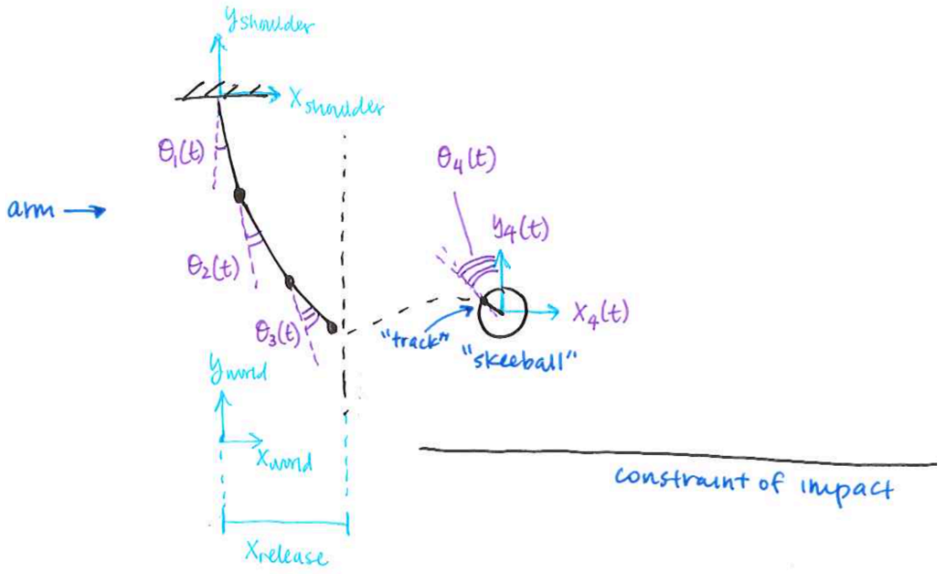
This project was inspired by "Skeeball", a game played at the arcade in which you take a solid ball (the size of a tennis ball) and roll it down lane and up a ramp, with the aim at getting it into various sized goals. My proposal was of similar nature, in which an arm would be modeled to throw the skeeball, the skeeball would roll down a constraint and up a ramp, and then it would hit a backboard with various holes marked as goals.

My final result is slightly simplified, where the goals were removed and the constraint as a flat plane/line (similar to throwing a bowling ball down a bowling lane). This simplification was chosen because the constraint function and additional impact equations became too much to attempt to debug. Instead, my focus was spent on making my simplified code scalable to changing parameters (i.e different height people, different points of releasing the ball, etc) so that I can manipulate parameters and see the effect it has.



There are two parts to the model – a triple pendulum arm, and a ball that impacts the floor many times to model the rolling.

There are three-ish frames in which the model takes place. The arm mechanics and the x4 & y4 coordinates of the skeeball are all calculated with respect to the world origin. The arm is then translated for the animation such that its body frame's origin is located where the shoulder might be with respect to the ground of impact. The rotation of the skeeball is tracked by following a line that connects the center of the ball to one particular point on its edge. The trajectory of the point, called "track", is mapped by a transformation that translates a point (whose distance is the radius of the skeeball away from the origin) by x4 and y4 (of the skeeball) and rotates by the trajectory of θ4.

ARM

Configuration q of θ1, θ2, θ3 for a triple pendulum with a Lagrangian that assumes point masses. The Euler-Lagrange equations has the typical left hand side (derivatives of the Lagrangian) while the right hand side is dictated by forcing parameters to each θ configuration that can be manipulated to see the effects of exerting muscles in different parts of the arm as you throw the ball. In summary, the mechanics of the arm is dictated by the following general equation:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = \begin{pmatrix} F_{\theta_1} \\ F_{\theta_2} \\ F_{\theta_3} \end{pmatrix}$$

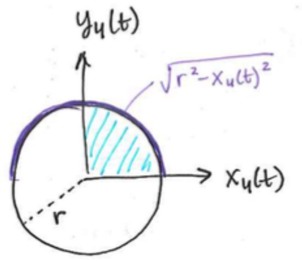
An x value is defined as the point of releasing the ball and the time of release is found using EventLocator in Mathematica. The trajectory of θ1, θ2, and θ3 are used to solve for trajectories of the last pendulum mass (the "hand") and stored as the first trajectory of the skeeball, from time 0 to time tr (time of release).

<u>ROLLING</u>

Configuration q2 of x4, y4, and θ4 of the skeeball location and rotation is used with a Lagrangian that incorporates both translational and rotational KE. Rotational inertia is calculated by the following integral:

$$\mathcal{I} = 4\int_0^r \int_0^{\sqrt{r^2 - x_4(t)^2}} \rho r^T r \, dy\,dx$$

$$= \rho \pi r^2 \left(\frac{r^2}{2}\right) = \frac{mr^2}{2}$$



The result $mr^2/2$ is plugged into the Lagrangian manually so that the mass parameter could be easily incorporated into the Lagrangian (as opposed to calculating the density).

The Euler-Lagrange equations again incorporate external forces, this time in the form of viscous forces to represent air drag. The mechanics of the skeeball is dictated by the following general EL equation (where all the q's are q2).

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = \begin{pmatrix} F_{x_4} \\ F_{y_4} \\ F_{\theta_4} \end{pmatrix}$$

To prepare for impact equations with the ground, an impact constraint is set to be such that y4 is a distance equal to the ball radius above the ground at all times. Impact equations for Hamiltonian and generalized momentum before and after impact are then solved using EventLocator for time and NDsolve to update impact conditions. The resulting trajectory for the ball is appended to the initial trajectory from the pendulum and animated.

<u>DISCUSSION</u>

The motion of the arm is not very realistic and there seems to be a lack of response of the trajectory with respect to changing external forces. This is odd because the setup is almost exactly the same as the external forces of the rolling motion, the latter of which is extremely responsive. I believe my setup is correct, but I had trouble with using the "Thread" command of Mathematica to match up each side of the Euler-Lagrange equations to put into the Solve command, which I believe is the source of error.

The representation of the rotation of the ball looks realistic, but it does not actually take into account the friction against the floor in a realistic manner. When it is truly rolling, there should be a constraint such that the distance traveled/rolled is related to the rotation and radius (arc length) of the ball (like a "no slip" condition). Instead, this model only shows numerous small impacts against the floor, and approaches, but never reaches this true rolling condition. However, the external force in the x4, y4 and θ4 configurations do incorporate a little bit realism to the energy dissipation to the model due to friction as well as air drag.