Read Me

This set of experiments was created from the idea that observations computed from physical equations might provide a superior signal to learning algorithms, compared to the raw state of the system. For example, the total mechanical energy contained in a physical system is

$$E_{\text{mek}} = E_{\text{pot}} E_{\text{kin}} = mgh + \frac{1}{2}mv^2 \tag{1}$$

which might work better for learning a strategy for the bouncing ball, compared to just observing v and p.

Experiments on the Bouncing Ball - story

Training a strategy to hit the bouncing ball by observing the full state (v, p) yields an average performance of 40 swings per 120 seconds. Unfortunately, observing only E_{mek} uses 146 swings per 120 seconds. This is because the outcome of hitting the ball depends on v and p, which leads to a lot of useless swings from the strategy.

Observing E_{mek} in addition to (v, p) leads to [data missing]

Since the hit[1] will only connect for certain values of v and p, I chose to also count the number of times the hits actually connected. This meant that the strategy observing just E_{mek} would not be penalised for all the unnecesary swings that had no effect. By this criterion, E_{mek} strategy then achieved a decent performance of 41 hits per 120 seconds. Which is close to the strategy observing (v, p).

Obseriving both (v, p) and E_{mek} led to 37 swings per 120 seconds.

Running the queries once, there does not seem to be any significant difference between the learning outcomes, except that E_{mek} alone is not enough to efficiently hit the ball, if useless swings are counted.

The queries would have to be repeated multiple times with a set number of training runs to discover if there is any difference.q

Experiments on the Bouncing Ball

The full set of learning queries is seen in Figure 5.

Strategies were learned using observations of either the full state, (v, p), or the mechanical energy formula in Equation 1. Or some combination of the two.

The learner was either trained to minimize the number of "swings" or the number of "hits." This is because the mechanical energy is not enough to determine whether it is possible to hit the ball at the given state. This forces the strategy to take a lot of unnecessary swings in order to be sure to hit the ball. Looking instead only at how often the ball was hit may be more "fair."

The result of running the experiment once with default learning parameters shows 40 swings and 37 hits, when trying to minimize the number of swings. Minimizing just hits, leads to 497 swings and 39 hits.

Observing only E_{mek} Leads to 146 swings and 44 hits when trying to minimize swings. Minimizing for hits seems to be the same, with 41 hits.

[1] This is slightly confusing because choosing the "hit" action does not necesarily hit the ball. It just swings at it. Only if $-4 < v \land p > 4$ does the ball actually get hit. See Figure 4

Also observing p gets the value down to 41 or 38, depending on if it is minimizing for swings or for hits.

Visualisations for the Bouncing Ball

Visualisations showing the mechanical energy of the bouncing ball for different states (v, p) are shown in Figure 1 and Figure 2.

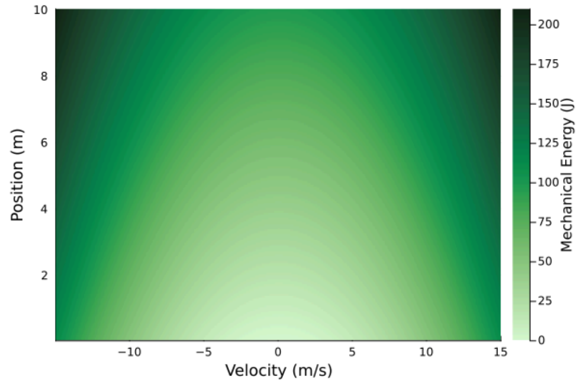

Figure 1: Bouncing ball E_{mek} values.

