Dynamic Modeling, Simulation, and Optimization of the Trebuchet

Christopher Brown – er6232
Punit Purohit – gd7198
Yongpan Ma – gk8458
Yogesh Karande – gj5334

Submitted to:
Dr. Valery Pylypchuk
Department of Mechanical Engineering
Wayne State University
Detroit, MI

Executive Summary

The trebuchet has adequately cemented itself in history as one of the most well-known and successful war machine of its time. In addition, it is widely regarded as being well ahead of its time in its technical complexity. Moreover, the dynamics and physics involved in the operation and effectiveness of this kind of machine makes for a fascinating challenge in modeling, simulating, and optimizing the design. The purpose of this report is to provide a brief history of the trebuchet, and to present, in detail, the method and results of our study into modeling, simulating, and optimizing the trebuchet system. The approach used in the analysis and modeling of this machine was to utilize the simplicity of the Lagrange method. Following the modeling of the system, the software Wolfram Mathematica was used to solve all differential equations of motion in order to simulate and optimize the design of the trebuchet.

Table of Contents

List of Figures

History of the Trebuchet

The trebuchet was first invented in China around 500 to 300 B.C. and was used in Europe after 600 A.D. It replaced other forms of artillery and was used until the invention of cannons and gunpowder. The trebuchet was a very powerful replacement for catapults. Unlike catapults that use the energy of a twisted rope, a trebuchet relies on the energy provided by a falling counterweight. While the average catapult could only launch 13 to 18 kilogram projectiles, trebuchets are reported to have launched projectiles as heavy as 500 kilograms, similar in mass to a modern day compact car, to a distance of 80 meters using 30-tonne counterweight. Unlike its predecessor, the trebuchet had been designed to throw projectiles much more accurately than other medieval artillery and were often times used to attack a specific target like a portion of a castle wall. Catapults, however, were normally used to instill fear rather than destroy a target. Trebuchets were developed and optimized in their very first years by engineers – indeed the word "engineering" is intimately related to them. "In Latin and the European vernaculars, a common term for trebuchet was 'engine' (from ingenium, 'an ingenious contrivance'), and those who designed, made, and used them were called ingeniators." It took years for engineers to develop the early designs through trial and error to increase the performance and precision of the trebuchet while decreasing its total weight. [1]

Technical Introduction of the Trebuchet

The trebuchet is a simple, spectacular, and hands-on design project that can be applied to the study of projectile motion, rotational motion, and the law of conservation of energy. The trebuchet itself is a simple device, easy to build, and entirely powered by gravity. The major parts of the traditional trebuchet are shown schematically in Figure 1. The projectile is held in a sling suspended from a hook on one side of the trebuchet arm. The counterweight is hung on the opposite, typically shorter, side of the arm. The arm rotates on a pivot that is rigidly mounted onto the base. [2]

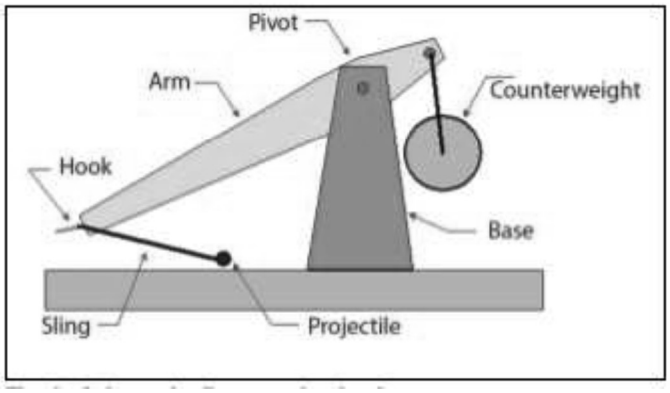


Figure 1 *Schematic diagram of the trebuchet*

To launch the projectile, the counterweight is raised until the hook rests on the ground and then released. As the counterweight falls, the arm rotates clockwise (according to the diagram in Figure 2), and the sling whips around until the projectile is released. The angle of the hook is critical here—the projectile is released when the sling and hook are aligned. The series of images in Figure 2 illustrates this principle step-by-step.

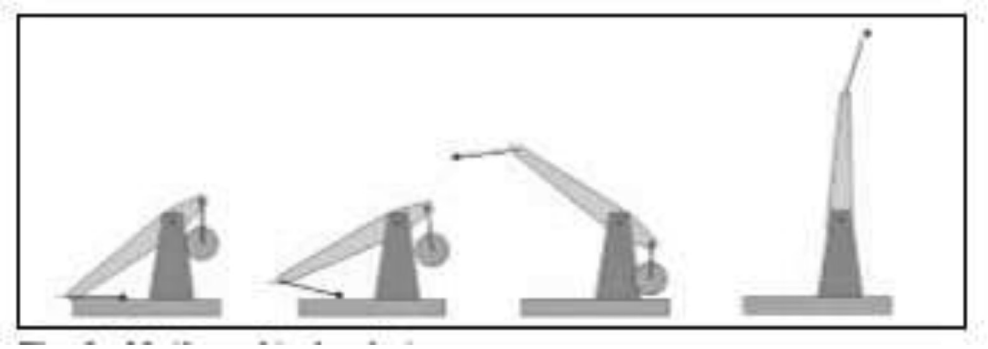


Figure 2 *Motion of the trebuchet*

The principle of the trebuchet was straightforward. The weapon consisted of a beam that pivoted around an axle that divided the beam into a long and short arm. The longer arm terminated in a cup or sling for hurling the missile, and the shorter one in an attachment for pulling ropes or a counterweight. When the device was positioned for launch, the short arm was aloft; when the beam was released, the long end swung upward, hurling the missile from the sling. Three major forms developed: traction machines, powered by crews pulling on ropes; counterweight machines, activated by the fall of large masses; and hybrid forms that employed both gravity and human power. When traction machines first appeared in the Mediterranean world at the end of the sixth century, their capabilities were so far superior to those of earlier artillery that they were said to hurl mountains and hills. The most powerful hybrid machines could launch shot about three to six times as heavy as that of the most commonly used large catapults. In addition, they could discharge significantly more missiles in a given time. [3]

A trebuchet was a kind of siege engine in the middle age that was characterized by an arm-and-sling mechanism and a heavy counterweight that stored the required energy for launching a projectile in its potential energy. The increased release height of this solution appeared to be useful today for fixed wing UAVs, because the widely used catapults, bungee starts and similar solutions feature low release height and low flight path angle during climb, that require clean, obstacle free area. However, a trebuchet releases the projectile about 5 to 10 meters above the ground, exceeding bushes, small trees, camp accessories, small buildings, etc. [4]

In terms of a "black box" analysis, a weight driven device designed to throw a projectile is about as easy an item to describe as is possible. The input energy is simply the potential energy of the driving mass, assumed lifted to some height h. Then, the output is simply how far a given projectile mass can be thrown. If the projectile departs with a velocity of v_0 at an angle α from the horizontal, and if air resistance is neglected, the range R at impact can be represented as

$$R = 2v_0^2 \, sin(\alpha) \, cos\left(\frac{\alpha}{g}\right)$$

where g is the acceleration due to gravity. The maximum range obviously occurs for a launch angle of 45˚, so this formula simplifies to

$$R_{max} = \frac{v_0^2}{g}$$

If all of the potential energy is assumed converted to kinetic energy, the condition at which the maximum range occurs can be described as

$$2h\left(\frac{m_1}{m_2}\right)$$

where m_1 and m_2 are the driving and projectile masses, respectively. The actual range achieved by any gravity driven engine divided by this theoretical maximum is usually termed the range efficiency (RE), which seems to be a very appropriate metric and will be the performance measure used herein. [5]

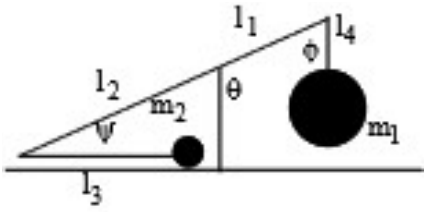


Figure 3 *Parameters of the trebuchet*

If one simply imagines the trebuchet as previously depicted, but with the driving mass m_1 suspended by a pivoting rod and a sling system for holding the projectile, one has a classical trebuchet. Interestingly, these simple modifications to the original design lead both to a challenge of analysis and a startling performance improvement. To consider the analysis first, just the addition of the pivoting driving weight renders the system almost impossible to analyze via torques, moments of inertia and angular accelerations. An appropriate analysis technique utilizes the method of Lagrange, and is therefore completely energy based.

Approach

To simplify the simulation of the trebuchet, we have reduced the problem by considering following assumptions.

- There is no air friction.
- The beam has no mass and hence does not add to moment of inertia.
- The counter weight and the projectile are assumed to be point masses.
- The sling is considered rigid and massless.

Even under the aforementioned assumptions formulation of equation of motion through Euler's equation is challenging as the contact with ground creates additional constraint which restricts the system to two degrees of freedom (under the assumption that the sling is rigid and has no mass), and as the projectile detaches from ground the system gets another degree of freedom.

To solve this problem, we were required to break it into two different stages; one during the ground constraint, and one after the ground constraint. Because of this, we formulated two sets of differential equations. The solution of the first stage provides the initial conditions for the second

stage. Finally, merging the data from both stages delivers the complete solution. Figure 4 shows the breakdown of these two stages.

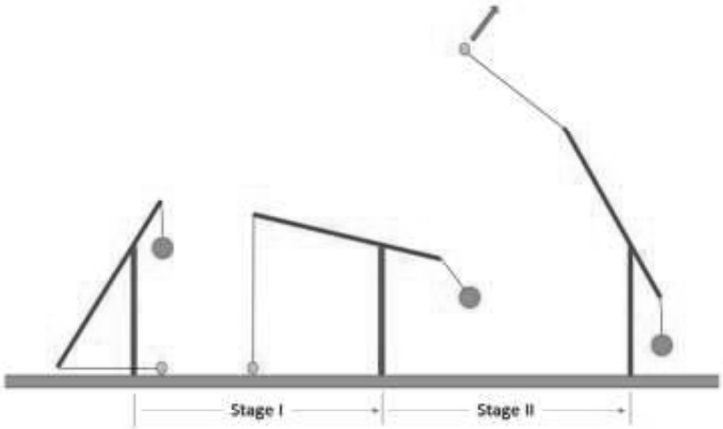


Figure 4 *Two-stage approach to the trebuchet*

Figure 5 below shows the trebuchet diagram with the variable labeling convention used throughout our analysis, solution, and optimization.

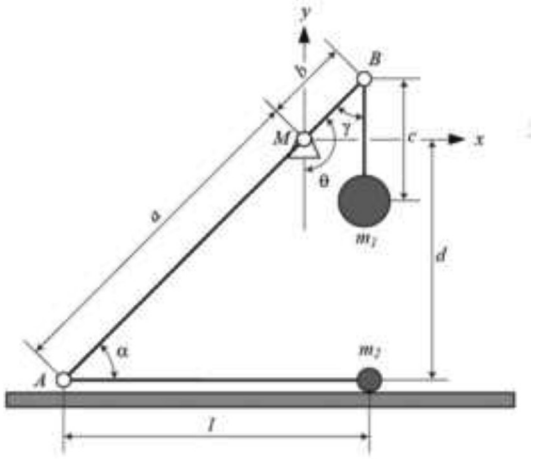


Figure 5 *Trebuchet diagram with variables labeled*

Geometric Analysis

The first step in applying the Lagrange method is to geometrically define the positions of both masses in the two dimensional plane. According to the variable labeling convention in Figure 5, the positions of these masses are defined as

$$x_1 = b\sin(\theta) - c\sin(\theta + \gamma)$$
$$y_1 = -b\cos(\theta) + c\cos(\theta + \gamma)$$
$$x_2 = -a\sin(\theta) + l\sin(\theta - \varphi)$$
$$y_2 = a\cos(\theta) - l\cos(\theta - \varphi)$$

where x_1 and y_1 are the horizontal and vertical positions of m_1, and x_2 and y_2 are the horizontal and vertical positions of m_2, respectively.

Equations of Motion during Ground Constraint

Now that the positions of both masses are defined, we can find the kinetic and potential energies of the system during the first stage of motion during the ground constraint. The kinetic energy of the system can simply be defined as the sum of the respective kinetic energies of each mass. This equation then takes the form of

$$T = \frac{1}{2}m_1\left({\dot{x}_1}^2 + {\dot{y}_1}^2\right) + \frac{1}{2}m_2\left({\dot{x}_2}^2 + {\dot{y}_2}^2\right)$$

In the same fashion as the kinetic energy, the potential energy of the system can be defined as the sum of the respective potential energies of each mass, and is therefore defined as

$$V = m_1 g y_1 + m_2 g y_2$$

Next, the Lagrange formulation is determined simply using the following formula.

$$L = T - V = \frac{1}{2}m_1\left({\dot{x}_1}^2 + {\dot{y}_1}^2\right) + \frac{1}{2}m_2\left({\dot{x}_2}^2 + {\dot{y}_2}^2\right) - m_1 g y_1 - m_2 g y_2$$

Lastly, we can determine the equations of motion using the Euler-Lagrange equations. During the ground constraint, we only have two generalized coordinates involved, and therefore only two equations of motion. The Euler-Lagrange equations used for this are shown below.

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = 0$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\gamma}}\right) - \frac{\partial L}{\partial \gamma} = 0$$

Using the *NDSolve* command in Mathematica, we were able to determine a numerical solution for this first stage during the ground constraint.

Initialization for Equations of Motion after Ground Constraint

The numerical data obtained for the first stage of the trebuchet motion describes the behavior of both masses while the projectile mass is still dragging on the ground. This solution only provides us with the motion of the two masses right up until the projectile mass is about to leave the ground. The data for both the generalized coordinates and generalized velocities at this moment were used as the initial conditions for the second stage after the ground constraint.

Equations of Motion after Ground Constraint

In the exact same way as outlined in the methodology and equations during the ground constraint, the kinetic and potential energies were calculated, along with the Lagrange.

The difference here is that now we have introduced the third generalized coordinate, and therefore a third equation of motion. That third equation of motion is defined using the Euler-Lagrange equation below.

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\alpha}}\right) - \frac{\partial L}{\partial \alpha} = 0$$

As previously mentioned, when using the *NDSolve* command in Mathematica, the initial conditions were defined as the final conditions of stage one of the trebuchet motion. By combining the numerical solutions from both stages, the complete solution of the trebuchet motion is captured.

Data Interpretation

In order to understand and quantify the system performance, the solution data needed to be interpreted by introducing a couple variables.

The first of these is release velocity. The sling on the end of the trebuchet releases upon an angle α, at which point the counterweight does no more work on the projectile. The velocity of the payload at that moment is called release velocity. By calculating the velocity in the x and y directions (V_x and V_y, respectively), total release velocity can be calculated.

The second variable introduced is the release angle. One can change this release angle by adjusting the angle of the holding pin at the end of the lever arm. The more in line with the lever arm the holding pin is, the smaller the release angle α is.

The last variable introduced is the range. The final displacement of the projectile in the horizontal and vertical direction after being released from the trebuchet is considered to be the horizontal and vertical range respectively. In our case, we will only be considering the horizontal range.

Defining the Initial Parameters

The initial parameters for the initial simulation were defined based on research on the topic of trebuchet optimization.

The first parameter we needed to define was the location of the fulcrum. By placing the fulcrum much closer to the counterweight end, a higher linear velocity can be achieved by the projectile. This becomes simple lever physics. However, the fulcrum can't too close to the counterweight, or the mechanical advantage will begin to dwindle. There will also come a point where

increasing the weight of the counterweight will do no good other than putting extra strain on the trebuchet.

