A semi-probabilistic solution for killer litter origin prediction

Jeremy Wang

September 2024

1 Introduction

1.1 Preface

"We may regard the present state of the universe as the effect of its past and the cause for the future." The French mathematician Laplace encapsulates the predictive power and associated with physical laws that have governed our understanding of the natural world since Sir Isaac Newton. Laplace also mentions that, given an intellect which knew of "all of the forces" within and all "mutual positions of the beings" that our universe is composed of, as well as sufficient ability to process that data, would be able to predict the future, or unveil the past with certainty.

Though the work of physicists in quantum theory during the 20^{th} century has put this deterministic model of the universe to question, it largely remains true for domains where classical physics is still considered a good enough approximation of reality. Given sufficient knowledge of the required parameters, and we should be able to predict with certainty the events of the future, and the events of the past. It is with this understanding that we present the contents of this report, one which explores our options when we lack this knowledge.

1.2 The Problem

We are given a simple problem in physics modelling. Given that an object (in this case a flowerpot or bottle) was found smashed at a particular point on the ground floor, we suspect that it was thrown from height from a particular window or door along a HDB flat. We want to predict which window or door the object was thrown from.

2 A deterministic model

2.1 Quantifying the problem

1. For all sections onwards, notation from this section will be used when referring to the modelling set-up used.

2. Space: We represent the space around the HDB investigated as a 3 dimensional Euclidean space $\mathbb{R}^3$ where all points can be represented as $p = (p_x, p_y, p_z) \in \mathbb{R}^3$, whereby p_x, p_y, p_z are the x, y, z coordinates of the point respectively.

3. Axes: The y axis represents the vertical direction, whereby $y = 0$ represents ground level. The x axis represents the horizontal direction along the face of the HDB flat, whereby $x = 0$ represents the left boundary of the HDB flat (when viewed from the perspective of the thrower), $x = 100$ represents the right boundary of the HDB flat. The z axis represents the horizontal direction leading away from and towards the HDB, with $z = 0$ representing the xy plane across the face of the HDB.

4. Discrete points:
   1. Origins: Given the description of the problem, we represent the possible origin points of an object thrown from the HDB as a set of points $O = \{o_1, o_2, ...\}$, whereby $\forall o_i \in O$, o_i has the form $(o_{i_x}, o_{i_y}, 0)$. All points in O are assumed to be on the xy plane $z = 0$, such that objects are dropped from exactly the edge of the parapet. The x and y values of these points correspond to the position of a particular window or door on a particular floor such that $\forall o_i \in O, o_{i_x} \in \{0, 24, 32, 40, 48, 56, 64, 72\}$, $o_{i_y} \in \{y | \forall y, y = 6.25 + 3.6 * f, f \in \{1, 2, ..., 25\}\}$.

   2. Landing points: All possible landing points of objects are represented as the set of points $L = \{l_1, l_2, ...\}$. $\forall l_i \in L$, l_i has the form $(l_{i_x}, 0, l_{i_z})$. All points in L are assumed to be on the xz plane $y = 0$, such that all landing points are at ground level. The range of the x and z coordinates of l_i are as follows: $l_{i_x} \in \mathbb{R}$, $l_{i_z} \in \mathbb{R}^+$.

5. Physical parameters:
   1. Throw velocity (T_v): The initial velocity at which the object was thrown, measured in ms^{-1}. $T_a \in \mathbb{R}^+$. (Assuming no backward throws are possible)

   2. Throw angle (T_a): The angle at which the object was thrown, measured in radians. $T_a \in [-\frac{\pi}{2}, \frac{\pi}{2}]$. If $T_a < 0$, then the throw is downwards, if $T_a = 0$, the throw is straight forward, if $T_a > 0$, then the throw is upwards.