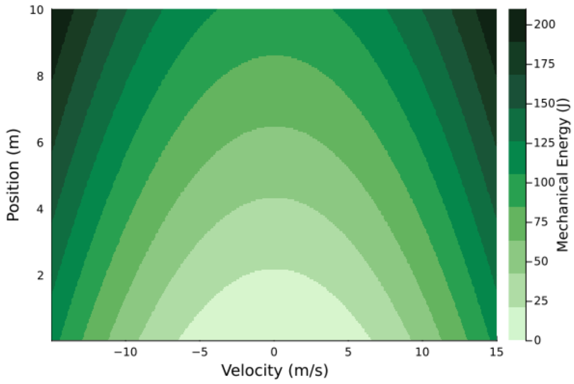

Figure 2: Bouncing ball E_{mek} values grouped into 10 discrete categories.

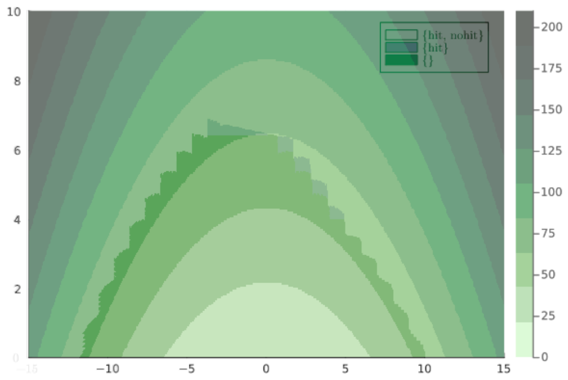

Figure 3: Bouncing ball shield overlaid with Figure 2.

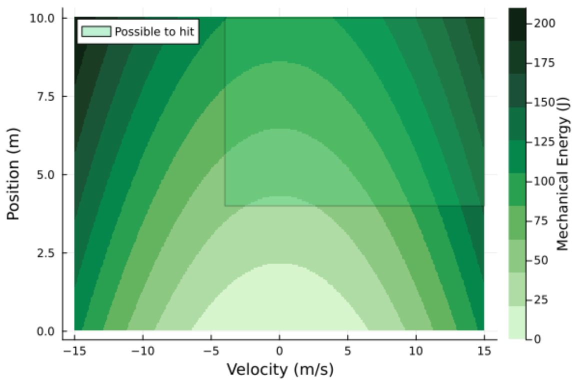

Figure 4: Overlay to Figure 2 showing states where it is possible to hit the ball.

It is not very scientific, but I made sure the axis matched a visualisation of the safety strategy for the bouncing ball, and then I used a graphics editing program to overlay one with the other, which is shown in Figure 3. Thus, the visualisation is not guaranteed to be pixel perfect, but it is very nearly correct.