The second parameter to define was the length of the payload arm. According to Trevor English in his paper *Designing a Trebuchet – Optimizing Weight and Length*, the length of the payload arm should be 3.75 times longer than the counterweight arm. [6]

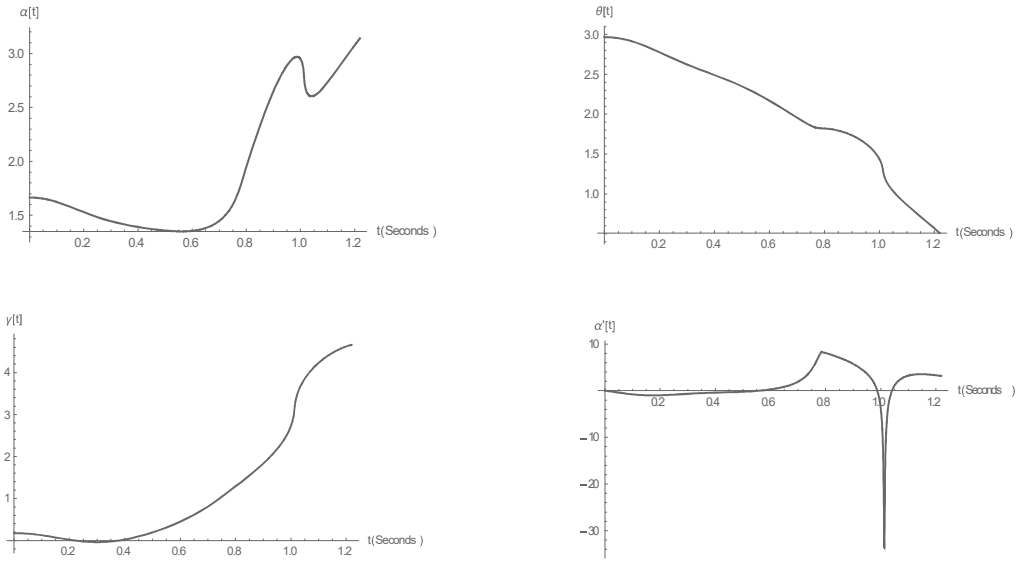
The third parameter to define was the initial angle between the counterweight arm and the support. This was set to be 45 degrees at launch. [6]

The fourth parameter to define was the beam ratio. Beam ratio is the ratio of length of the payload arm to counterweight arm. There have been many trebuchet designs that used ratios closer to 4:1 or even 5:1. Increasing the ratio increases the launch velocity of the projectile, but it also means the counterweight must be much larger to provide the necessary force.

The fifth parameter to define is the mass ratio. The generally accepted ratio is 133:1, which means the counterweight needs to be 133 times as heavy as your intended projectile.

Initial Simulation

Using the defined parameters aforementioned in the previous section, the plots below show the results of the initial simulation. Included in these seven plots are the time histories of each of the three generalized coordinated, time histories of each of their respective generalized velocities, and a parametric plot of the mass positions.



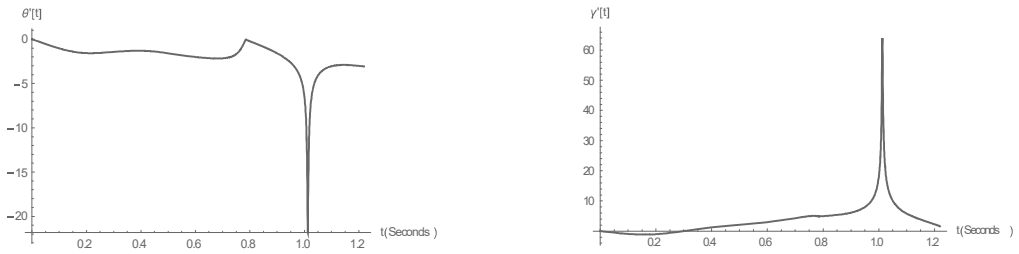


Figure 6 *Time history plots of the position and angular velocities of angles α, θ, and γ.*

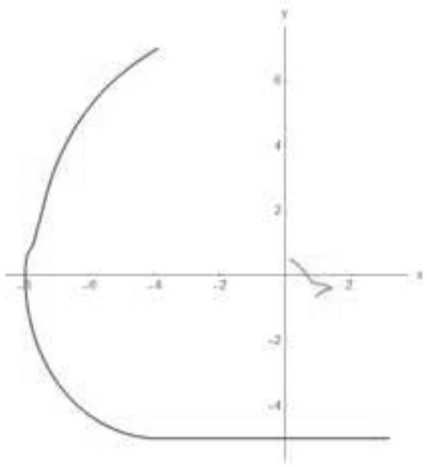


Figure 7 *Parametric plot of the positions of m_1 and m_2*

This simulation yielded a range of 121.193 meters, a release angle of 29.579 degrees, and a release velocity of 37.2118 meters per second.

Validation

After the first successful simulation results, it was important that we validate our simulation method in such a way that we can be sure our results are sound. To do this, we simply needed to plot the total energy from this first simulation and observe its behavior. The total energy of the system is defined as the sum of the kinetic energy and potential energy of the system.

$$E = T + V$$

Theoretically, the total energy should stay completely constant as the trebuchet progresses through its motion, as the conservation of energy principle certainly applies here. The time history of the total energy for this initial simulation is shown in Figure 8.

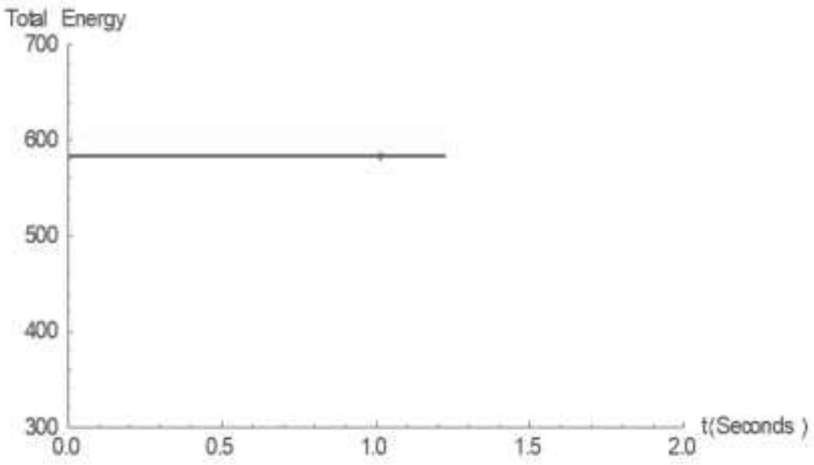


Figure 8 *Time history plot of the total energy*

This plot is obvious in showing that the total energy of the system in conserved throughout the motion of the trebuchet. This adequately validates the methodology and calculations of the simulations.

Single Variable Optimization

The approach to optimization for this trebuchet evolved as we gathered more data. The initial approach was to go through individual parameters and understand the impact of each parameter on the range of projectile. However, we moved on to studying impact of varying two, three and four parameters simultaneously. This way, we could make sure that we have reached maximum range under our constraints.

This section of the report walks through each individual variable parameter and its effect on the effectiveness of the system.

These variable parameters include:

- Sling length
- Release angle
- Counterweight hang length
- Throwing arm length
- Mass ratio

Sling Length

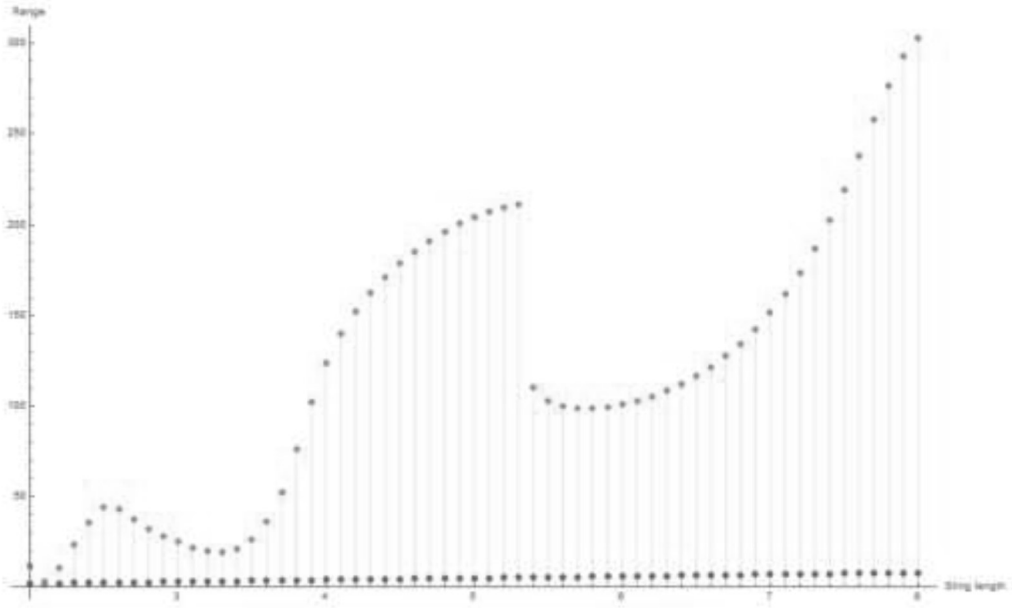


Figure 9 *Range as a function of sling length*

We understand that range of projectile depends both on the release angle and release velocity; hence, we do not have a continuously growing range, but rather a sinusoidal graph with varying amplitude.

The peaks represent the values of sling length when the projectile angle nears 45 degrees at release point.

Release Angle

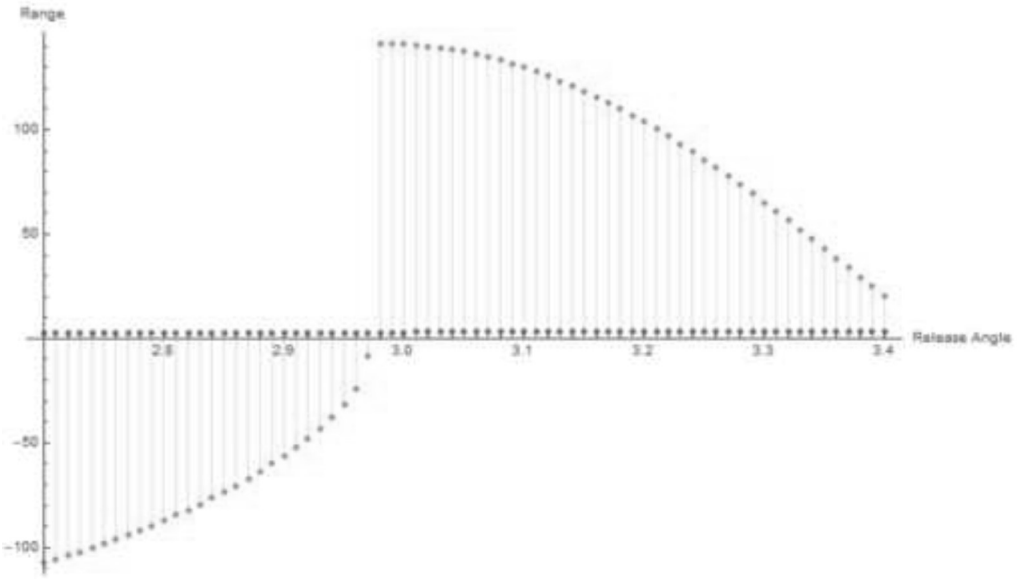


Figure 10 *Range as a function of release angle*

By manipulating the hook, we can vary the release angle of the trebuchet. Therefore, to optimize the range of projectile, we must understand the impact of release angle on range. As reflected in Figure 10, a lower release angle might release the projectile too early and result in a negative range. But as we pass through the 2.9 radian mark, the beam gets to swing further and we get the positive range. We can see that the maximum range can be obtained at release angle 3.0 radians given that all other parameters are constant.

Counterweight Hang Length

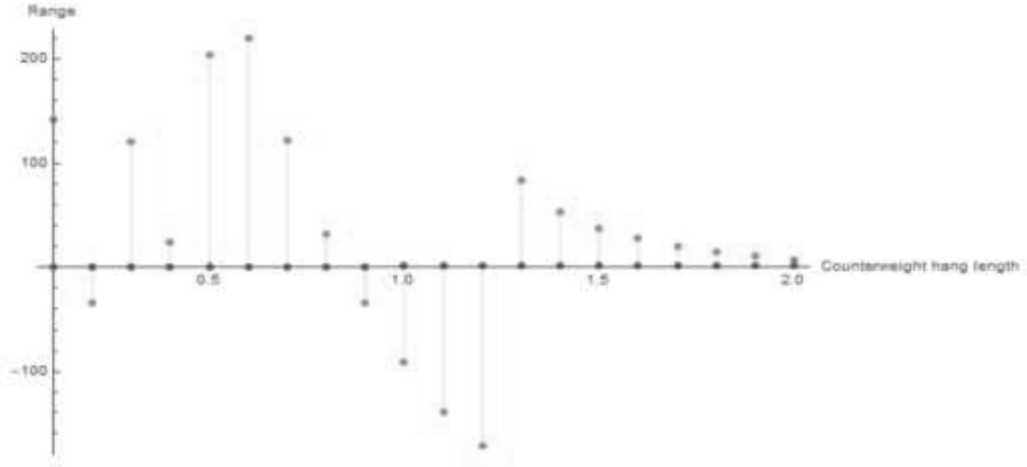


Figure 11 *Range as a function of counterweight hang length*

The hinged counter weight allows the counterweight to fall in a straight line, thus consuming less energy and transferring the maximum amount of energy to the projectile. At the same time, the motion of the counterweight needs to be in sync with the projectile. The projectile should reach the release point when the counterweight is at minimum velocity. This allows most of the kinetic energy of the system to be content in the projectile. Therefore, the length of the counterweight affects the range significantly. We can pick a length of 0.6 meters from the above graph for optimum range.

Throwing Arm Length

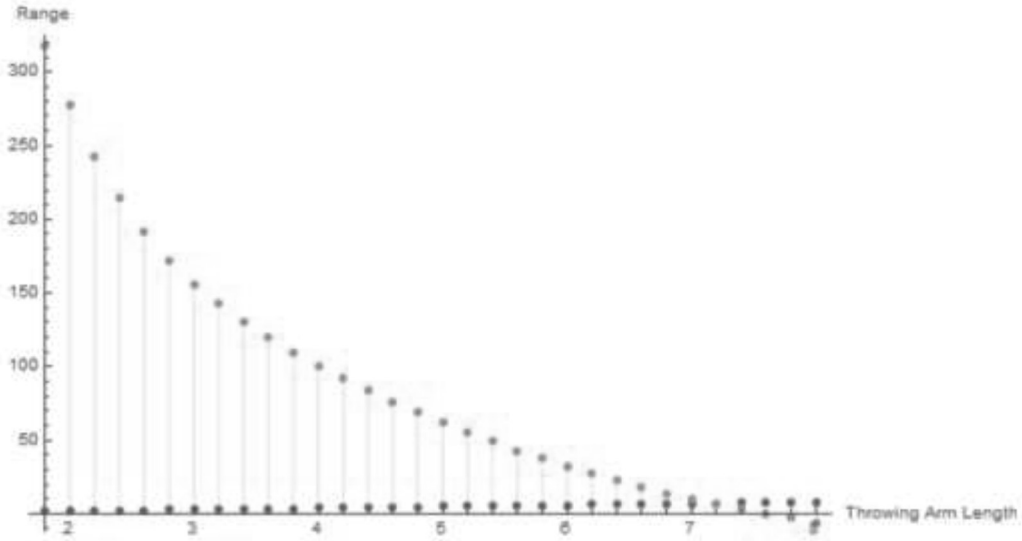


Figure 12 *Range as a function of throwing arm length*

From Figure 12, we can see that the shorter throwing arm is good for the initial parameters that we have set. But the plot is not conclusive as varying other parameters might affect the range and the optimum length of the throwing arm might be different for those parameters.

