The Oblique Throw

2022-03-20

What is the project about?

When we throw a ball (ignoring wind resistance) we know the only acceleration acting upon the ball is the gravitational constant after it leaves our hand.

The simulations

For the simulations I used the following equations to get the lenght travelled by the ball

$t = solve(-h = v \cdot \sin(\alpha) \cdot x - \frac{1}{2} \cdot g \cdot x^2)$

$s_0 = v \cdot \cos(\alpha) \cdot t$

Using this I wrote the following python code to simulate the throws

```
import math
import sympy

def calc_dist(v, a, h = 10):
    """
    Take in an velocety, angle and a height and calculate the distance it will travel
    """
    g = 9.82 # Gravatational constant
    x = sympy.Symbol('x')

    # Left side of the equatoin
    c1 = -1 * h

    # Left side of the equation
    c2 = v * math.sin(math.radians(a)) * x - 1/2 * g * x**2

    # Solve it, we can discard the negative solution
    t = max(sympy.solve(sympy.Eq(c1, c2), (x,)))
    s = v * math.cos(math.radians(a)) * t

    return s
```

Here is an example

```
print(calc_dist(15, 10), # Velocety: 15 m/s, angle: 10 degree
      calc_dist(20, 15), # Velocety: 20 m/s, angle: 15 degree
      calc_dist(5, 25)) # Velocety: 5 m/s, angle: 25 degree
```

```
## 25.3608136859927 39.5736277406020 7.51523973272992
```

I wrote a script to run this 22860 times but I wont go over that in this document, its linked as `main.py`

Visualising the data

Lets first take a look at what type of data we are working with here

```
summary(data)
```

```
##       vel             ang             dist          
##  Min.   : 0.2   Min.   : 0.0   Min.   :  0.00505  
##  1st Qu.:12.8   1st Qu.:22.0   1st Qu.: 15.00609  
##  Median :25.5   Median :44.5   Median : 46.72623  
##  Mean   :25.5   Mean   :44.5   Mean   : 66.86415  
##  3rd Qu.:38.2   3rd Qu.:67.0   3rd Qu.:101.62410  
##  Max.   :50.8   Max.   :89.0   Max.   :272.61055
```

