],
    [           0.,            0., 0., 1.]
 ])
```

```python
g_dj1 = sym.Matrix([
    [1., 0., 0., 0.],
    [0., 1., 0., -Lj],
    [0., 0., 1., 0.],
    [0., 0., 0., 1.]
 ])

g_dj2 = sym.Matrix([
    [1., 0., 0., -Lj],
    [0., 1., 0., 0.],
    [0., 0., 1., 0.],
    [0., 0., 0., 1.]
])

g_dj3 = sym.Matrix([
    [1., 0., 0., 0.],
    [0., 1., 0., Lj],
    [0., 0., 1., 0.],
    [0., 0., 0., 1.]
])

g_dj4 = sym.Matrix([
    [1., 0., 0., Lj],
    [0., 1., 0., 0.],
    [0., 0., 1., 0.],
    [0., 0., 0., 1.]
])
```

```python
g_wb = g_wa * g_ab #World to box frame transformation

g_wd = g_wc * g_cd #World to jack frame transformation

g_wj1 = g_wd * g_dj1 #World to jack end 1 transformation

g_wj2 = g_wd * g_dj2 #World to jack end 1 transformation

g_wj3 = g_wd * g_dj3 #World to jack end 1 transformation

g_wj4 = g_wd * g_dj4 #World to jack end 1 transformation
```

3. The first step to modeling the dynamics of this system was to calculate its lagrangian. To do this I needed to find the kinetic and potential energy of both the box and the jack as the lagrangian can be described by the equation L= KE - U. The potential energy of the system was simply the sum of the weights of each object multiplied by their respective y-coordinate/height. Finding the kinetic energy of the system required finding the body velocity of the objects in the system since we must account for the translational and rotational kinetic energy in each body as well as their rotational inertia. To calculate

the body velocity for each object, I took the inverse of the g_wd (world to jack) and g_wb (world to box) transformations, multiplied them by their respective time derivatives, and then unhat the resulting matrix. I then plugged these vectors along with the mass and rotational inertia about the z-axis for each respective body into the equation for kinetic energy (in terms of body velocity) to obtain each body's kinetic energy. Adding these energies together and subtracting by the system's total potential energy gave me the lagrangian of the system.

Next, I took the derivative of the lagrangian with respect to each configuration variable and it's time derivative so that I could plug these expressions into the euler-lagrange equations to get a system of equations that I could solve to determine the dynamics of the system for normal, non-collision time steps. I applied external forces to the equations corresponding to the x and y components of the box so that the box could achieve an oscillating/bouncing motion. I simply added these expressions to the right side of the xBox and yBox equations of the system to incorporate them into the system's dynamics.

To ensure that the box would bounce around the center of the simulation and actually simulate someone shaking a cup with a jack in it, the external forces applied to the x and y direction of the box are intended to produce oscillations about the center of the screen. When originally setting up my external forces I chose to use a sine function that took an input of time to vary the force applied to the box, but to prevent time from being defined explicitly in the euler-lagrange equations, I decided to change these forces to use only the x and y position of the box, so that I could keep time defined implicitly which made simulating the system numerically rather than symbolically a bit easier. The final version is essentially a proportional controller with a very high gain that causes oscillations about the y and x axis based on the box's distance from its starting point. The external force in the y direction also adds the weight of the box to this proportional controller value to ensure that the box does not fall out of view due to gravity.

I then moved on to defining the characteristics of the system during impacts between the box and the jack. The first step in describing motion during impact was to determine when exactly these impacts occur. To detect impacts I defined coordinate frames at each end point of the jack which allowed me to convert the origins of each of these frames into the world frame and then from the world frame into the coordinate frame of the box (frame b) so that I could compare the x and y values of the jack in the b frame to the known x and y limits provided by the box boundaries. This coordinate transformation made detecting impacts much simpler, so that I could now create 16 different phi conditions that simply subtract half the side length of the box from the x and y coordinate of each end of the jack in the b frame. This means that the phi condition would equal 0 as expected when an end of the jack makes contact with the wall. Each of these phi conditions are checked at every timestep of the simulation to determine whether an impact has occurred at any point on the jack.

Now I needed to describe the dynamics of the system directly after the impact between the jack and the box as the typical lagrangian representation would no longer be accurate during this short time span of collision. To do this, I found the partial derivative of the lagrangian with respect to the time derivative (the momentum) of each configuration variable and substituted the symbolic functions in this expression to symbolic dummy variables. I also created a second version of this expression where the time derivatives of each configuration variable were substituted for q+ variants which denote the configuration variable velocities after impact. The configuration variables themselves did not need post-collision variants as we assume the position of both objects to be basically the same directly before and directly after impact. This allowed me to set up a system of equations following the impact update laws where the difference in momentum directly before and after collision was set equal to λ times the partial derivative of the triggered phi condition with respect to the configuration variables. To complete the system, I computed the difference in hamiltonian (described by H = p*qdot -L) between these time steps and set that quantity equal to zero. This made for a total of 7 equations that I could then solve symbolically for the configuration variable velocities directly after impact. Additionally, to prevent having to develop 16 different versions of these equations for each phi condition, I made a function that takes the desired phi condition as an input and returns the system of equations corresponding to that phi condition.

4. At the start of the simulation, the jack and the box are both given initial velocities and are both set to begin at the same coordinate of (0, 5) so that the jack is fully inside of the box. During the simulation, the jack bounces from wall to wall of the box as the box moves and oscillates about its starting location. I believe this simulation to be an accurate representation of such a dynamic system because it seems to uphold the conservation of momentum, respond to gravity, and correctly predicts the interaction between two rigid bodies (they don't phase through each other).