Mass Ratio

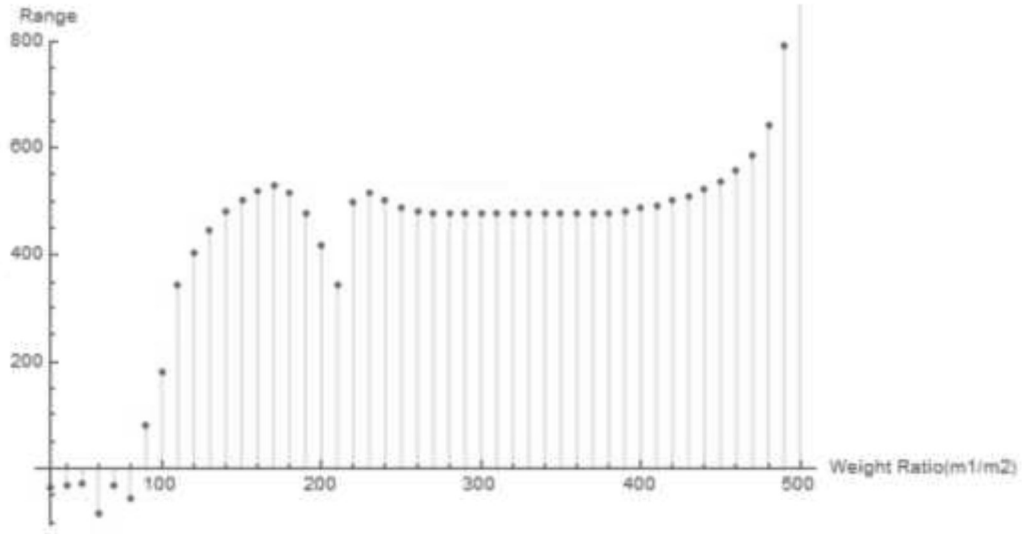
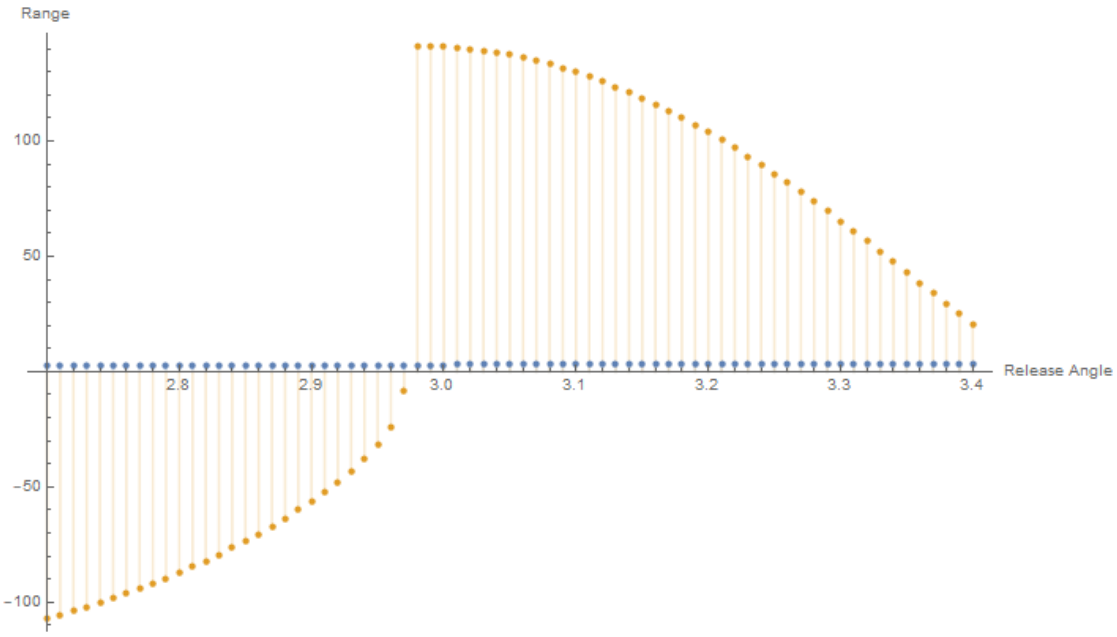


Figure 13 *Range as a function of mass ratio*

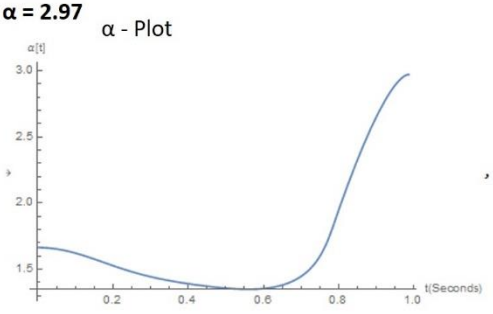
The mass ratio iteration clearly shows the optimum mass ratio is 120 to 160. Increasing the mass further would not result in significant benefits.

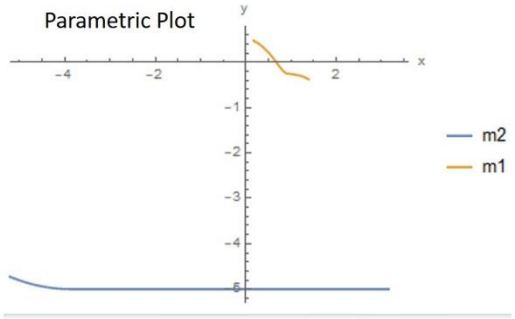
Analyzing Sudden rise in Plot between 2.97 and 2.98 Release angle



If we run the simulation it is clear that at α = 2.97 the projectile is released too early causing the negative range while at α = 2.98 the beam gets more rotation and the release is more towards the 45 Degree from ground as can be seen from below plots

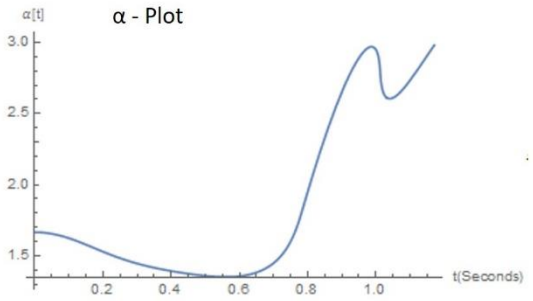
Release at α = 2.97

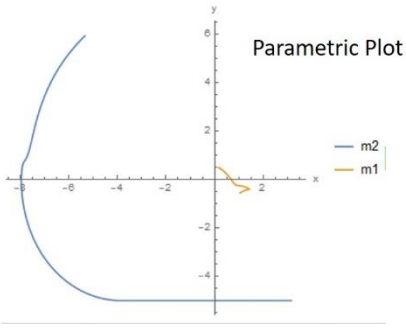




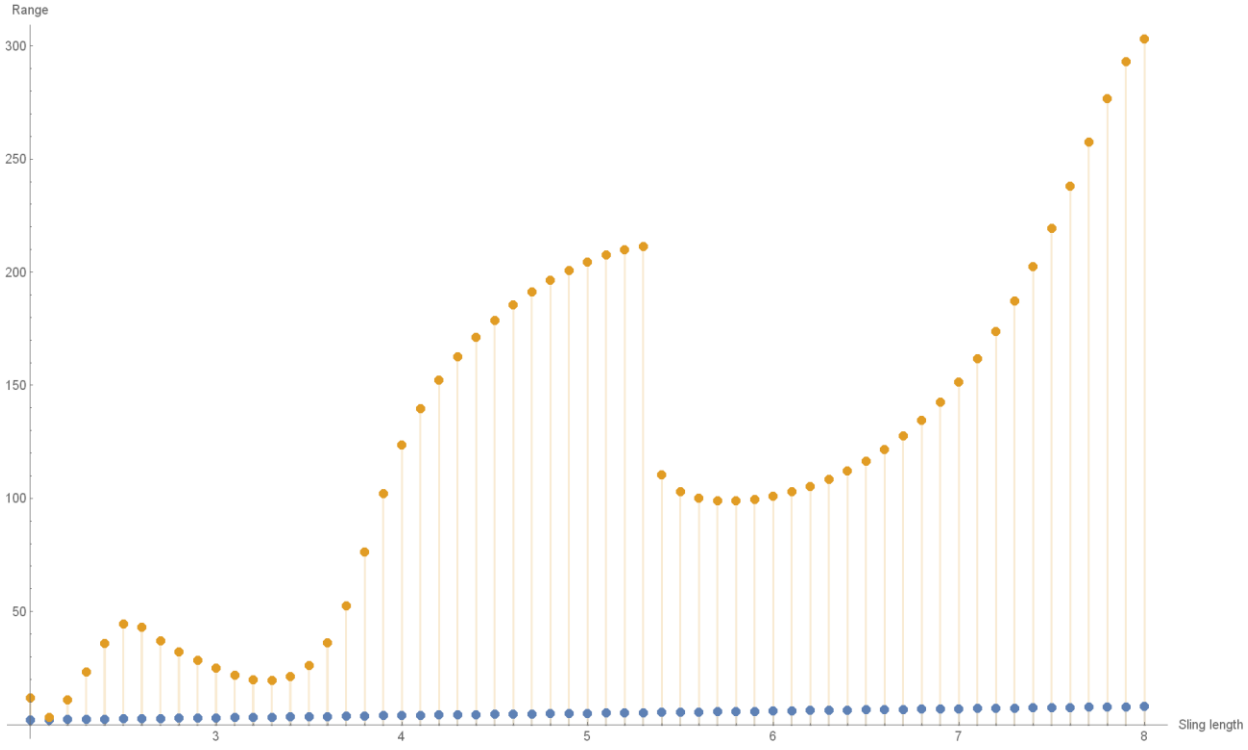
Notice that the Projectile is released at an obtuse angle causing a negative range, Also the projectile is released to early, While at α = 2.98, the system gets sufficient time where α drops back and theta gains value after which means when reach α = 2.98 the projectile angle is closer to 45 Degree from ground

Release at α = 2.98

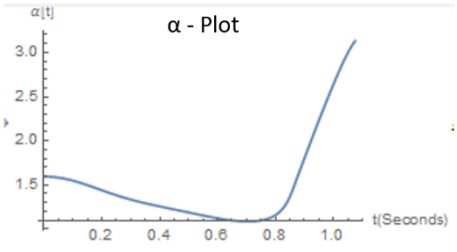


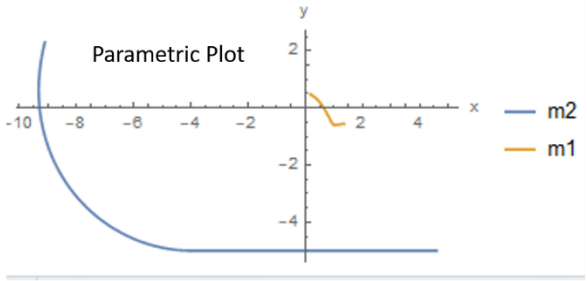


Analysis of sudden fall in range between sling length 5.3 to 5.4



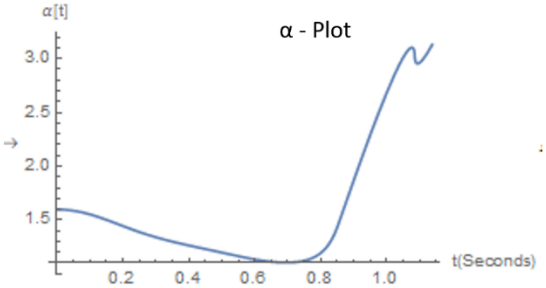
Sling Length =5.4

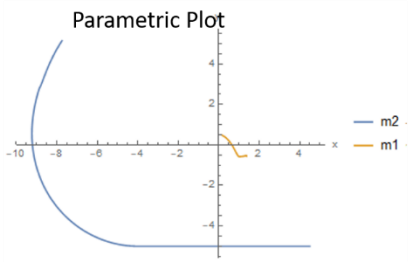




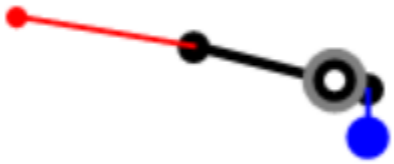
Similar to the effect we observed in Release angle(α) there exists a threshold value above which the α begins to fall and θ rises which gives system more time and at this new θ when the α value reaches to the release point projectile angle is closer to 45 Degree from ground angle. We can clearly se the drop in Alpha value in the graph for sling length 5.3 where as in sling length 5.4 the projectile is released quite early

Sling Length =5.3





Animation



Multi-Variable Optimization

Following our single variable optimization study, it became clear that all parameters are inter-linked. As a result, we cannot fix one parameter to fully studying that one parameter. We need to study the relation between these parameters and then decide on the optimum value.

First, we studied the varying two variables simultaneously. For adequate result visualization, we used a three-dimensional discrete plot shown in Figure 14.

Release Trigger Angle and Sling Length

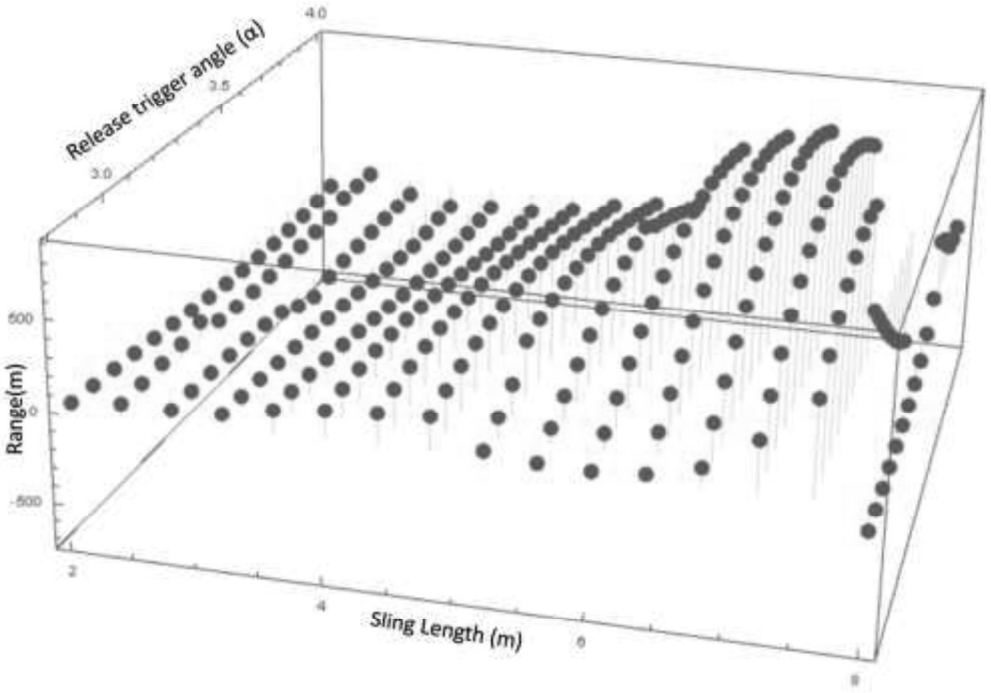


Figure 14 *Range as a function of release trigger angle and sling length*

This 3D plot allows us to visualize the impact of two variables on the range. According to Figure 14, the ideal release trigger angle would be between 3.1 and 3.5 with a sling length of about 6 to 7.5 meters given that all other parameters are at the picked initial condition.

This visualization begged us to look for a better visualization tool where we could study the impact of even more variables simultaneously. We determined two different approaches would be best suited: the three dimensional contour plot and the density plot.

Figure 15 shows some initial plots we used to get an understanding of how each plot displays the data and to choose which would be the ideal tool for understanding and optimizing the trebuchet.

