ME 314 Final Project

Sonia Yuxiao Lai

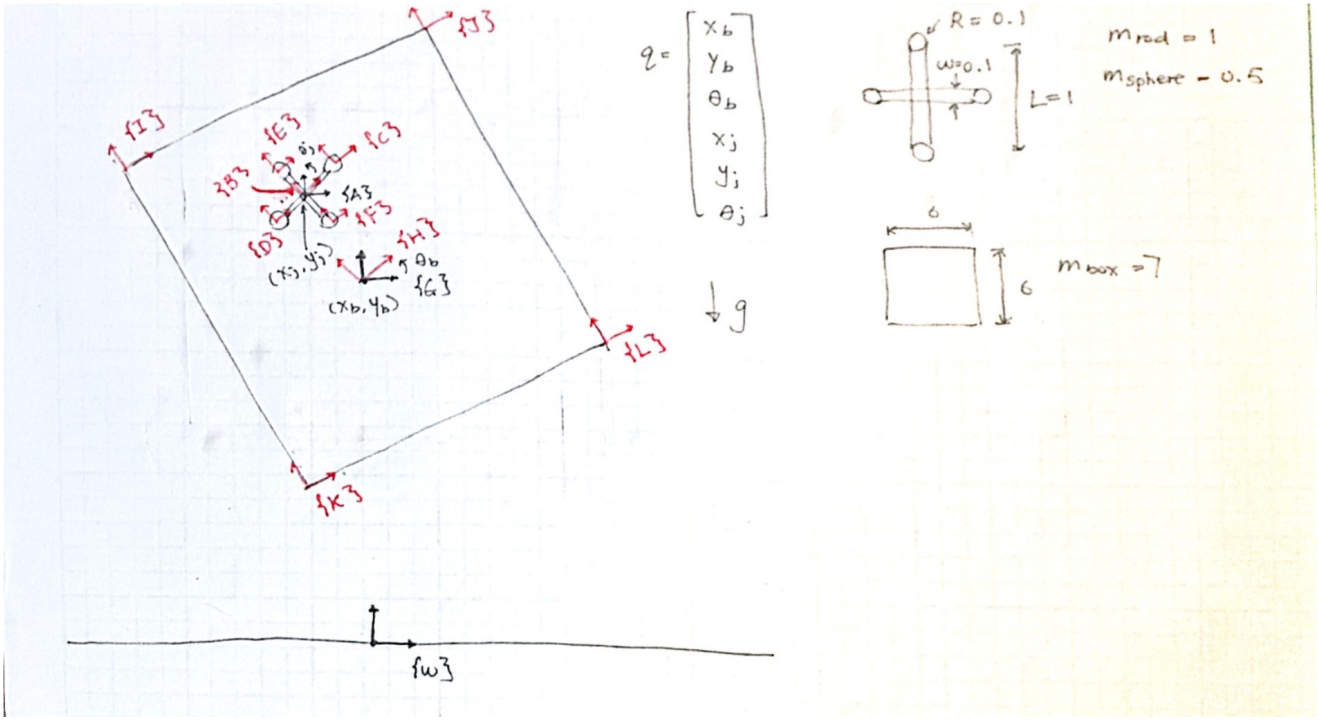

Figure 1. Configuration, frames, and parameters of the system.

Introduction:

The goal of this project is to simulate a jackson bouncing inside a box. The configuration and reference frames are defined as shown in Figure 1. Refer to Appendix A for detailed transformation matrices used during the simulation. The animation simulates jackson starting at center of box and with zero initial velocities and theta for 5 seconds.