   3. Facing angle (F_a): The angle the thrower is facing when throwing the object, measured in radians. $F_a \in [0, \pi]$. If $0 \leq F_a < \frac{\pi}{2}$, then the person is facing right. If $\frac{\pi}{2} < F_a \leq \pi$, then the person is facing left. If

$F_a = \frac{\pi}{2}$, the person is facing straight forward.

4. Wind speed (W_s): The speed of the wind, measured in ms^{-1}. This is a scalar parameter with no directional component.

5. Wind direction (W_d): The direction of the wind. $W_d \in \{-1, 1\}$. If $W_d = -1$, then the wind is blowing towards the left, if $W_d = 1$, then the wind is blowing towards the right.

6. Mass of object m: Mass of the object thrown, measured in kg. $m \in \mathbb{R}^+$.

7. Surface area S_a: Surface area of the object thrown, measured in m^2. $S_a \in \mathbb{R}^+$.

8. Drag coefficient D_c: Drag coefficient of the object thrown, dimensionless. $D_c \in \mathbb{R}^+$.

6. Now, we are ready to quantify the problem. Given that an object landed at a landing point $l \in L$ and a set of physical parameters $\{T_v, T_a, F_a, W_s, W_d, m, S_a, D_c\}$, predict the origin point $o \in O$ that this object was thrown from.

2.2 A deterministic model

1. Position computation:
After quantifying the problem, we can now construct a physics model in order to solve our problem in a deterministic manner. We choose to model the object as it flies in a trajectory through the air in a step-wise manner, updating its position every 0.001 (chosen to balance between computational cost and accuracy) seconds based on a $3-D$ velocity vector. For instance, suppose the object is at position (i_x, i_y, i_z) at a particular step with a velocity vector (v_x, v_y, v_z), then the object will be at $(i_x + 0.001v_x, i_y + 0.001v_y, i_z + 0.001v_z)$ at the next step. The simulation begins when the object is thrown from a position (x_s, y_s, z_s) and continues until the y coordinate of the object is less than or equal to 0, indicating that the object has reached ground level.

2. Velocity vector and acceleration computation:
The velocity vector is a $3D$ vector (v_x, v_y, v_z) that represents the velocities of the object along the x, y, z axes. Given an initial throw velocity T_v, we decompose the value into its x, y, z components with the following formulae to obtain the initial velocity vector:
$v_x = T_v \cos(T_a) \cos(F_a) + W_s \cdot W_d$ (wind is added as a constant given the constraint that the wind in this problem will always be along the HDB)
$v_y = T_v \sin(T_a)$
$v_z = T_v \cos(T_a) \sin(F_a)$

We then update the velocity vector every step in our step-wise model through computation of acceleration via the effects of gravity and drag. Assume we start with a velocity vector (v_x, v_y, v_z). Then given the mass m, drag coefficient D_c and surface area S_a of the object, the drag forces are computed:
$F_D = \frac{1}{2}\rho v^2 D_c S_a$, then given that the approximate density ρ of the fluid (air) that our object is travelling through is 1.225:
$F_{D_x} = 0.6125 v_x^2 D_c S_a$
$F_{D_y} = 0.6125 v_y^2 D_c S_a$
$F_{D_z} = 0.6125 v_z^2 D_c S_a$

Then we compute the acceleration of the object:
$a_x = -\frac{F_{D_x}}{m}\frac{v_x}{|v_x|}$, where the second fraction obtains the direction which the object is moving with respect to the axis
$a_x = -\frac{F_{D_y}}{m}\frac{v_y}{|v_y|} - 9.81$, where 9.81 is the acceleration due to gravity downwards
$a_z = -\frac{F_{D_z}}{m}\frac{v_z}{|v_z|}$

After obtaining the acceleration of the object at a particular step, we can then compute the velocity vector $(v_{x_n}, v_{y_n}, v_{z_n})$ of the next step through the following formulae:
$v_{x_n} = v_x + 0.001 a_x$
$v_{y_n} = v_y + 0.001 a_y$
$v_{z_n} = v_z + 0.001 a_z$

3. Error computation:
After we terminate the step wise simulation, the object would have landed at a particular landing spot $l_s = (x_{l_s}, 0, z_{l_s})$. We then compute the $2D$ Euclidean distance between l and l_s, given by $\epsilon = \sqrt{(x_l - x_{l_s})^2 + (z_l - z_{l_s})^2}$. ϵ is then the "error" produced by this particular origin point (x_s, y_s, z_s) with respect to the actual landing point.