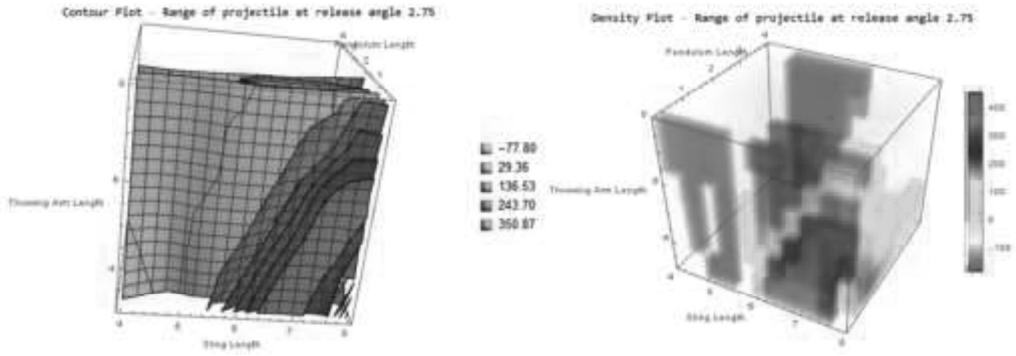
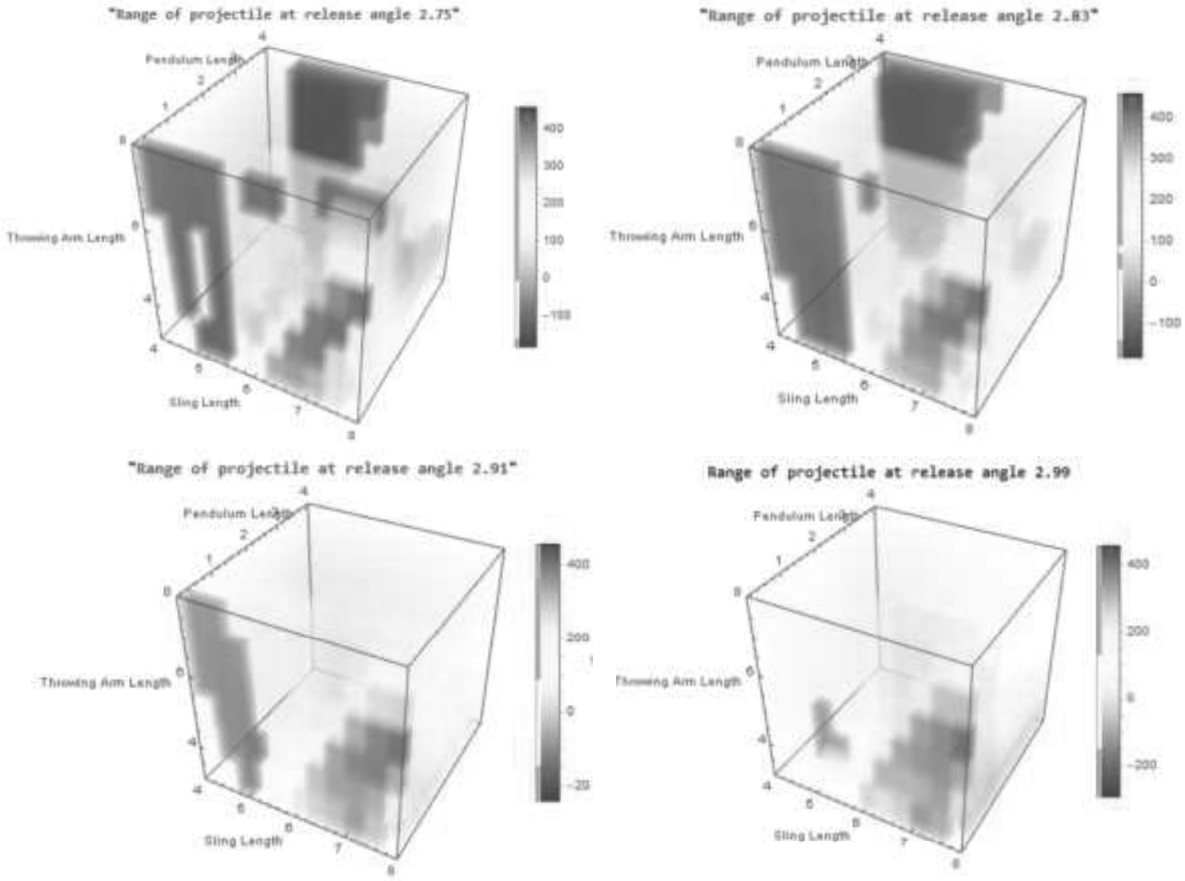


Figure 15 *Initial trial for the contour plot and density plot*

Following a study into both options, we decided to move on with the density plot for further purposes of optimization due to ease of data visualization.

Simultaneous Variation of Four Parameters



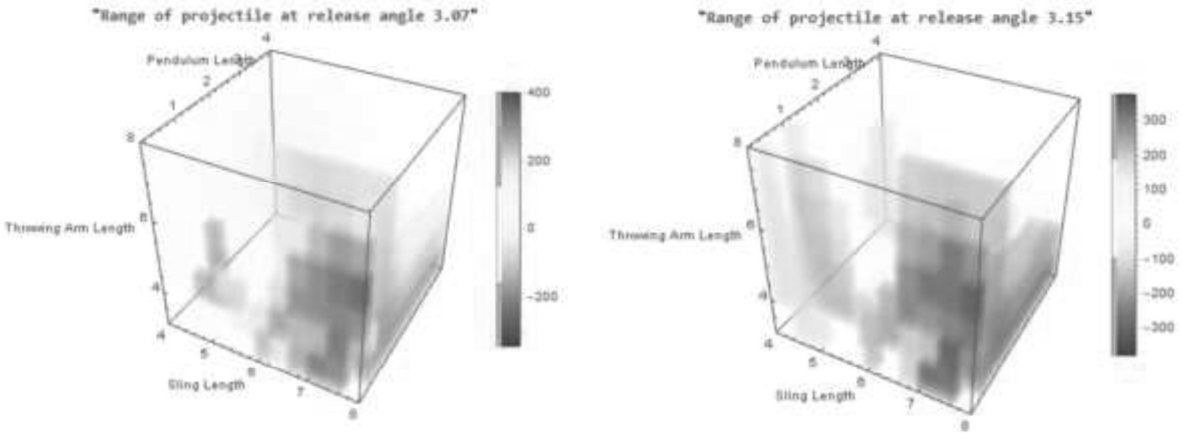


Figure 16 *Set of density plots showing interrelation between four parameters*

The set of 3D graphs in Figure 16 allows us to study the interrelation of each parameter with all other parameters. These density plots were generated using the following four-dimensional domain:

- Throwing arm ratio: 1:4 to 1:8
- Counterweight hang length: 0.1 to 4.1 meters
- Sling length: 3 to 8 meters
- Release angle: 2.75 to 3.15

Although a lot of information and meaning can be taken from these density plots. However, due to the scope of optimization for this study, we simply pick the optimum parameters from the plots.

Optimized Parameters

According to the results shown in Figure 16, the optimized parameters are as follows:

Sling Length: 5 meters
Release Angle: 2.83
Counterweight Hang Length: 0.1 meters
Throwing Arm Ratio: 1:6.4
Mass Ratio: 1:133

Final Simulation

Figure 17, Figure 18, and Figure 19 show the complete results of the final simulation utilizing these optimized parameters.

Projectile Angle: 45.0032 degrees
Projectile Velocity: 67.09 meters per second
Range: 458 meters

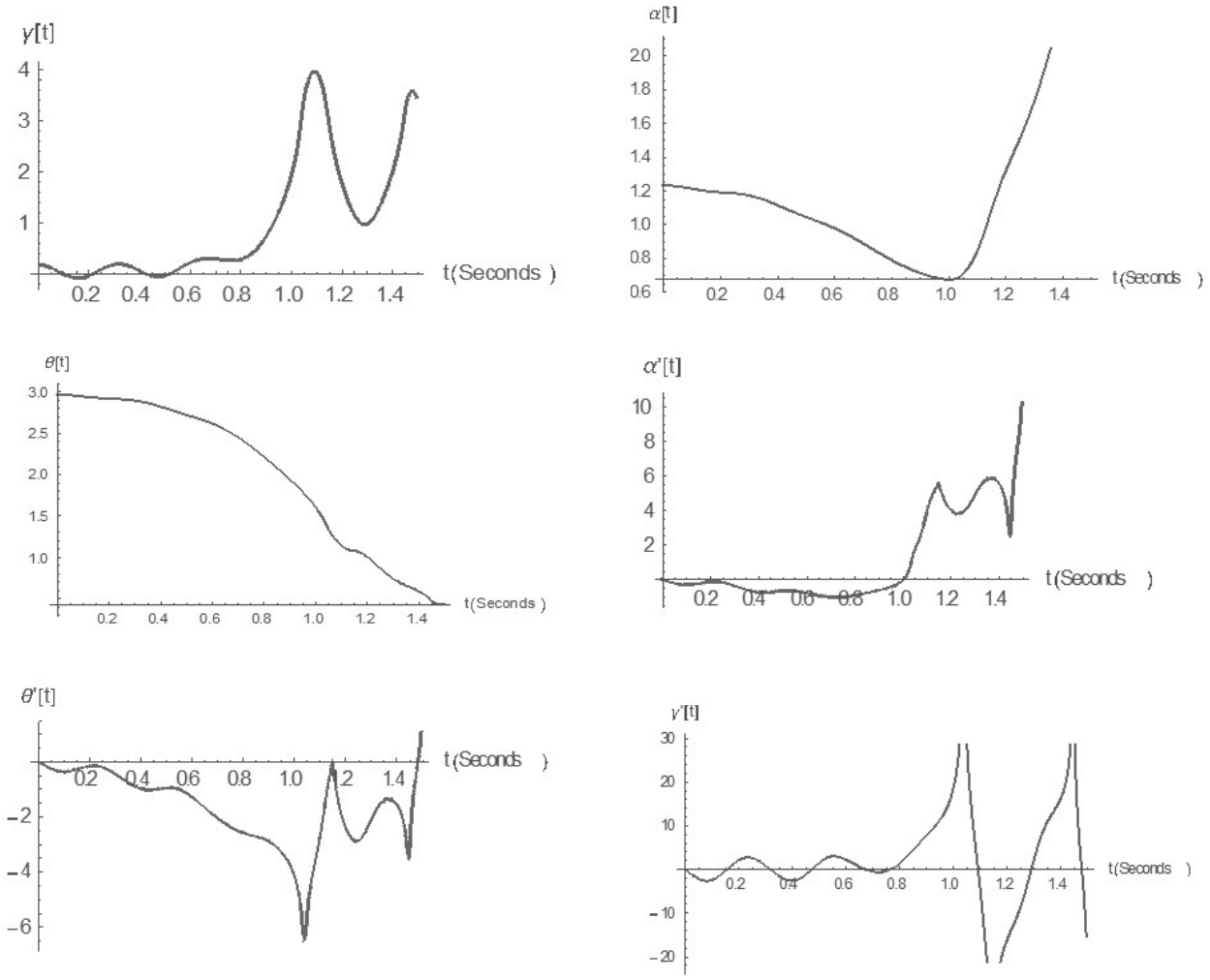


Figure 17 *Results of the final simulation using optimized parameters*

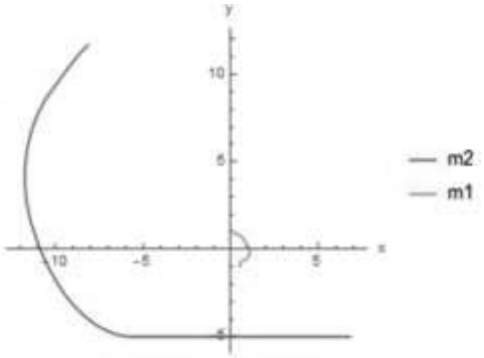


Figure 18 *Parametric plot of both masses until point of release*

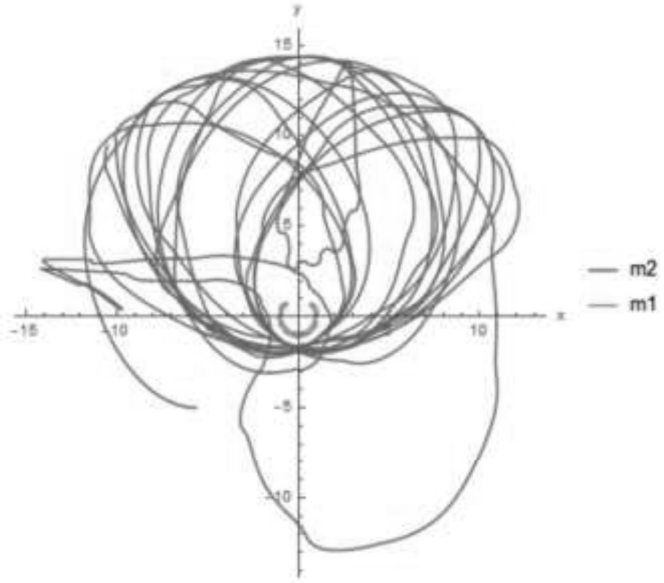


Figure 19 *Complete parametric plot through point of release*

Conclusion

In addition to the rich history associated with the trebuchet, all of the calculations and analysis involved the modeling, simulation, and optimization of the design have proven to be equally rich in challenge and fascination. The Lagrange method utilized in this study was a very straightforward method to obtain the equations of motion. The ground constraint proved to be challenging, but this was simply handled by creating an initial condition problem to derive the complete solution. In order to set up the simulations for optimization, we needed to interpret the data by defining quantities that would reflect system performance. These quantities were range, release angle, and release velocity.

During the optimization process, it proved inadequate to only optimize one variable at a time due the interconnection between the variables. We found the best way to optimize all parameters is to use a density plot so we could incorporate four of the variables. Ultimately, this proved to be very successful, as we were able to significantly optimize the system from our initial simulation to our final simulation.

References

[1] Jahed, H. (2006). "Trebuchet Design." ME380 Project Manual.

[2] Constans, E. and A. Constans (2015). "Treb-Bot: development and use of a Trebuchet simulator." Physics Teacher 53(6): 347-348.

[3] Chevedden, P. E., et al. (1995). "Trebuchet." Scientific American 273(1): 66-66.

[4] Gati, B. (2016). Mobile launching trebuchet for UAVs. 30th Congress of the International Council of the Aeronautical Sciences, ICAS 2016, September 25, 2016 - September 30, 2016, Daejeon, Korea, Republic of, International Council of the Aeronautical Sciences.

[5] Rhoten, R. P. (2006). Trebuchet energy efficiency - Experimental results. 44th AIAA Aerospace Sciences Meeting 2006, January 9, 2006 - January 12, 2006, Reno, NV, United states, American Institute of Aeronautics and Astronautics Inc.

[6] English, Trevor. (2017). Designing a Trebuchet – Optimizing Weight and Length, January 16, 2017.