Appendix

```
E[ ≤ 120;500] (max:swings)                                               598.99 ± 1.52848 (95% CI)
E[ ≤ 120;500] (max:hits)                                                196.694 ± 0.616479 (95% CI)
E[ ≤ 120;500] (max:resets)                                                                    ≈ 0
Pr[ ≤ 120;500] (<> resets > 0)                                                ≤ 0.00735061 (95% CI)
strategy SwingWell = minE (swings + resets*1000 ) [ ≤ 120] {} → {p, v}: <> time ≥ 120
E[ ≤ 120;500] (max:swings) under SwingWell                               40.162 ± 0.284237 (95% CI)
E[ ≤ 120;500] (max:hits) under SwingWell                                 37.774 ± 0.236314 (95% CI)
E[ ≤ 120;500] (max:resets) under SwingWell                                                    ≈ 0
Pr[ ≤ 120;500] (<> resets > 0) under SwingWell                                ≤ 0.00735061 (95% CI)
strategy HitWell = minE (hits + resets*1000 ) [ ≤ 120] {} → {p, v}: <> time ≥ 120
E[ ≤ 120;500] (max:swings) under HitWell                                497.768 ± 1.55248 (95% CI)
E[ ≤ 120;500] (max:hits) under HitWell                                   39.018 ± 0.24288 (95% CI)
E[ ≤ 120;500] (max:resets) under HitWell                                  0.022 ± 0.0129013 (95% CI)
Pr[ ≤ 120;500] (<> resets > 0) under HitWell                         0.0191072 ± 0.0121748 (95% CI)
strategy EMekSwingWell = minE (swings + resets*1000 ) [ ≤ 120] {} → {m*g*p + 0.5*m*v*v}: <> time ≥ ...
E[ ≤ 120;500] (max:swings) under EMekSwingWell                          146.402 ± 2.33689 (95% CI)
E[ ≤ 120;500] (max:hits) under EMekSwingWell                             44.832 ± 0.329089 (95% CI)
E[ ≤ 120;500] (max:resets) under EMekSwingWell                            0.024 ± 0.0134612 (95% CI)
Pr[ ≤ 120;500] (<> resets > 0) under EMekSwingWell                    0.030962 ± 0.0155714 (95% CI)
strategy EMekHitWell = minE (hits + resets*1000 ) [ ≤ 120] {} → {m*g*p + 0.5*m*v*v}: <> time ≥ 120
E[ ≤ 120;500] (max:swings) under EMekHitWell                            159.932 ± 2.05187 (95% CI)
E[ ≤ 120;500] (max:hits) under EMekHitWell                               41.436 ± 0.29184 (95% CI)
E[ ≤ 120;500] (max:resets) under EMekHitWell                              0.018 ± 0.0116935 (95% CI)
Pr[ ≤ 120;500] (<> resets > 0) under EMekHitWell                     0.0250274 ± 0.0139951 (95% CI)
strategy EMekPSwingWell = minE (swings + resets*1000 ) [ ≤ 120] {} → {p, m*g*p + 0.5*m*v*v}: <> tim ...
E[ ≤ 120;500] (max:swings) under EMekPSwingWell                         187.124 ± 1.61312 (95% CI)
E[ ≤ 120;500] (max:hits) under EMekPSwingWell                            41.296 ± 0.298995 (95% CI)
E[ ≤ 120;500] (max:resets) under EMekPSwingWell                                               ≈ 0
Pr[ ≤ 120;500] (<> resets > 0) under EMekPSwingWell                  0.0151755 ± 0.0107594 (95% CI)
strategy EMekPHitWell = minE (hits + resets*1000 ) [ ≤ 120] {} → {p, m*g*p + 0.5*m*v*v}: <> time ≥ ...
E[ ≤ 120;500] (max:swings) under EMekPHitWell                           263.428 ± 1.36982 (95% CI)
E[ ≤ 120;500] (max:hits) under EMekPHitWell                              38.852 ± 0.27503 (95% CI)
E[ ≤ 120;500] (max:resets) under EMekPHitWell                                                 ≈ 0
Pr[ ≤ 120;500] (<> resets > 0) under EMekPHitWell                             ≤ 0.00735061 (95% CI)
strategy EMekVPSwingWell = minE (swings + resets*1000 ) [ ≤ 120] {} → {p, m*g*p + 0.5*m*v*v}: <> ti ...
E[ ≤ 120;500] (max:swings) under EMekVPSwingWell                        145.656 ± 1.77239 (95% CI)
E[ ≤ 120;500] (max:hits) under EMekVPSwingWell                           39.168 ± 0.260968 (95% CI)
E[ ≤ 120;500] (max:resets) under EMekVPSwingWell                                              ≈ 0
Pr[ ≤ 120;500] (<> resets > 0) under EMekVPSwingWell                          ≤ 0.00735061 (95% CI)
strategy EMekVPHitWell = minE (hits + resets*1000 ) [ ≤ 120] {} → {p, m*g*p + 0.5*m*v*v}: <> time > ...
E[ ≤ 120;500] (max:swings) under EMekVPHitWell                          310.156 ± 2.10456 (95% CI)
E[ ≤ 120;500] (max:hits) under EMekVPHitWell                             38.316 ± 0.251862 (95% CI)
E[ ≤ 120;500] (max:resets) under EMekVPHitWell                                                ≈ 0
Pr[ ≤ 120;500] (<> resets > 0) under EMekVPHitWell                            ≤ 0.00735061 (95% CI)
```

Figure 5: Learning queries.