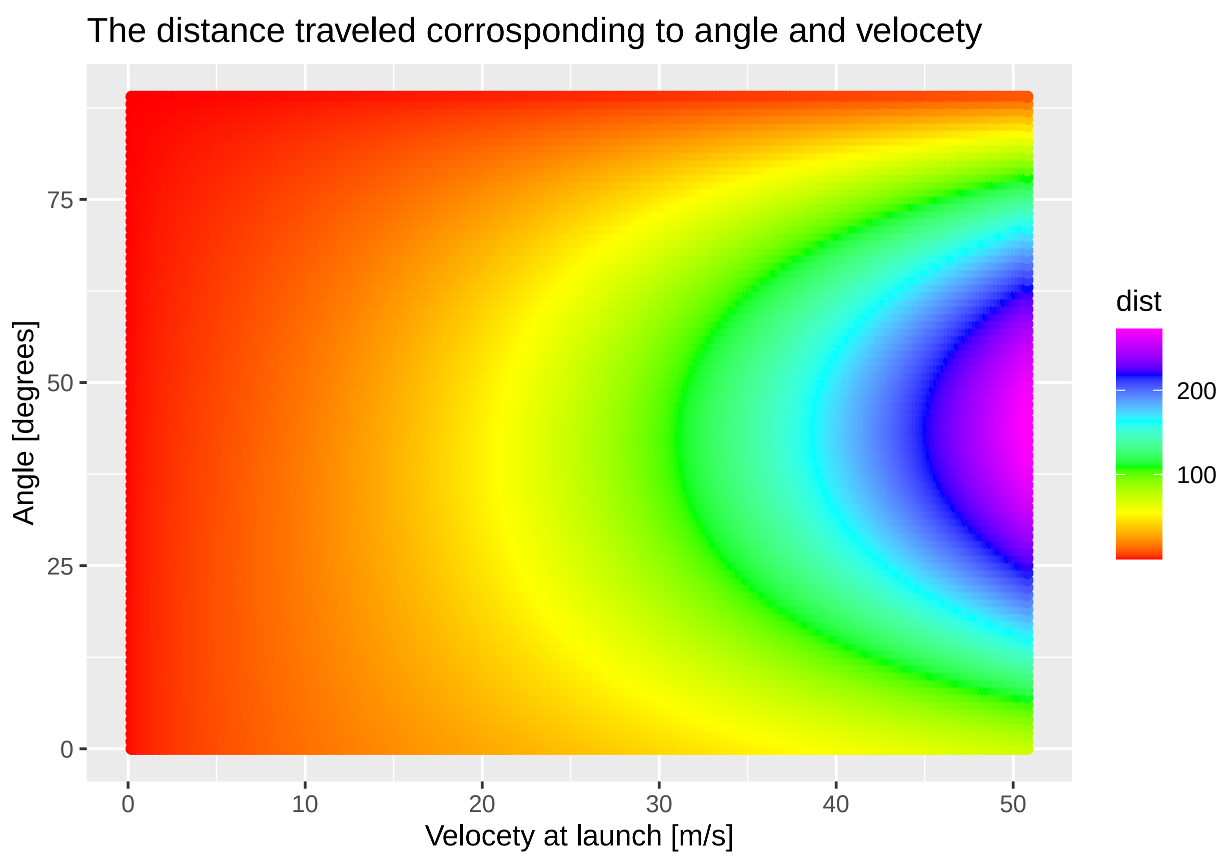
From taking a look at the summary we can see that the higest distance we reached was 272 meters with an average of 66 meters

```
str(data)
```

```
## spec_tbl_df [22,860 x 3] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ vel : num [1:22860] 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 ...
##  $ ang : num [1:22860] 0 1 2 3 4 5 6 7 8 9 ...
##  $ dist: num [1:22860] 0.285 0.285 0.285 0.285 0.285 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   vel = col_double(),
##   ..   ang = col_double(),
##   ..   dist = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

From the str function we get a preview of our data and we find that we have 22860 datapoints

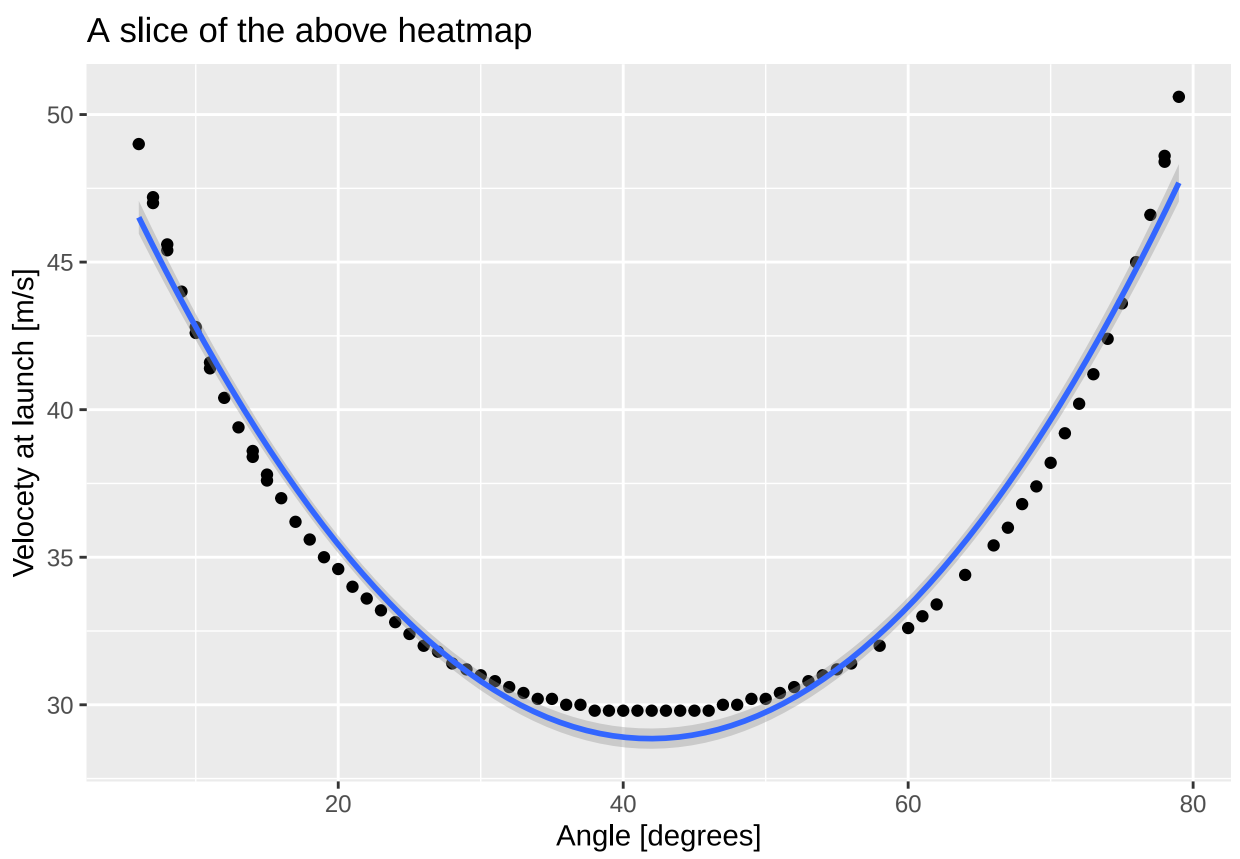
Why not try and plot it?



Now what can we do with this data?

Lets try to isolate a range of distances and with this range we can also try to use a regression

Here i got all the datapoints with a dist value between 99 and 100m



Here we got all points where the distance is between 100 and 99,

From that parabalae we find the fomula to be `f(x) = 0.01369*x²-1.14801*x+52.91235` with a r^2 value of 0.975, so quite a nice fit

My theroy is that we can calculate the top of the parabola and find the optimal angle for all velocities when the height is 10

Using the formula we can find the x corodinate of the parabalae

$T_x = \frac{-b}{2 \cdot a}$