4. Finding the origin:
We repeat the step-wise simulation $\forall o \in O$ while keeping the set of parameters the same. The origin o_{min} which produces the smallest ϵ is then deemed the "most likely origin" of the object thrown.

3 The unknown and probability

3.1 Limitations of the deterministic model

1. The deterministic model is, as Laplace's quote suggests, an idealistic model that assumes we have sufficient knowledge of the "state" of the problem, such as the explicit values of all the required parameters. However, given the constraints of the problem, we clearly do not have such knowledge.

2. Possible solutions: One possible solution is to find an origin $o \in O$ and a set of parameters that leads to a very small ϵ. We can then try to tune the parameters manually such that we find the set of parameters that reduces ϵ close to 0.

   The problem with this approach quickly becomes apparent. Aside from the set of origin points O, many of the unfixed parameters (parameters excluding mass, drag coefficient and surface area of object) are continuous. There is an infinite number of values that they are able to take, each producing a slightly different ϵ. In a complex system such as our step wise simulation, it is very difficult to optimize for parameters that would minimize ϵ.

   Furthermore, this does not solve a more fundamental problem in our analysis. For instance, how can we be sure that a person threw the object at a particular velocity, angle at a time with a particular wind speed? We are unable to justify why we select specific values for parameters except to minimise ϵ, which defeats the purpose of modelling, especially given that multiple different origins with multiple different sets of parameters can produce ϵ of similar magnitude. How can we be sure that it was a throw at one specific origin with one specific set of parameters that resulted in the object's landing point?

   The answer is, we can't! Which leads us to a probabilistic interpretation of the problem.

3.2 Modelling probabilities

1. While we cannot say for certain that a person threw the object at a specific angle, velocity at a particular wind speed, we can make statements such as: The person **likely** threw the object straight forward, with strength similar to the **average** human throw. The terms "likely" and "average" then suggest a probabilistic element to our problem. Therefore, instead of trying to find an origin and a set of corresponding parameters that produces the smallest ϵ, we now try to find a point that produces the smallest ϵ, given the way our parameters tend to behave in the real world, which has an element of probability. We call this the "most likely" origin.

2. One way to do so is via the Monte Carlo method. The Monte Carlo method defines a domain of possible parameters (which we have done so in section 2.1), generates inputs randomly from a probability distribution over the domain, performs a deterministic computation over the set of generated inputs (which we can do via our deterministic model) and aggregates the results based on a chosen metric. The Monte Carlo method is often used to simulate complex systems in real life where significant uncertainty in inputs, such as in our case right now, where we do not have the explicit

values of many parameters to work with. It is also useful where the problem can have a probabilistic interpretation, which we have demonstrated above, whereby instead of making declarative statements with certainty, we make statements that refer to a certain probability for a certain event.

4 Our solution

4.1 Adapting the Monte Carlo method

1. In order to adapt the Monte Carlo method for our problem, we construct the general architecture of our solution as such:
   1. For unknown parameters, specifically T_v, T_a, F_a, W_s, W_d, we introduce probability by assuming they are random variables with a particular distribution.
   2. We obtain a single sample for T_v, T_a, F_a, W_s, W_d and, using the deterministic model we had previously, we compute the origin for this specific sampled set of parameters.
   3. We choose a metric, in this case a distance weighted score to assign to the origin obtained for a single set of sampled parameters. We then aggregate this score.
   4. We repeat step 2 and 3 for many times, and by the law of large numbers, the aggregated score of each origin over many throws attempted with many sets of parameters will reflect the probability of each origin being the actual origin of the throw that resulted in the object landing at the specified landing point.