Appendix

```
Needs["DifferentialEquations`NDSolveProblems`"];
Needs["DifferentialEquations`NDSolveUtilities`"];
Needs["DifferentialEquations`InterpolatingFunctionAnatomy`"];
Needs["GUIKit`"];
(*RangeOfProjectile[Throwing Arm length a,weight arm length b,
  Weigth pendulum length c,height of pivot d,length of sling l,mass 1 m1,
  mass to be thrown m2,release angle trigger rA,show log  1 or 0]
 This function takes the variables of the trebuchet and
 provides all the significant values in the output.
  this implementstion of the process as a function Simpliflies
 iteration process and makes the optimization faster
 slog values allows you to print the values of variables in current
 iteration the logging if switched off by default and can be
 enabled by passing value 1*)
iniTheta = 170 Degree;
initialAlfa = 10 Degree;
RangeOfProjectile[a_, b_, c_, d_, l_, m1_, m2_, rA_, Slog_: 0] := (designDelay = 5;
  ϕ[t_] = θ[t] - ArcCos[(d + a * Cos[θ[t]]) / l];
  x1[t_] = b Sin[θ[t]] - c Sin[θ[t] + γ[t]];
y1[t_] = -b Cos[θ[t]] + c Cos[θ[t] + γ[t]];
x2[t_] = -a Sin[θ[t]] + l Sin[θ[t] - ϕ[t]];
y2[t_] = a Cos[θ[t]] - l Cos[θ[t] - ϕ[t]];
  Te1 =
   1/2 m1 ((D[x1[t], t]^2) + (D[y1[t], t]^2)) + 1/2 m2 ((D[x2[t], t]^2) + (D[y2[t], t]^2));
  V1 = m1 g y1[t] + m2 g y2[t];

  L1 = Te1 - V1;


  Eqn1 = {D[D[L1, θ'[t]], t] - D[L1, θ[t]] == 0,
    D[D[L1, γ'[t]], t] - D[L1, γ[t]] == 0, θ[0] == iniTheta, θ'[0] == 0, γ[0] == initialAlfa,
    γ'[0] == 0, WhenEvent[θ[t] == ArcCos[(l - d)/a] + 0.00001, "StopIntegration"]};


  sol1 = NDSolve[Eqn1, {γ, θ}, {t, 0, 2}];
  end = Last[First[InterpolatingFunctionDomain[First[θ /. sol1]]]];
  datat = Table[Flatten[{t, Evaluate[{θ[t]} /. sol1]}], {t, 0, end, 0.0005}];
  datag = Table[Flatten[{t, Evaluate[{γ[t]} /. sol1]}], {t, 0, end, 0.0005}];
  dataa = Table[Flatten[{t, Evaluate[{ϕ[t]} /. sol1]}], {t, 0, end, 0.0005}];

  thE = First[θ[end] /. sol1];
  thvE = First[θ'[end] /. sol1];
  gmE = First[γ[end] /. sol1];
  gmvE = First[γ'[end] /. sol1];
  alfE = First[ϕ[end] /. sol1];
  alfEV = First[ϕ'[end] /. sol1];;
```

```
  x2[t_] := -a Sin[θ[t]] + l Sin[θ[t] - α[t]];
y2[t_] := a Cos[θ[t]] - l Cos[θ[t] - α[t]];

  Te =
   1/2 m1 ((D[x1[t], t]^2) + (D[y1[t], t]^2)) + 1/2 m2 ((D[x2[t], t]^2) + (D[y2[t], t]^2));
  V = m1 g y1[t] + m2 g y2[t];
  L = Te - V;
  Eqn2 = {D[D[L, α'[t]], t] - D[L, α[t]] == 0,
    D[D[L, θ'[t]], t] - D[L, θ[t]] == 0, D[D[L, γ'[t]], t] - D[L, γ[t]] == 0,
    θ[end] == thE, θ'[end] == thvE, γ[end] == gmE, γ'[end] == gmvE, α[end] == alfE,
    α'[end] == alfEV, WhenEvent[α[t] == rA, "StopIntegration"]};
  sol2 = NDSolve[Eqn2, {γ, θ, α}, {t, end, 20}];
  end2 = Last[First[InterpolatingFunctionDomain[First[θ /. sol2]]]];

  datat2 = Table[Flatten[{t, Evaluate[{θ[t]} /. sol2]}], {t, end, end2, 0.0005}];

  datag2 = Table[Flatten[{t, Evaluate[{γ[t]} /. sol2]}], {t, end, end2, 0.0005}];

  dataa2 = Table[Flatten[{t, Evaluate[{α[t]} /. sol2]}], {t, end, end2, 0.0005}];


  SOL = {{θ -> Interpolation[Join[datat, datat2]],
     γ -> Interpolation[Join[datag, datag2]], α → Interpolation[Join[dataa, dataa2]]}};
  theta = First[θ[end2 - 0.001] /. SOL];
  thetaDot = First[θ'[end2 - 0.001] /. SOL];
  alfa = First[α[end2 - 0.001] /. SOL];
  alfaDot = First[α'[end2 - 0.001] /. SOL];

  RelseaseAngle =
   ArcTan[(First[y2'[end2 - 0.001] /. SOL]) / (First[x2'[end2 - 0.001] /. SOL])];

  ReleaseVelocity = ((-l Cos[alfa - theta] (alfaDot - thetaDot) - a Cos[theta] thetaDot)^2 +
      (l Sin[alfa - theta] (alfaDot - thetaDot) - a Sin[theta] thetaDot)^2)^(1/2);
  If[Slog > 0, Print[{a, b, c, d, l, m1, m2, rA}], klp = a(*do nothing*)];
  {solution → SOL, range → (ReleaseVelocity^2 * Sin[2 * RelseaseAngle]) / g,
   releaseangle → RelseaseAngle / Degree, relVel → ReleaseVelocity, Plot_α → Plot[
     α[t] /. SOL, {t, 0, end2}, AxesLabel → {"t(Seconds)", "α[t]"}, PlotRange → Full],
   Plot_θ → Plot[θ[t] /. SOL, {t, 0, end2}, AxesLabel → {"t(Seconds)", "θ[t]"},
     PlotRange → Full],
   Plot_γ → Plot[γ[t] /. SOL, {t, 0, end2}, AxesLabel → {"t(Seconds)", "γ[t]"},
     PlotRange → Full],
   Plot_dα → Plot[α'[t] /. SOL, {t, 0, end2},
     AxesLabel → {"t(Seconds)", "α'[t]"}, PlotRange → Full],
   Plot_dθ → Plot[θ'[t] /. SOL, {t, 0, end2},
     AxesLabel → {"t(Seconds)", "θ'[t]"}, PlotRange → Full],
   Plot_dγ → Plot[γ'[t] /. SOL, {t, 0, end2},
     AxesLabel → {"t(Seconds)", "γ'[t]"}, PlotRange → Full],
   Para_P → ParametricPlot[{{First[x2[t] /. SOL], First[y2[t] /. SOL]},
      {First[x1[t] /. SOL], First[y1[t] /. SOL]}},
```

```
    {t, 0, end2}, AxesLabel → {"x", "y"}, PlotLegends → {"m2", "m1"}]}

)

g = 9.81;
a = 4;
b = 1;
c = 0.5;
d = 5;
l = 4;
m1 = 133;
m2 = 1;

rA = 2.75;
```

Initial Solution

```
Solution = RangeOfProjectile[4, b, c, d, 4, m1, m2, 3.14]
```

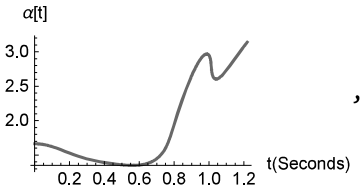
$\{$solution $\to \{\{\theta \to$ InterpolatingFunction[⟨Domain: {{0., 1.22}}, Output: scalar⟩],

$\gamma \to$ InterpolatingFunction[⟨Domain: {{0., 1.22}}, Output: scalar⟩],

$\alpha \to$ InterpolatingFunction[⟨Domain: {{0., 1.22}}, Output: scalar⟩]$\}\}\}$,

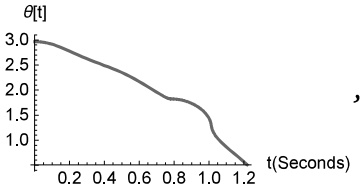
range → 121.193, releaseangle → 29.579, relVel → 37.2118,

Plot_$\alpha \to$



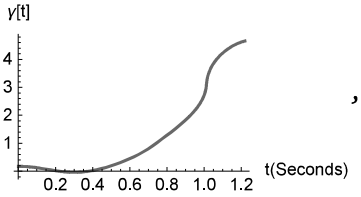
,

Plot_$\theta \to$



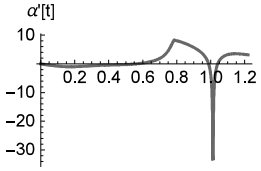
,

Plot_γ →



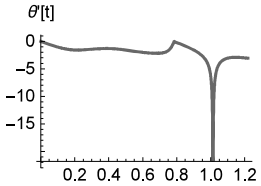
,

Plot_dα →

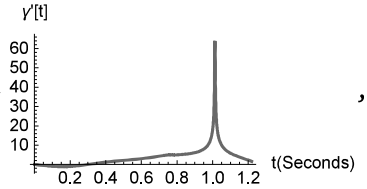


,

Plot_dθ →

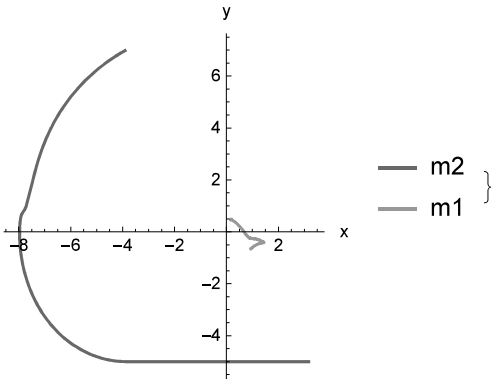


, Plot_dγ →



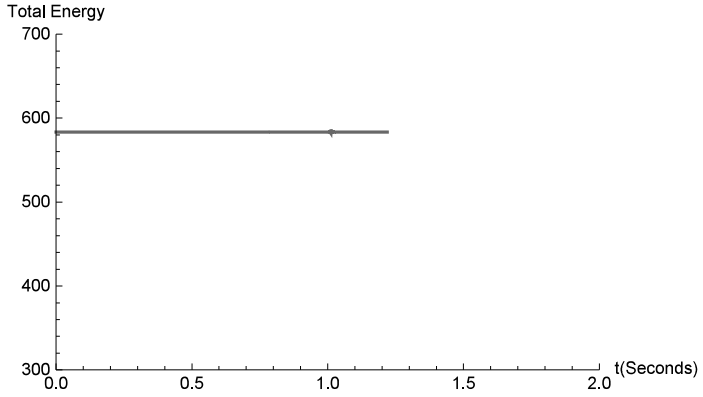
,

Para_P →

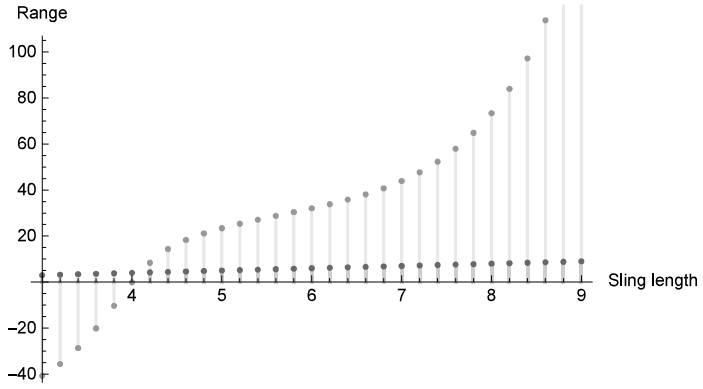


Validation

```
Plot[Te + V /. SOL, {t, 0, end2},
 AxesLabel → {"t(Seconds)", "Total Energy"} , PlotRange → {{0, 2}, {300, 700}}]
```



```
DiscretePlot[{x, range /. RangeOfProjectile[6, b, c, d, x, m1, m2, 3.13]},
 {x, 3, 9, 0.2}, AxesLabel → {"Sling length", "Range"}]
```



```
(*DO not run the next cell Use the text data from below,
it takes more that 2 Hours to run*)

data3dr2 = Table[range /. RangeOfProjectile[z, b, y, d, x, m1, m2, u, 1],
  {z, 4, 8, 0.8}, {y, 0.1, 4.1, 0.4}, {x, 3, 8, 0.5}, {u, 2.75, 3.15, 0.08}]


(*Calling function  RangeOfProjectile[Throwing Arm length a,weight arm length b,
  Weigth pendulum length c,height of pivot d,length of sling l,mass 1 m1,
  mass to be thrown m2,release angle trigger rA, Show log  1 or 0]*)
Solution = RangeOfProjectile[4, b, c, d, 4, m1, m2, 3]
```

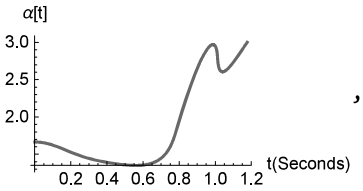
$\{$solution $\to \{\{\theta \to$ InterpolatingFunction[Domain: {{0., 1.18}} Output: scalar],

$\gamma \to$ InterpolatingFunction[Domain: {{0., 1.18}} Output: scalar],

$\alpha \to$ InterpolatingFunction[Domain: {{0., 1.18}} Output: scalar]$\}\}\}$,

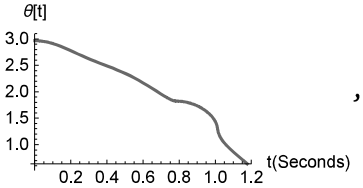
range $\to$ 140.957, releaseangle $\to$ 42.3589, relVel $\to$ 37.265,

Plot_$\alpha \to$



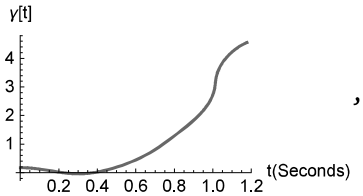
,

Plot_$\theta \to$



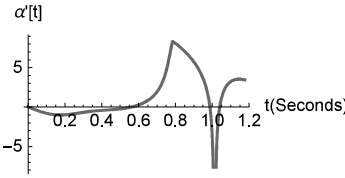
,

Plot_$\gamma \to$



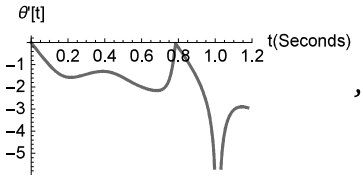
,

Plot_d$\alpha \to$

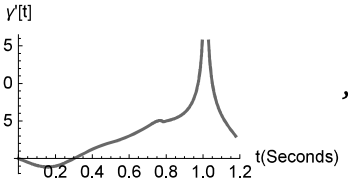


,

Plot_dΘ →

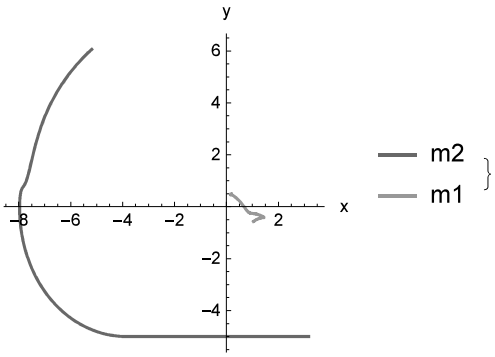


, Plot_dγ →



,

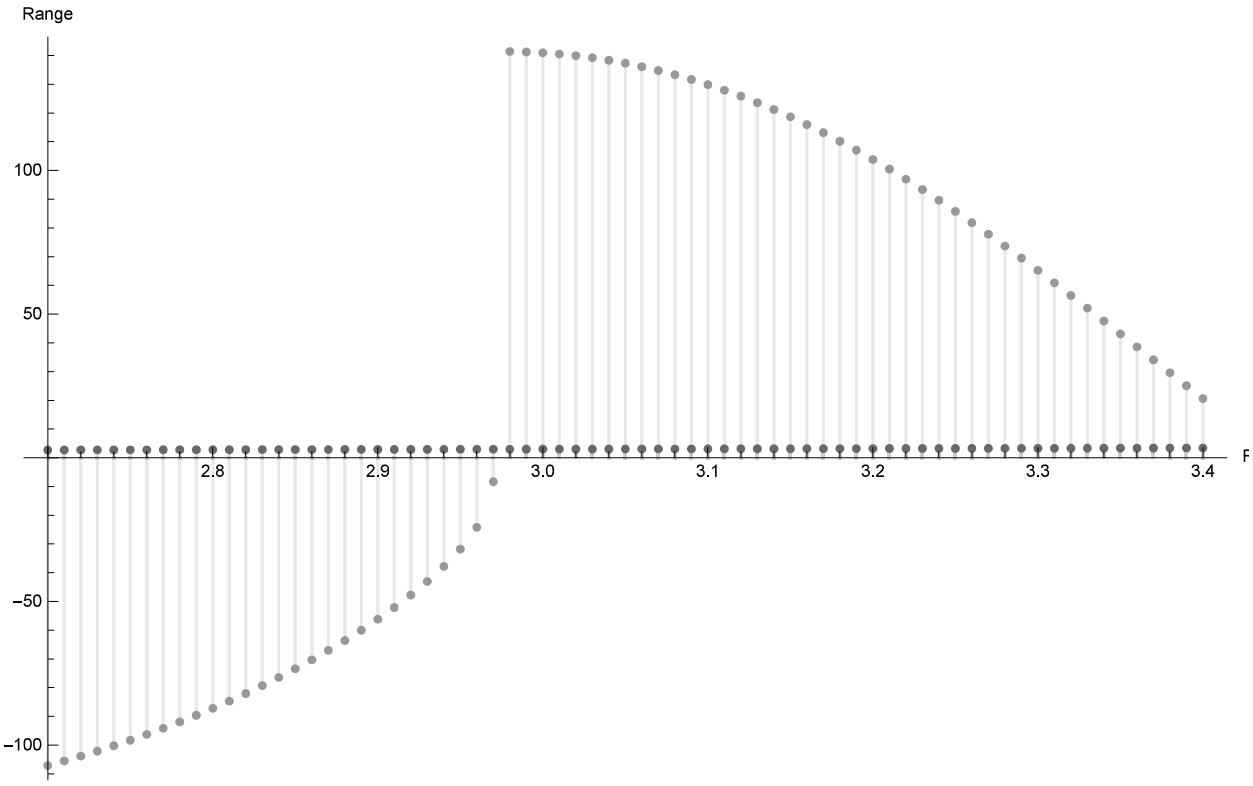
Para_P →




```
(*Calling function  RangeOfProjectile[Throwing Arm length a,weight arm length b,
  Weigth pendulum length c,height of pivot d,length of sling l,mass 1 m1,
  mass to be thrown m2,release angle trigger rA, Show log  1 or 0]*)
```