```
a <- 0.01369 # To get the point we are looking for we only need the a and b constants
b <- -1.14801

print((-1*b)/(2*a)) # -b/2a
```

```
## [1] 41.92878
```

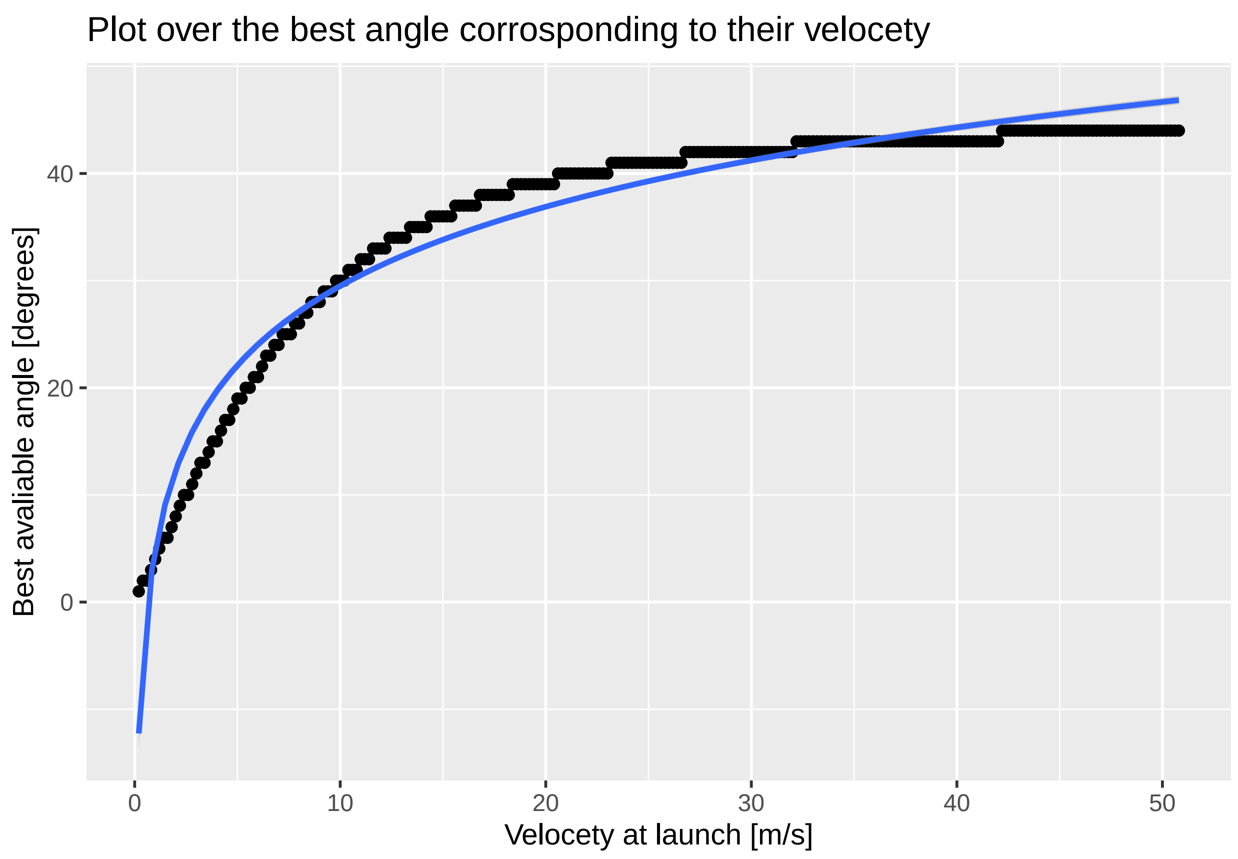
So the optimal angle for the oblique throw if my theroy is right at a height of 10m would be 41.93 degree

```
bestData <- data %>% # Find the optimal angle in the specified velocety
  group_by(vel) %>%
  mutate(x = max(dist)) %>%
  mutate(bestAng = as.double(subset(data, dist == x)[2]))

bestData <- bestData[!duplicated(bestData[c('vel')]), ] # Remove duplicates of velocety

ggplot(data = bestData, aes(x = vel, y = bestAng)) +
  geom_point() + # Create a pointplot
```

```
labs(title = "Plot over the best angle corrosponding to their velocety",
     x = "Velocety at launch [m/s]",
     y = "Best avaliable angle [degrees]"
) +
geom_smooth(method = glm, formula = y ~ log(x))
```



So in fact we have a concave power function with the formula

$f(x) = b \cdot x^a$

Running a quick analasys on the data we can check the estimate we made erlier

```
summary(bestData[,5])
```

```
##     bestAng
##  Min.   : 1.00
##  1st Qu.:34.00
##  Median :41.00
##  Mean   :36.33
##  3rd Qu.:43.00
##  Max.   :44.00
```

As we can see the median is 41 dagree, and if we had more datapoints i belive that we would get closer to our calculated estimate of 41.93

Tho it is worth mentioning that i was wrong in the angle being the same for all veloceties, as we can clearly see there is a concave power function developing, so the faster the speed the closer we will get to flattening out at arround 45 degree