4.2 Quantitative specifics

1. Step 1: Choosing distributions for parameters and justification
   1. $T_v \sim \gamma(x; \alpha, \beta)$: Throw velocity is modelled as a gamma random variable due to the domain of T_v being only positive real numbers. Similarly, the value of a sample taken from a gamma distribution will always be greater than 0. Furthermore, gamma distributions are a particular case of the normal distribution, which we expect the velocities of human throws (given enough samples) to follow, since its likely that most throws are likely not too hard, or not too soft.

   2. $T_a \sim \mathcal{N}(\mu, \sigma^2)$: Throw angle is modelled as a normal random variable. In particular, it is modelled as a normal distribution with mean 0 as we expect most throws to be either straight forward or slightly up/down. We expect only a small fraction of the throws to be at more extreme angles. Given that our domain for T_a is $[-\frac{\pi}{2}, \frac{\pi}{2}]$, we want the vast majority of our samples to fall within this range. Given that 0.997 of values fall within 3 standard deviations of the mean, we choose our standard deviation to be $\sigma = \frac{\pi}{6}$, such that 99.7 percent of all samples fall within our

domain.

3. $F_a \sim \mathcal{N}(\mu, \sigma^2)$: Similar to throw angle, facing angle is modelled as a normal random variable. We expect most people to throw while facing directly forwards, or at a slight angle towards the left or right. We expect only a small portion of people to be throwing the object while at a large angle towards either direction. In particular, we set the mean $\mu = \frac{\pi}{2}$, and since our domain for F_a is given as: $[0, \pi]$, our standard deviation is once again $\sigma = \frac{\pi}{6}$ to ensure that 99.7 percent of our samples fall within our domain.

4. $W_s \sim Wei(b, c)$: Wind speed is modelled as a Weibull random variable after some research on related materials, including a study conducted in Singapore by researchers looking to estimate wind speed for wind energy purposes.

5. $W_d \sim Bernoulli(p)$: Wind direction is modelled as a Bernoulli random variable. In particular, we expect the wind to be blowing left or right an equal proportion of times given enough real life samples, and thus, we choose $p = 0.5$.

2. Step 2 and 3: Process for one iteration
1. After choosing the distributions for the parameters, we generate a set of parameters by taking a single sample from the parameters' respective distributions. Given our distributions, sets of parameters representing what we think is more likely to happen in real life, such as the person facing straight forward, throwing straight forward etc. are more likely to be generated than sets of parameters that represent what we think is less likely to happen in real life, such as a person throwing directly upwards.

1b. This means that, across many iterations, the iterations themselves represent what we think is more likely to happen in real life, with throws that we think are more likely to happen occurring more times in our simulation.

2. We then plug this set of parameters into our deterministic model described in section 2.2 and compute the origin point given this set of parameters and the designated landing point.

3. The origin point computed will also have an associated ϵ value. We use the ϵ value to compute a distance adjusted score s_{DA} for this origin point using the exponential decay function: $s_{DA} = e^{-}\lambda\epsilon$. The exponential decay function ensures that origins and parameters that produce a smaller ϵ are weighted more heavily. λ is a parameter that decides how much we want to punish deviation from the actual landing point, with larger λ values penalizing distance from the actual landing point more heavily.

4. If there were previous iterations that resulted in the same origin already having a distance adjusted score associated with it, we add this new distance adjusted score to the previously aggregated score. Else, we save the origin and its associated s_{DA}, so that future iterations which result in the same origin can add to the distance adjusted score.

3. Step 4: Aggregation
1. We choose the number of iterations we wish to run. Empirical experimentation with this model in Unity suggests that results generally converge once more than 25000 iterations are run, with further iterations rarely affecting the result. The maximum magnitude of iterations that was able to run within reasonable time on available devices was 10^5.