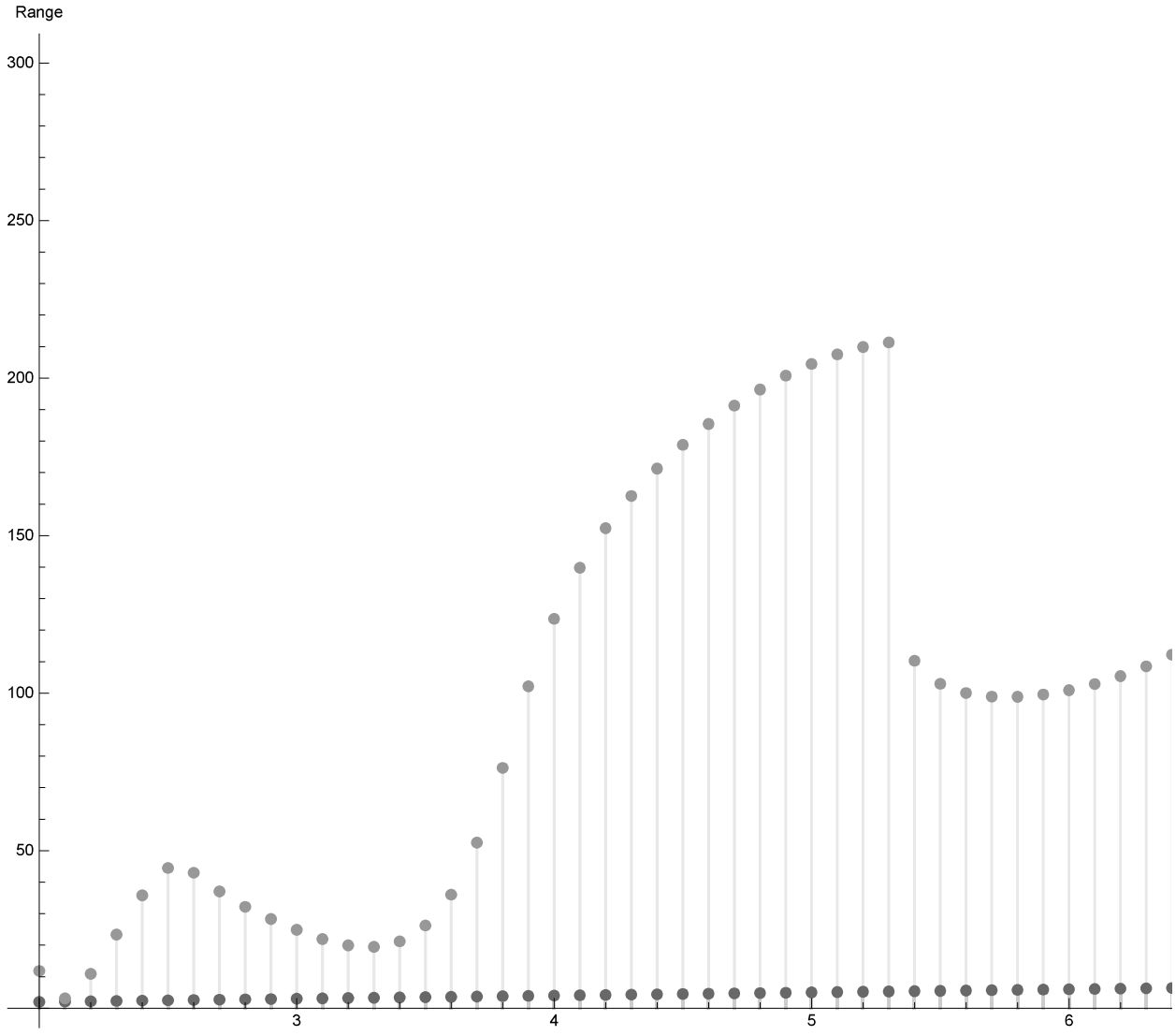
Release Angle iteration

```
DiscretePlot[{x, range /. RangeOfProjectile[4, b, c, d, 4, m1, m2, x, 1]},
 {x, 2.7, 3.4, 0.01}, AxesLabel → {"Release Angle", "Range"}]
```



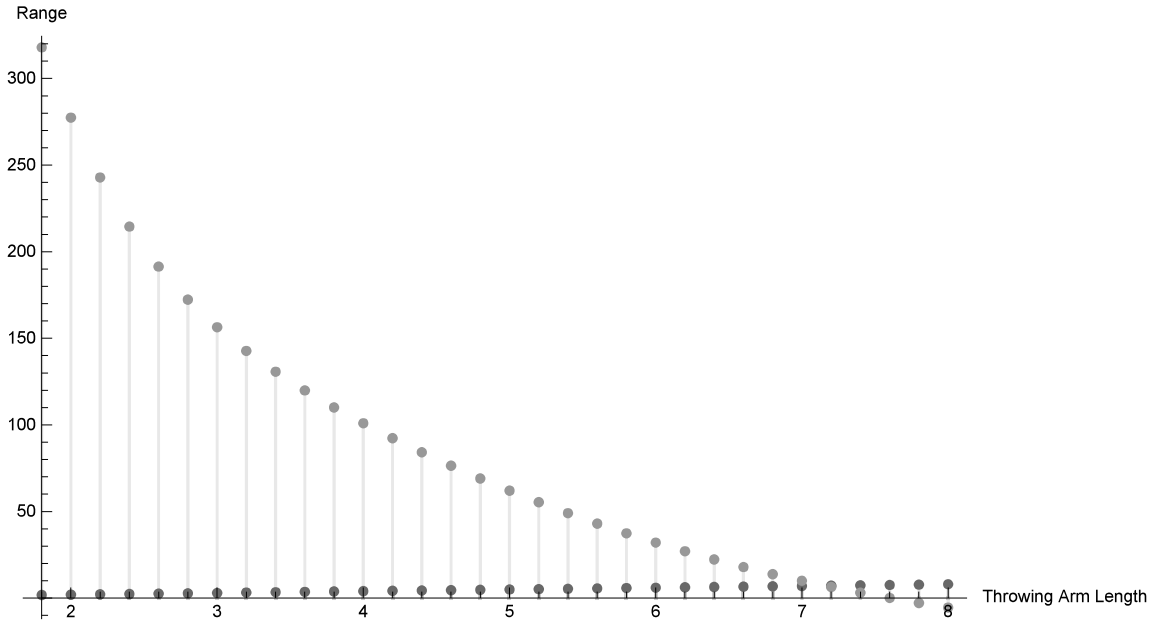
Sling length iteration

```
(*RangeOfProjectile[Throwing Arm length a,weight arm length b,
 Weigth pendulum length c,height of pivot d,length of sling l,
 mass 1 m1,mass to be thrown m2,release angle trigger rA]*)
DiscretePlot[{x, range /. RangeOfProjectile[4, b, c, d, x, m1, m2, 3.13]},
 {x, 2, 8, 0.1}, AxesLabel → {"Sling length", "Range"}]
```



Throwing arm length iteration

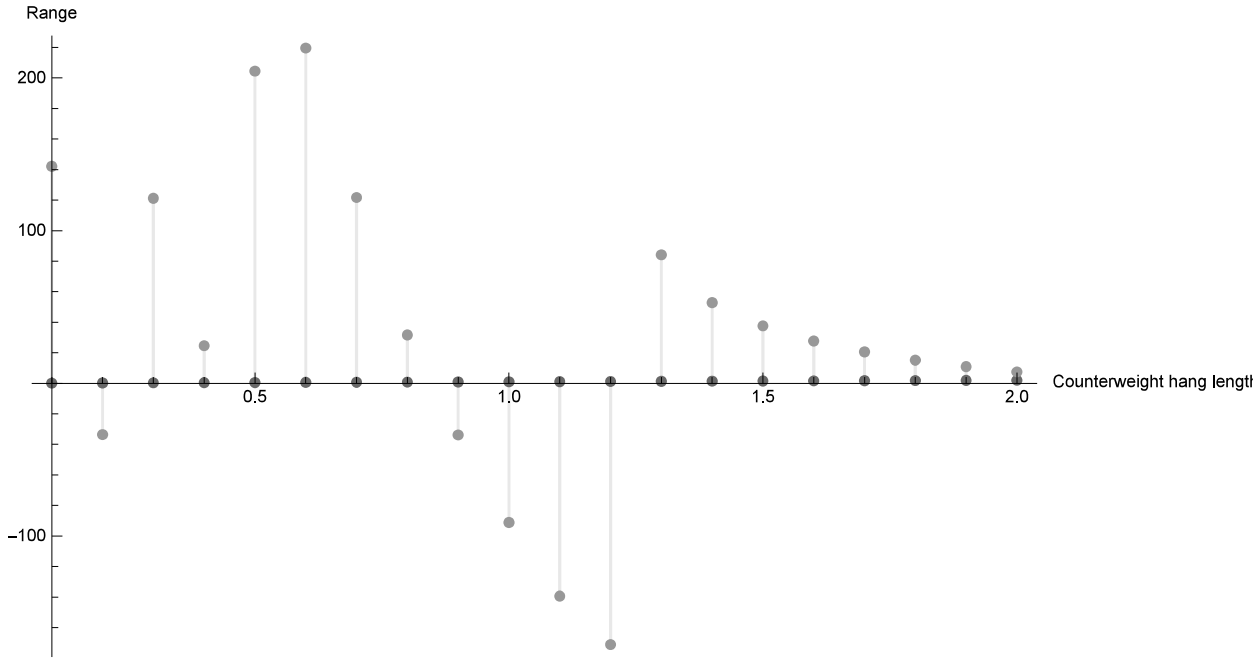
```
(*RangeOfProjectile[Throwing Arm length a,weight arm length b,
 Weigth pendulum length c,height of pivot d,length of sling l,
 mass 1 m1,mass to be thrown m2,release angle trigger rA]*)
DiscretePlot[{x, range /. RangeOfProjectile[x, b, c, d, 6, m1, m2, 3.13, 1]},
 {x, 1.8, 8, 0.2}, AxesLabel → {"Throwing Arm Length", "Range"}]
```



pendulum length iteration

```
(*RangeOfProjectile[Throwing Arm length a,weight arm length b,
 Weigth pendulum length c,height of pivot d,length of sling l,
 mass 1 m1,mass to be thrown m2,release angle trigger rA]*)
```

```
DiscretePlot[{x, range /. RangeOfProjectile[4, b, x, d, 5, m1, m2, 3.13]}, {x, 0.1, 2, 0.1},
 AxesLabel → {"Counterweight hang length", "Range"}, PlotRange → Full]
```



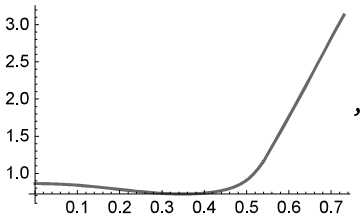
Test at choice of value from iteration

RangeOfProjectile[5, b, c, d, 6, m1, m2, 3.13]

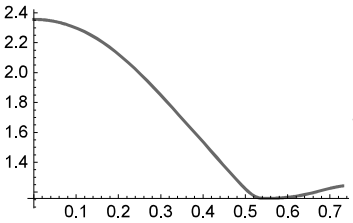
{solution → {{θ → InterpolatingFunction[],

γ → InterpolatingFunction[],

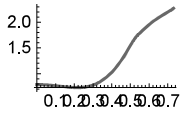
α → InterpolatingFunction[]}},

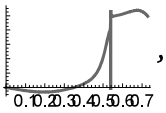
range → 183.453, Plot_α → 

, releaseangle → 72.3875,

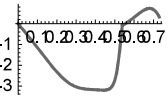
relVel → 55.8583, Plot_θ → 

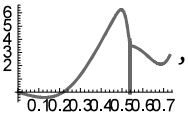
, Plot_γ →

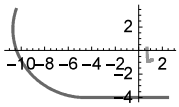


, Plot_dα → 

,

Plot_dθ → 

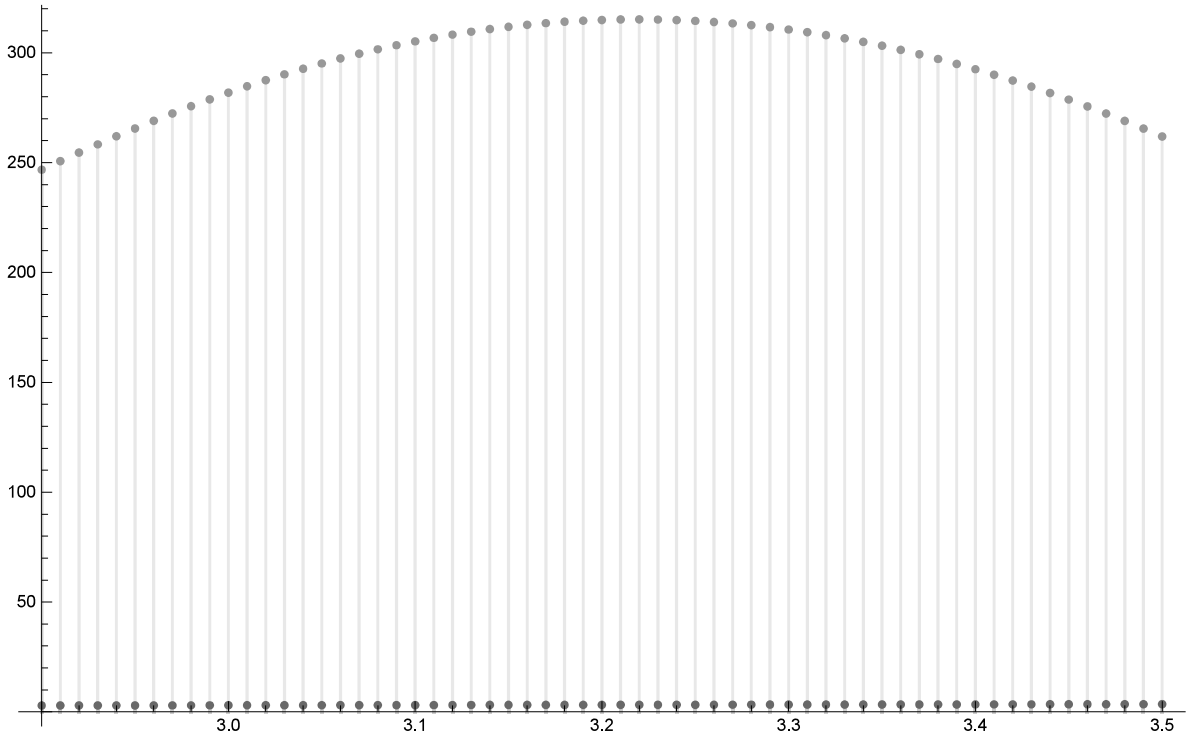
, Plot_dγ → 

, Para_P → 

}

release angle iteration again

```
DiscretePlot[{x, range /. RangeOfProjectile[4, b, 0.4, d, 5, m1, m2, x]},
 {x, 2.9, 3.5, 0.01}, AxesLabel → {"Release Angle", "Range"}]
```



