MECH_ENG 314 Final Project Report

Bill Yen

For my final project, I chose to solve the default dice in a cup problem where I have a small cube bouncing around inside a larger cube under gravity (9.8 m/s^2). The larger outer cube is forced in both the x and y direction to shake up the inner cube. I chose to use three frames in my calculations, which are {w}, {j}, and {b} where they represent the world, the center of mass of the jack (a.k.a. dice), and the center of mass of the box (a.k.a. cup) respectively. They are labeled in Figure 1 below. The coordinates system I chose here is q = [x_b, y_b, theta$_b$, x_j, y_j, theta$_j$] where x_b and y_b are the x and y displacement of {b} from {w}, x_j and y_j and the x and y displacement of {j} from {b}, theta$_b$ is the angle between the x axis of {b} and the x axis of {w}, and theta$_j$ is the angle between the x axis of {j} and the x axis of {b}. This allowed me to derive g_{wb} and g_{bj} and consequently g_{wj} ($g_{wb}g_{bj} = g_{wj}$) so I can represent everything in the world frame.

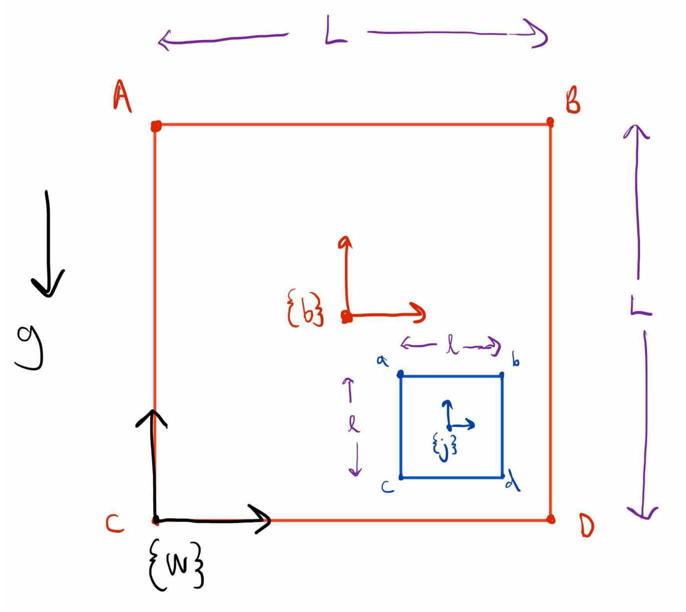

Figure 1: System diagram with {b}, {j}, and {w} frames

I calculated the Euler-Lagrange equation by first identifying the kinetic and potential energy of the entire system through the transformations mentioned above. I calculated the total kinetic energy as 0.5*$V_b(g_{wb})^T$*M_b*$V_b(g_{wb})$ + 0.5*$V_b(g_{wj})^T$*M_j*$V_b(g_{wj})$ where M_j and M_b are:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.166666666666667 \end{bmatrix}$$

$$\begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 166.666666666667 \end{bmatrix}$$

The bottom right hand element corresponds to the moment of inertia of the cube about z (formula of (m/12)(2*L^2) from Wikipedia), and the diagonal elements on the top left are the masses of the jack and the box. V_b also corresponds to $(g^{-1}g')^{hat}$, which we derived in class. The total potential energy is essentially just the height of both {j} and {b} frames (since they are at the center of mass of the jack and the box respectively) times their masses and gravity, which became m*9.8*(Gwj*[0, 0, 0, 1])[1] + M*9.8*(Gwb*[0, 0, 0, 1])[1]. Doing KE-PE gets us the Lagrangian (L), and doing $\frac{-dL}{dq} + \frac{d\frac{dL}{d\dot{q}}}{dt} = F$ will give us the forced Euler-Lagrange equations that we want (since we need to move the cup to stimulate the dice). F in this case is a vector [Fx, Fy, 0, 0, 0, 0].

The impact conditions of this system is calculated by recognizing that when the sum of the distance between one corner of the dice (say corner a) and two of the box corners (say B and A) equals the length of the box's side L, then it will be touching a wall and impact should occur. Practically for detecting impact of point a on the top wall, this was achieved by doing g_{wj}*[-l/2, l/2, 0, 1], g_{wb}*[-L/2, L/2, 0, 1], g_{wb}*[L/2, L/2, 0, 1] to get a, A, and B in terms of the world frame respectively. More detail can be found in the code. A total of 16 impact conditions were identified to simulate this problem (4 dice corner hitting 4 walls).

The external forces were simply achieved through a combination of guess and check and feedback control. Since we do not want the cup to fall down too much, I set F_y = 50*cos(50*t)-(M*g)-50*((Gwb*[0, 0, 0, 1])[1]-L/2) to implement some degree of gravity offset and positional feedback control. Note that the y coordinate of the box in the world frame matters in F_y. I also added the cosine term to add some oscillations. F_x is simpler, with just 200*cos(50*t) to add some sideway shaking.

Since there are 16 different impact conditions, I derived 16 different impact update laws, one for each impact scenario, and appended them into a list to loop through throughout my code. The impact laws were:

$$\frac{\partial L}{\partial \dot{q}}^{+} - \frac{\partial L}{\partial \dot{q}} = \lambda \nabla \varphi(q)$$

$$H^{+} - H = 0$$

Where H is the Hamiltonian of the system and + indicates that it is taken after the impact (e.g. the accelerations variables will be different but position will be the same). Note that the lambda used for each of the 16 conditions will also be different, so I have 16 lambdas as well. I first identified which of the 16 types of impact happened, then I used the system of equations above to solve for xbdplus, ybdplus, thetabdplus, xjdplus, yjdplus, thetajdplus, and the lambda of that condition every time an impact occurs. Only if lambda is not equal to zero will I take that list of solutions, otherwise they won't be valid.

My code does indeed work, and the simulation shows the dice bouncing around in the box, deflecting in a different direction and rotating through the air every time it hits one of the walls. The box actually moves a little bit every time the dice hits as well, and this makes sense because some of its kinetic energy is getting transferred to the box and the masses of the two objects are only an order of magnitude apart. The dice also falls down under gravity (though not noticeably due to scaling of the forces), which is expected. Only thing that seems unnatural here is that if observed carefully, one can notice that the dice doesn’t always touch the walls of the box exactly and sometimes bounces just before, but I believe this can be attributed to integration error and can be solved by simply decreasing dt.