Euler-Lagrange Equations:

To calculate Euler-Lagrange equations, first find kinematic energy of the system. The system consists of seven objects: four spheres, two rods, and a box. The velocity of each object is calculated from respective rigid-body transformations from world frame (g_wh for box, g_wb1 and g_wb2 for two rods, and g_wc, g_wd, g_we, and g_wf for four spheres). Since all motions are assumed to be planar, the inertia of all objects involve only rotation around z-axis (out of

page). Therefore, the inertia matrices is defined as diagonal matrix of vector (1, 1, inertia around z). As for the four spheres, use parallel axis theorem to find the inertia of a sphere rotating around center of jackson. Since the parameters of spheres and distance between sphere and rod are the same, the four spheres have the same rotational inertia. The potential energy of the system is also calculated from the rigid-body transformations of all objects in world frame.

The external forces in the system are applied to the x and y configurations of box. These forces is designed to move box along a diagonal line. Assuming that the forces are large enough that the box is fully constrained along the trajectory. As a result, the impact should not affect x and y velocities of box during simulation.

Impact:

The impact update laws now only involve configurations of jackson and theta configurations of box. To detect impact, a phi matrix is setup by calling phi_impact function. This function will return a 8-vector matrix consists of eight elements: x_hc, x_hd, x_he, x_hf, y_hc, y_hd, y_he, y__hf. Each corresponds to x and y values of sphere C, D, E, and F in box frame. The configurations are calculated using transformations g_hc, g_hd, g_he, and g_hf based a input current configuration. Since size of box is the same, phi only needs to consider whether jackson is going to hit box in positive or negative x and y directions. Jackson is likely to hit box if its moving faster than box. Therefore, if velocity of jackson is smaller than box in x or y direction (for example, jackson is moving left but box is moving right, so it's going to hit side IK), the four phi equations should be in form x_sphere = -(L_box/2-2R). If velocity is larger, then they should be in form x_sphere = L_box/2-2R. Once one of these phi equations change sign, impact may occur. However, when jackson is so close to box that one sphere already exceeds boundary, when function impact_condition is called, sign of phi may not have changed. To capture these cases, when calling phi_impact, the function checks if any x or y values of sphere (calculated from g_hc, g_hd, g_he, and g_hf) exceeds a maximum (L_box/2-2R) and returns corresponding index of phi in the matrix that should be experiencing impact.

The impact_update function takes a configuration and calculates the next configuration after impacting. At start of function, phi_impact is called to receive a matrix fo phi equations. The index of phi to use is determined based on the minimum distance between x or y values of a sphere and box side. The impact update assumes that configuration for x and y of box stays

the same because these are controlled by external forces. The function returns new configuration that is calculated from a non-zero lambda result.

There are two special situations where impact does not actually happen but is captured by the impact_condition function. These are handled in the impact_update function. One happens when sign of phi changes but jackson is actually not in collision with box. This typically happens when jackson changes direction due to gravity. One example is that jackson moves up but loses acceleration, and eventually falls down before hitting box. To handle this special case, the impact_update function first checks the x and y values of four spheres in box frame before updating to determine whether jackson is experiencing impact for real. If no impact occurs, the same configuration is returned. Another special situation is that jackson is still in collision with box after updating impact. This usually happens when jackson is at a corner and that the configuration after updating impacts drive jackson to hit an adjacent wall. In this case, even though jackson is trying to move away from box, sign of phi equations may change. That is, impact is detected on two opposite directions. As a result, jackson will oscillate at edge of box. To solve this problem, a small step size (0.005) is used so that smoother trajectory can be generated. Especially, when jackson is bouncing between two adjacent walls, if velocity of jackson increases after hitting one wall, there will be more opportunities for impact_condition to capture impact before going over the boundary of another wall. However, Even if step size is decreased, occasionally some impacts are not correctly captured due to large velocity. To handle this situations, impact_update function checks whether jackson is trying to leave box by comparing distance between jackson and box. Since this case usually occur when using the same phi index of the phi matrix, the function only needs to compare x or y values of the same sphere. If the distance is smaller than the previous one, then impact_update should return the same configuration.