Test at choice of value from iteration

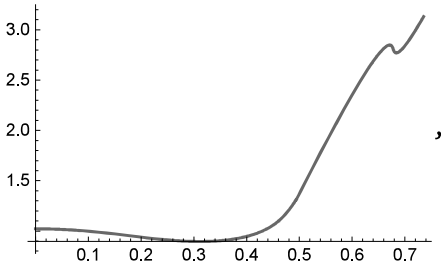
```
RangeOfProjectile[4, b, 0.4, d, 5, m1, m2, 3.13]
```

$\{$solution $\to \{\{\theta \to$ InterpolatingFunction[⊞ Domain: {{0., 0.735}} Output: scalar],

$\gamma \to$ InterpolatingFunction[⊞ Domain: {{0., 0.735}} Output: scalar],

$\alpha \to$ InterpolatingFunction[⊞ Domain: {{0., 0.735}} Output: scalar]$\}\}$,

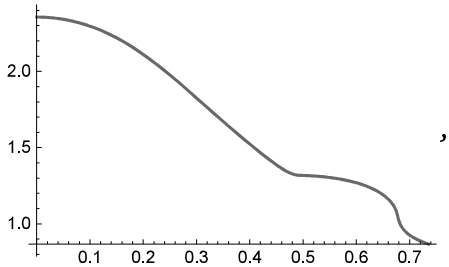
range → 309.6, Plot_$\alpha \to$



,

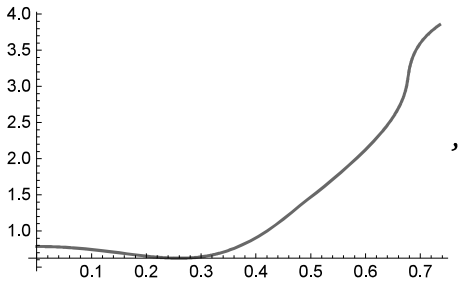
releaseangle → 50.8574, relVel → 55.6937,

Plot_θ →



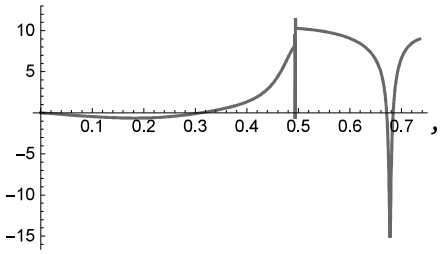
,

Plot_γ →



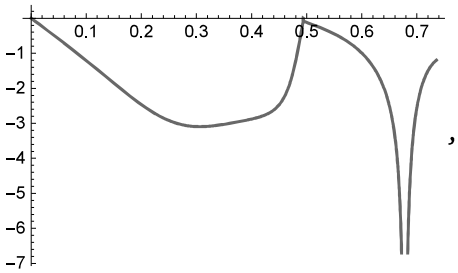
,

Plot_dα →



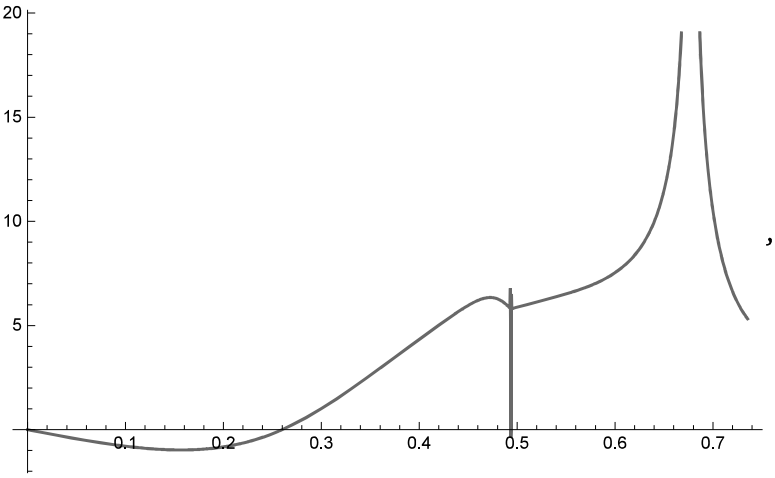
,

Plot_dθ →



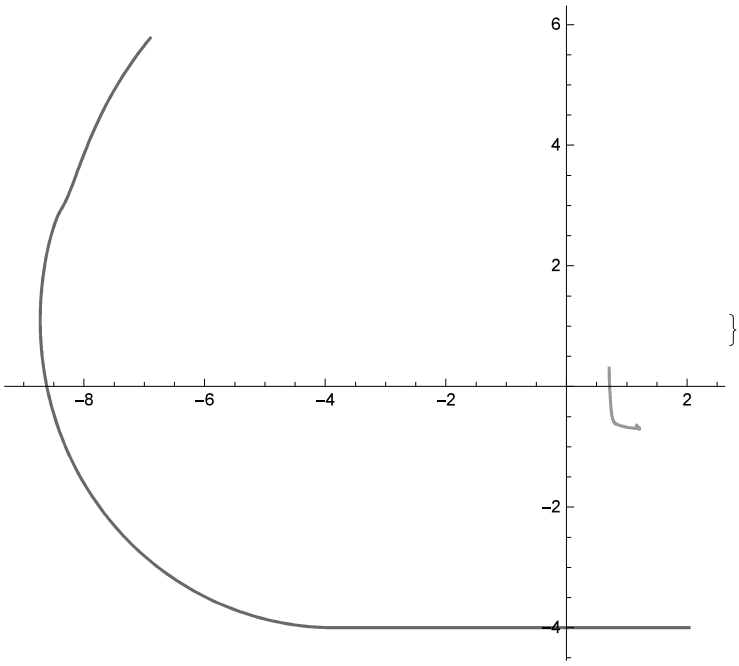
,

Plot_dγ →



,

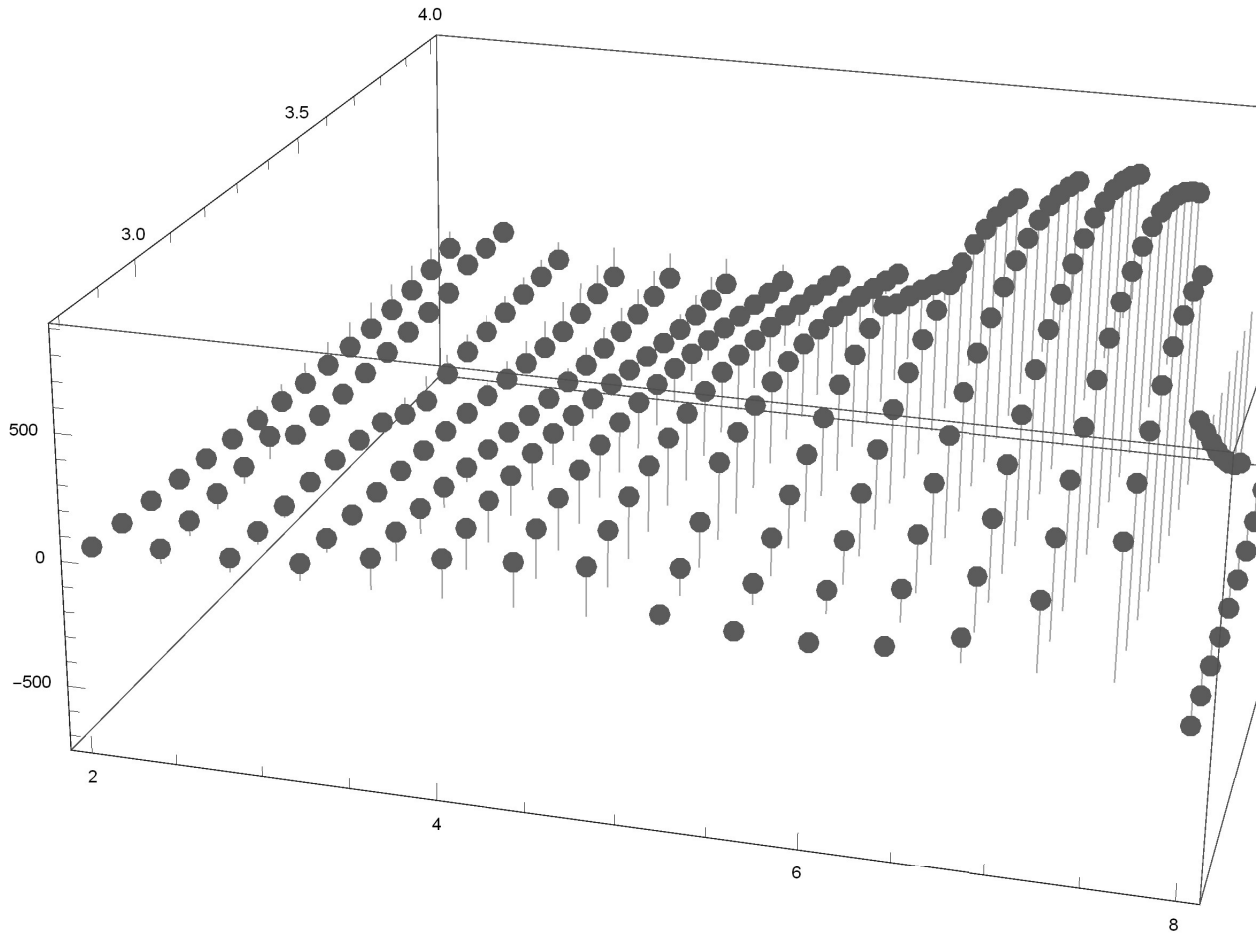
Para_P →



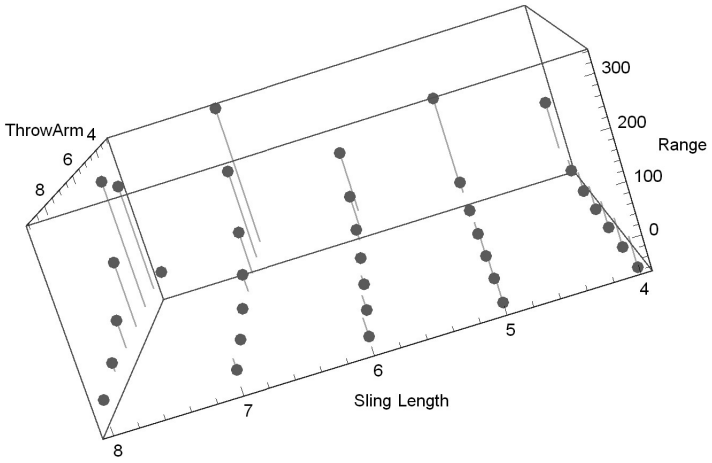
}

2D iteration across release angle and sling length

```
DiscretePlot3D[range /. RangeOfProjectile[4, b, 0.4, d, x, m1, m2, y, 1], {x, 2, 8, 0.4},
 {y, 2.8, 4, 0.08}, AxesLabel → {"Sling Length", "Release Angle", "Range"}]
```



```
DiscretePlot3D[range /. RangeOfProjectile[y, b, 0.4, d, x, m1, m2, 3.13, 1],
 {x, 4, 8, 1}, {y, 3, 9, 1}, AxesLabel → {"Sling Length", "ThrowArm", "Range"}]
```



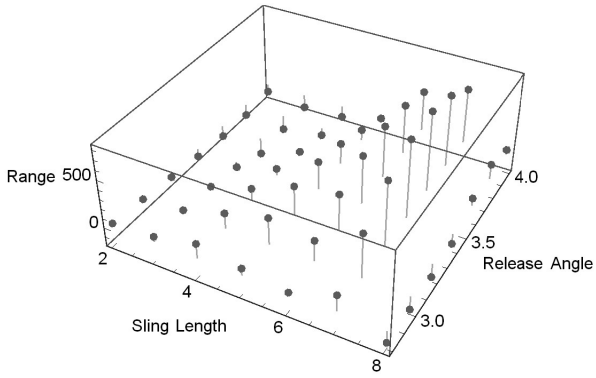

```
DiscretePlot3D[range /. RangeOfProjectile[4, b, 0.4, d, x, m1, m2, y], {x, 2, 8, 1},
 {y, 2.8, 4, 0.2}, AxesLabel → {"Sling Length", "Release Angle", "Range"}]
```

NDSolve: At t == 0.6499320637752753`, step size is effectively zero; singularity or stiff system suspected.

NDSolve: At t == 0.6499320637752753`, step size is effectively zero; singularity or stiff system suspected.

NDSolve: At t == 0.6499320637752753`, step size is effectively zero; singularity or stiff system suspected.

General: Further output of NDSolve::ndsz will be suppressed during this calculation.




```
data3d = Table[range /. RangeOfProjectile[z, b, 0.4, d, x, m1, m2, y, 1],
  {z, 2, 6, 0.5}, {y, 2.8, 3.3, 0.1}, {x, 2, 6, 0.5}]
```

Animation

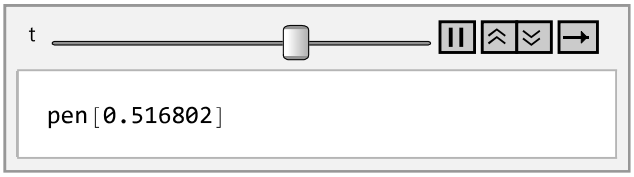
```
Clear[pen]
```

```
pen[s_] := Module[
  {g = g, a = a, b = b, c = c, d = d, l = l, m1 = m1, m2 = m2, tmax = 20, ptA, ptB, t , ptm1, ptm2},
  x1[t] = b Sin[θ[t]] - c Sin[θ[t] + γ[t]];
y1[t] = -b Cos[θ[t]] + c Cos[θ[t] + γ[t]];
  x2[t] := -a Sin[θ[t]] + l Sin[θ[t] - α[t]];
y2[t] := a Cos[θ[t]] - l Cos[θ[t] - α[t]];
  ptA = {First[-a Sin[θ[s]] /. SOL], First[a Cos[θ[s]] /. SOL]};
  ptB = {First[b Sin[θ[s]] /. SOL], First[-b Cos[θ[s]] /. SOL]};
  ptm1 = {First[x1[s] /. SOL], First[y1[s] /. SOL]};
  ptm2 = {First[x2[s] /. SOL], First[y2[s] /. SOL]};
  Show[Graphics[{Thickness[.01], Line[{{0, 0}, ptA}], PointSize[.03], Black,
     Point[ptA], Thickness[.01], Line[{{0, 0}, ptB}], PointSize[.03], Black,
     Point[ptB], Gray, PointSize[.06], Point[{0, 0}], Black, PointSize[.04],
     Point[{0, 0}], White, PointSize[.02], Point[{0, 0}], Thickness[.005], Blue,
     Line[{ptB, ptm1}], PointSize[.08], Blue, Point[ptm1], Thickness[.005],
     Red, Line[{ptA, ptm2}], PointSize[.02], Red, Point[ptm2]}], Axes → False,
   Ticks → None, AxesStyle → GrayLevel[.5], PlotRange → {{-7, 7}, {-7, 7}}]]
```

```
ptA = {-a Cos[θ[s] - 90 Degree], -a Sin[θ[s] - 90 Degree]} /. SOL[[1]]
```

$$\{-4 \sin[\text{InterpolatingFunction}[\langle\text{Domain: } \{\{0.,\ 0.775\}\} \text{ Output: scalar}\rangle][s]],\ 4 \cos[\text{InterpolatingFunction}[\langle\text{Domain: } \{\{0.,\ 0.775\}\} \text{ Output: scalar}\rangle][s]]\}$$

```
mov = Animate[pen[t], {t, 0, end2}, AnimationRate → 0.15, RefreshRate → 50]
```