2. After the completion of all iterations, the k points with the k largest aggregated s_{DA}s are the k "most likely" origin points. The probability of each of these origin points o_i being the actual origin points is given by: $\frac{s_{DA}(o_i)}{s_{DA}(Total)}$. The program saves the:
i) Mean parameters of the "most likely" origin across all iterations

ii) The "local optima" parameters, parameters which produced the smallest ϵ given the "most likely" origin
iii) The "global optima" parameters, parameters which produced the smallest ϵ across all throws across all iterations

Trajectory simulation can then be performed using these parameters.

5 Alternatives, limitations and further work

5.1 Alternative metrics

1. Threshold for landing points: We recognise that in our current solution, there can be some cases whereby a "close second" is not accounted for in our modelling. Hence, we have prepared an alternative model, whereby instead of only accounting for one origin point for each set of sampled parameters, we consider all origin points which produce an $\epsilon \leq k$, where k is a arbitrary threshold. (This means that all origin points that produce a landing point within k meters of the actual landing point given a specific set of sampled parameters will be assigned a s_{DA} value to be aggregated.) In practice however, across many rounds of testing, this method did not produce results that were significantly different from our current model.

2. No thresholds or limits: We have also considered a model that accounted for every single throw from every single origin for every set of sampled parameters, assigning all of them a s_{DA} value to be aggregated. However, since computation of s_{DA} already significantly penalizes large ϵ vales, we

decided against this method as it would likely add significant computational cost for very little difference in results, as origin points that were not already accounted for in our threshold model (which does not produce a significant difference in results) result in large ϵ values will have a very small s_{DA} score and hence will not affect the aggregated s_{DA} scores significantly.

5.2 Limitations

1. The limitations of semi-probabilistic model employing the Monte Carlo method are obvious:
   1. Foremost, we are making big assumptions regarding the distribution of the unknown variables. It is entirely likely that empirical studies will find that our assumptions of the distributions governing these unknown variables are incorrect. In order to manage this limitation, our product allows for users to customize the parameters for the distributions for the unknown variables should they have more accurate values than the default ones used.

   2. Secondly, while the Monte Carlo method in general works for complex systems such as the problem presented. There is a nonzero chance that for this particular problem, it produces a poor approximation of real life. This can be due to specific interactions between different parameters, human behaviour unaccounted for amongst people who throw killer litter etc. In order to minimize the chances of this occurring, this model will need to be augmented and optimized with help with other unrelated models that verify its accuracy and assumptions made.

   3. Finally, the computational costs of Monte Carlo simulations are enormous and we have only been able to simulate up to a magnitude of 10^5 iterations thus far. Simulations with iterations that have magnitudes orders larger could produce different conclusions compared to the ones we have now.

5.3 Further work

1. Integration with Cloud Virtual Machines: Given the huge computational costs of Monte Carlo simulations, we are looking to integrate our model with cloud virtual machines such that we can offload demanding computations to them. This will allow us to run much more iterations, where we can be more certain about our results and conclusions.

2. Cross validation with model based on shattering physics: Thus far, our model has been reliant on trajectory simulation. We are in the process of constructing a separate model that estimates an origin based upon the energetics of the object (such as a glass bottle) hitting the ground and

shattering upon impact. The distance whereby we can find fragments from the landing point could be used to estimate the energy the object possessed when hitting the ground, and hence, be used to estimate a possible origin point of it. An alternative model can be used to verify that the results produced by this model are a good approximation of reality.

3. Support for continuous values for origin points: In real life, origin points of killer litter are unlikely to be discrete points along the face of the HDB. Instead, people can throw from anywhere along the parapet, and hence, we are working on making o_x, the x coordinate of origin points a continuous random variable.

4. Integration with machine learning tools: As of right now, we lack related datasets to train any machine learning model on to augment our existing model based on mathematical and statistical methods. But we will be on the lookout for related datasets that can allow us to make use of machine learning tools to improve our current model.