Simulation:

Finally, to simulate jackson bouncing in box, first simulate the system without considering impact. Then loop through the configurations stored previously and use phi matrix to capture impact. If sign of one phi equations changes, then call impact_update function to receive updated configuration and simulate the system with the new configurations. Once all elements

stored are checked, the simulation with impact is complete. See lai_final_demo.mp4 for an animation of a 5-second-simulation. The jackson is bouncing between the sides of the box. In the video, jackson does not oscillate or suddenly change direction at unreasonable position. Also, there does not appear to be unreasonable loses of configurations during the animation, so the strategies explained above should work.

Appendix A.

Below are transformation matrices used in calculating Euler-Lagrange equations, external forces, and impact update laws.

g_wh is the transformation from world to box frame.

g_wa is the transformation from world to center of jackson.

g_wb1 is the transformation from world to one rod of jackson.

g_wb2 is the transformation from world to another rod of jackson.

g_wc, g_wd, g_we, and g_wf are the transformations from world to sphere C, D, E, and F respectively.

g_hc, g_hd, g_he, and g_hf are the transformations from box frame to sphere C, D, E, and F respectively.

g_wh =

$$\begin{bmatrix} -0.447362468930969 & 0.894352738796046 & 0 & x_b(t) \\ -0.894352738796046 & -0.447362468930969 & 0 & y_b(t) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

g_wa =

$$\begin{bmatrix} 1 & 0 & 0 & x_j(t) \\ 0 & 1 & 0 & y_j(t) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

g_wb1 =

$$\begin{bmatrix} \cos(\theta_j(t)) & -\sin(\theta_j(t)) & 0 & x_j(t) \\ \sin(\theta_j(t)) & \cos(\theta_j(t)) & 0 & y_j(t) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

g_wb2 =

$$\begin{bmatrix} -\sin(\theta_j(t)) & -\cos(\theta_j(t)) & 0 & x_j(t) \\ \cos(\theta_j(t)) & -\sin(\theta_j(t)) & 0 & y_j(t) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

g_wc =

$$\begin{bmatrix} \cos(\theta_j(t)) & -\sin(\theta_j(t)) & 0 & x_j(t) + 0.6\cos(\theta_j(t)) \\ \sin(\theta_j(t)) & \cos(\theta_j(t)) & 0 & y_j(t) + 0.6\sin(\theta_j(t)) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

g_wd =

$$\begin{bmatrix} \cos(\theta_j(t)) & -\sin(\theta_j(t)) & 0 & x_j(t) - 0.6\cos(\theta_j(t)) \\ \sin(\theta_j(t)) & \cos(\theta_j(t)) & 0 & y_j(t) - 0.6\sin(\theta_j(t)) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

g_we =

$$\begin{bmatrix} \cos(\theta_j(t)) & -\sin(\theta_j(t)) & 0 & x_j(t) - 0.6\sin(\theta_j(t)) \\ \sin(\theta_j(t)) & \cos(\theta_j(t)) & 0 & y_j(t) + 0.6\cos(\theta_j(t)) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

g_wf =

$$\begin{bmatrix} \cos(\theta_j(t)) & -\sin(\theta_j(t)) & 0 & x_j(t) + 0.6\sin(\theta_j(t)) \\ \sin(\theta_j(t)) & \cos(\theta_j(t)) & 0 & y_j(t) - 0.6\cos(\theta_j(t)) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

g_hc =

$$\begin{bmatrix} \sin(\theta_b(t))\sin(\theta_j(t)) + \cos(\theta_b(t))\cos(\theta_j(t)) & \sin(\theta_b(t))\cos(\theta_j(t)) - \sin(\theta_j(t))\cos(\theta_b(t)) & 0 & (x_j(t) + 0.6\cos(\theta_j(t)))\cos(\theta_b(t)) + (y_j(t) + 0.6\sin(\theta_j(t)))\sin(\theta_b(t)) - x_b(t)\cos(\theta_b(t)) - y_b(t)\sin(\theta_b(t)) \\ -\sin(\theta_b(t))\cos(\theta_j(t)) + \sin(\theta_j(t))\cos(\theta_b(t)) & \sin(\theta_b(t))\sin(\theta_j(t)) + \cos(\theta_b(t))\cos(\theta_j(t)) & 0 & -(x_j(t) + 0.6\cos(\theta_j(t)))\sin(\theta_b(t)) + (y_j(t) + 0.6\sin(\theta_j(t)))\cos(\theta_b(t)) + x_b(t)\sin(\theta_b(t)) - y_b(t)\cos(\theta_b(t)) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

g_hd =

$$\begin{bmatrix} \sin(\theta_b(t))\sin(\theta_j(t)) + \cos(\theta_b(t))\cos(\theta_j(t)) & \sin(\theta_b(t))\cos(\theta_j(t)) - \sin(\theta_j(t))\cos(\theta_b(t)) & 0 & (x_j(t) - 0.6\cos(\theta_j(t)))\cos(\theta_b(t)) + (y_j(t) - 0.6\sin(\theta_j(t)))\sin(\theta_b(t)) - x_b(t)\cos(\theta_b(t)) - y_b(t)\sin(\theta_b(t)) \\ -\sin(\theta_b(t))\cos(\theta_j(t)) + \sin(\theta_j(t))\cos(\theta_b(t)) & \sin(\theta_b(t))\sin(\theta_j(t)) + \cos(\theta_b(t))\cos(\theta_j(t)) & 0 & -(x_j(t) - 0.6\cos(\theta_j(t)))\sin(\theta_b(t)) + (y_j(t) - 0.6\sin(\theta_j(t)))\cos(\theta_b(t)) + x_b(t)\sin(\theta_b(t)) - y_b(t)\cos(\theta_b(t)) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

g_he =

$$\begin{bmatrix} \sin(\theta_b(t))\sin(\theta_j(t)) + \cos(\theta_b(t))\cos(\theta_j(t)) & \sin(\theta_b(t))\cos(\theta_j(t)) - \sin(\theta_j(t))\cos(\theta_b(t)) & 0 & (x_j(t) - 0.6\sin(\theta_j(t)))\cos(\theta_b(t)) + (y_j(t) + 0.6\cos(\theta_j(t)))\sin(\theta_b(t)) - x_b(t)\cos(\theta_b(t)) - y_b(t)\sin(\theta_b(t)) \\ -\sin(\theta_b(t))\cos(\theta_j(t)) + \sin(\theta_j(t))\cos(\theta_b(t)) & \sin(\theta_b(t))\sin(\theta_j(t)) + \cos(\theta_b(t))\cos(\theta_j(t)) & 0 & -(x_j(t) - 0.6\sin(\theta_j(t)))\sin(\theta_b(t)) + (y_j(t) + 0.6\cos(\theta_j(t)))\cos(\theta_b(t)) + x_b(t)\sin(\theta_b(t)) - y_b(t)\cos(\theta_b(t)) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

g_hf =

$$\begin{bmatrix} \sin(\theta_b(t))\sin(\theta_j(t)) + \cos(\theta_b(t))\cos(\theta_j(t)) & \sin(\theta_b(t))\cos(\theta_j(t)) - \sin(\theta_j(t))\cos(\theta_b(t)) & 0 & (x_j(t) + 0.6\sin(\theta_j(t)))\cos(\theta_b(t)) + (y_j(t) - 0.6\cos(\theta_j(t)))\sin(\theta_b(t)) - x_b(t)\cos(\theta_b(t)) - y_b(t)\sin(\theta_b(t)) \\ -\sin(\theta_b(t))\cos(\theta_j(t)) + \sin(\theta_j(t))\cos(\theta_b(t)) & \sin(\theta_b(t))\sin(\theta_j(t)) + \cos(\theta_b(t))\cos(\theta_j(t)) & 0 & -(x_j(t) + 0.6\sin(\theta_j(t)))\sin(\theta_b(t)) + (y_j(t) - 0.6\cos(\theta_j(t)))\cos(\theta_b(t)) + x_b(t)\sin(\theta_b(t)) - y_b(t)\cos(\theta_b(t)) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$