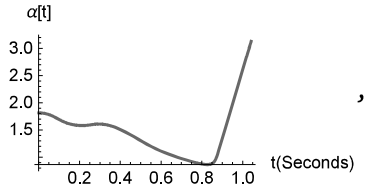



```
RangeOfProjectile[2, b, 0.4, 4, 5, m1, m2, 3.13]
```

$$\{\text{solution} \to \{\{\theta \to \text{InterpolatingFunction}[\langle\text{Domain: } \{\{0.,\ 1.04\}\} \text{ Output: scalar}\rangle],\ \gamma \to \text{InterpolatingFunction}[\langle\text{Domain: } \{\{0.,\ 1.04\}\} \text{ Output: scalar}\rangle],\ \alpha \to \text{InterpolatingFunction}[\langle\text{Domain: } \{\{0.,\ 1.04\}\} \text{ Output: scalar}\rangle]\}\},$$

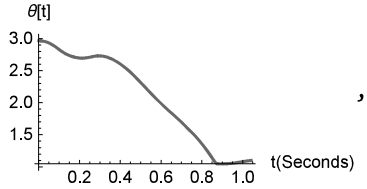
range → 284.205, releaseangle → 64.2954, relVel → 59.7245,

Plot_α →



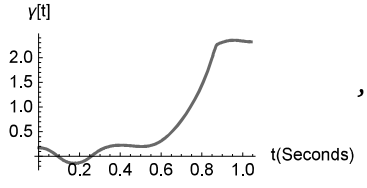
,

Plot_θ →



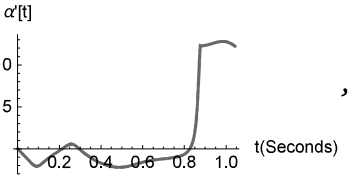
,

Plot_γ →



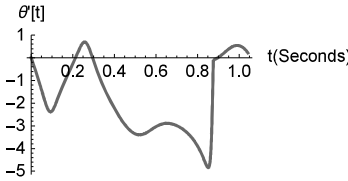
,

Plot_dα →

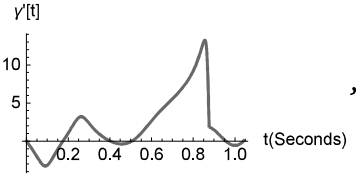


,

Plot_dθ →

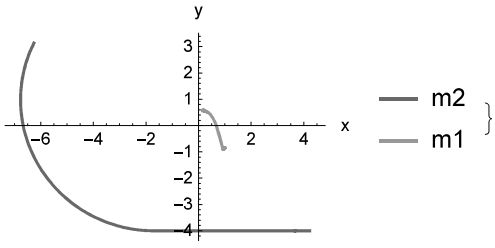


, Plot_dγ →



,

Para_P →



}

```
a = 2; c = 0.4; d = 4; l = 5;
```

```
data3dr2
```

```
RangeOfProjectile[5.5, b, 0.4, d, 8, m1, m2, 2.75,]
```

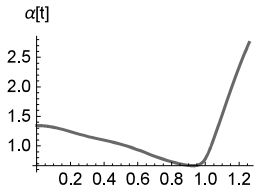
{solution → {{θ → InterpolatingFunction[],

$\gamma \to$ InterpolatingFunction[Domain: {{0., 1.26}} Output: scalar],

$\alpha \to$ InterpolatingFunction[Domain: {{0., 1.26}} Output: scalar]},

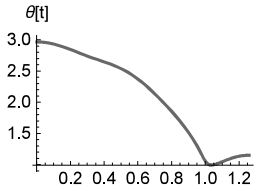
range → 14.9923, releaseangle → 88.4887, relVel → 52.8125,

Plot_$\alpha \to$



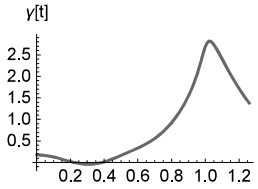
,

Plot_$\theta \to$



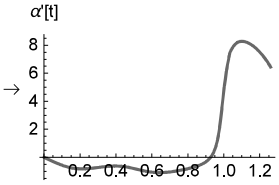
,

Plot_$\gamma \to$



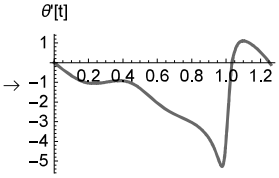
,

Plot_d$\alpha \to$

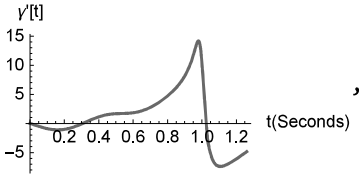


,

Plot_d$\theta \to$

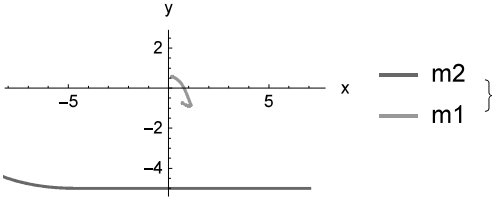


, Plot_d$\gamma \to$



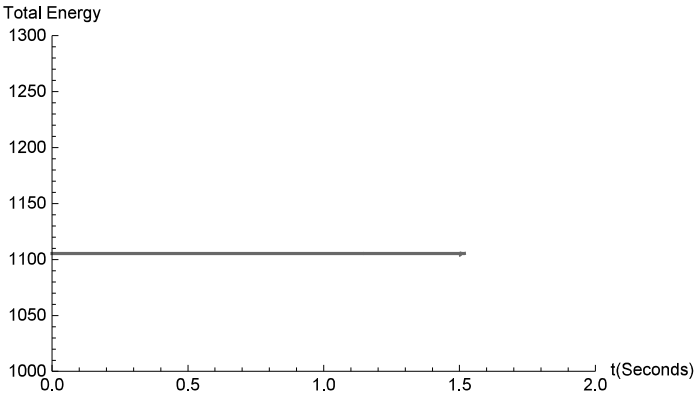
,

Para_P →



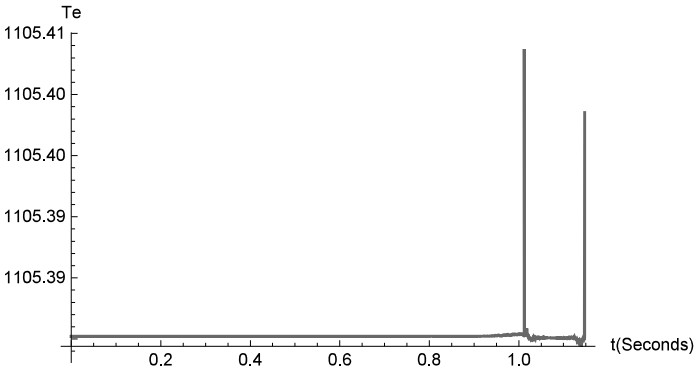
}

```
Plot[Te + V /. SOL, {t, 0, end2},
 AxesLabel → {"t(Seconds)", "Total Energy"}, PlotRange → {{0, 2}, {1000, 1300}}]
```

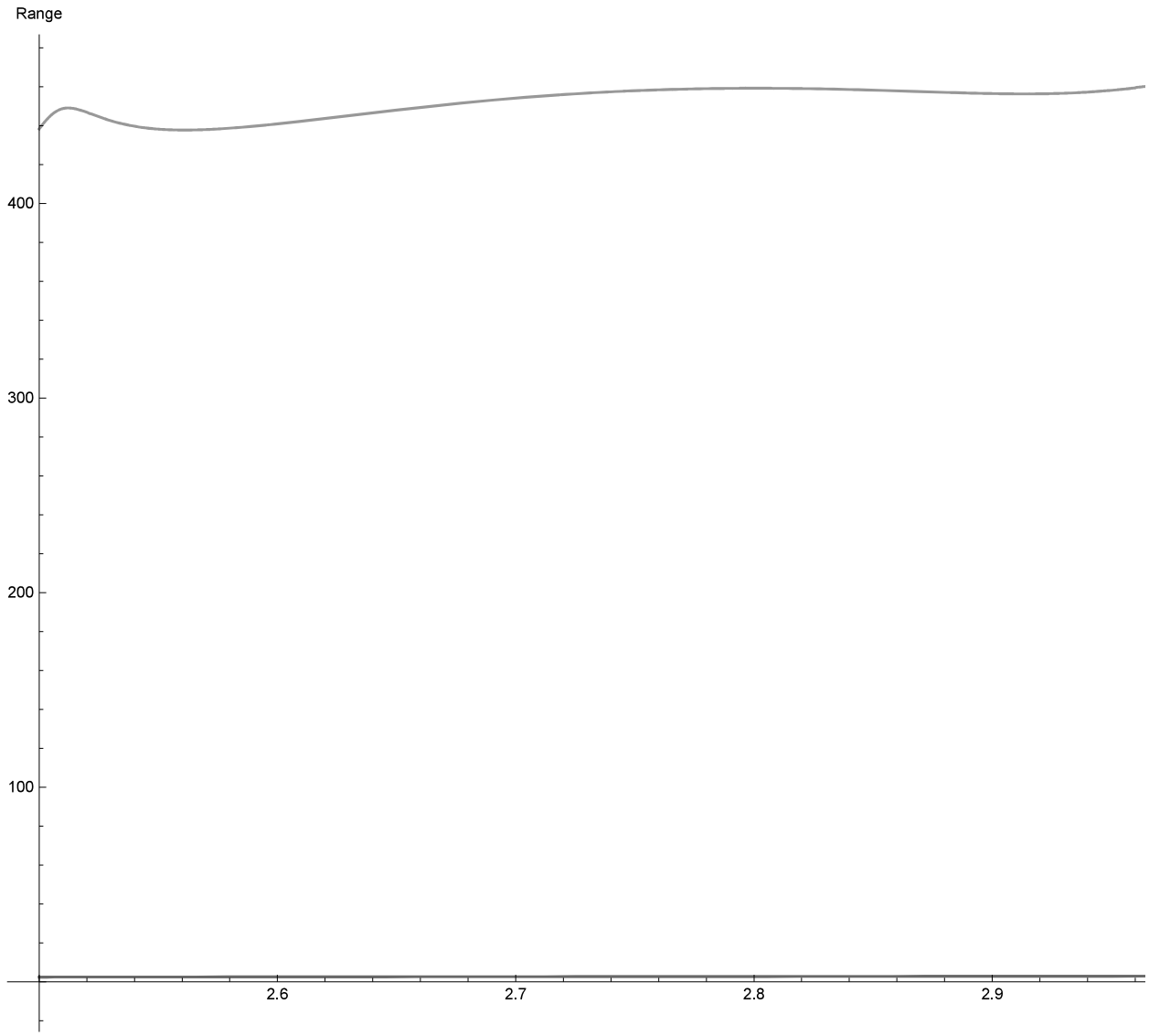



```
DiscretePlot[{x, range /. RangeOfProjectile[6.4, b, 0.1, d, 8, m1, m2, x]},
 {x, 2, 3.15, 0.01}, AxesLabel → {"Release Angle", "Range"}]
```

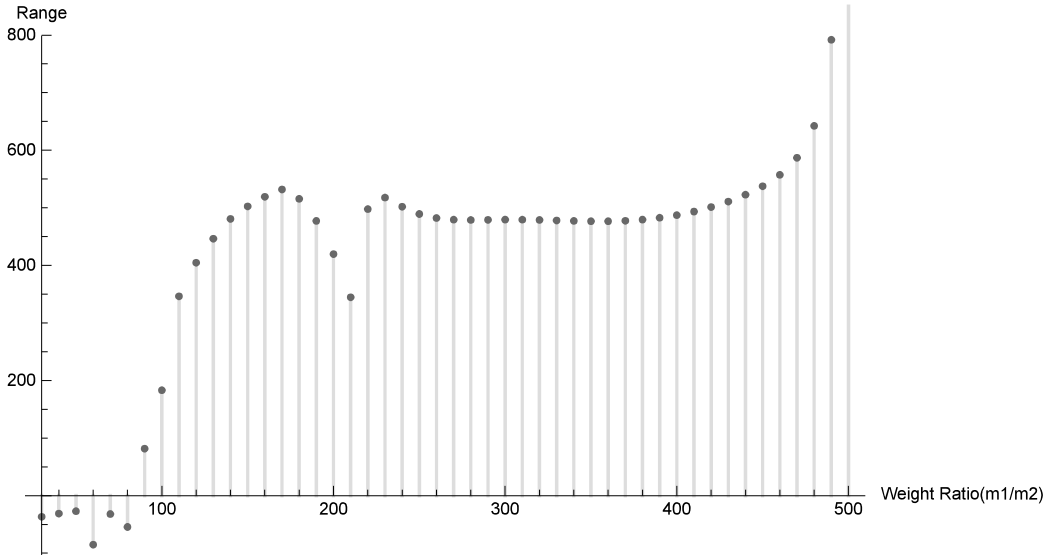
```
Plot[Te1 + V1 /. sol1, {t, 0, end}, AxesLabel → {"t(Seconds)", "Te"}, PlotRange → Full]
```



```
Plot[{x, range /. RangeOfProjectile[6.4, b, 0.1, d, 8, m1, m2, x, 1]},
 {x, 2.5, 3.15}, AxesLabel → {"Release Angle", "Range"}]
```



```
DiscretePlot[{range /. RangeOfProjectile[6.4, b, 0.1, d, 8, x, 1, 2.83, 1]},
 {x, 30, 500, 10}, AxesLabel → {"Weight Ratio(m1/m2)", "Range"}]
```

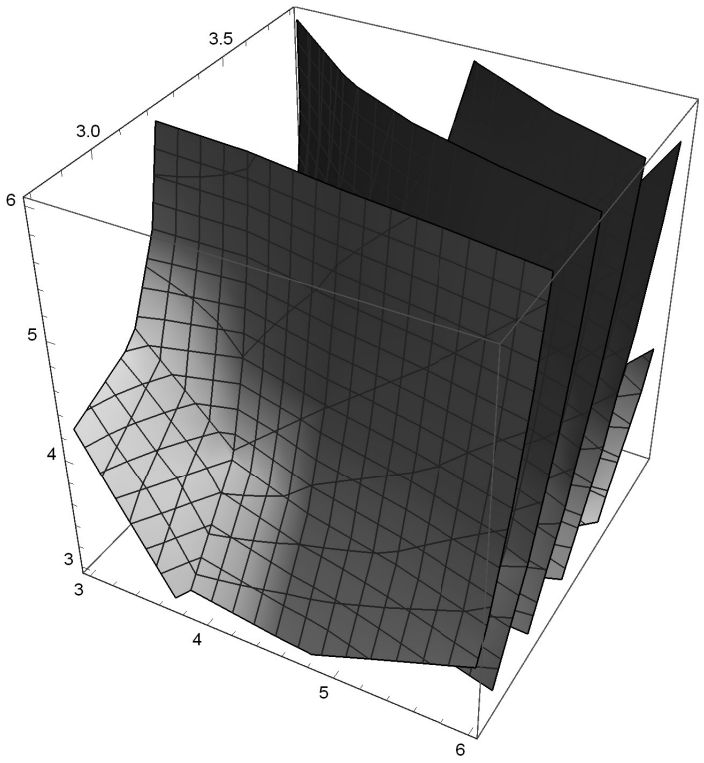


sol1

$\{\{\gamma \to$ InterpolatingFunction[Domain: {{0., 2.}} Output: scalar],

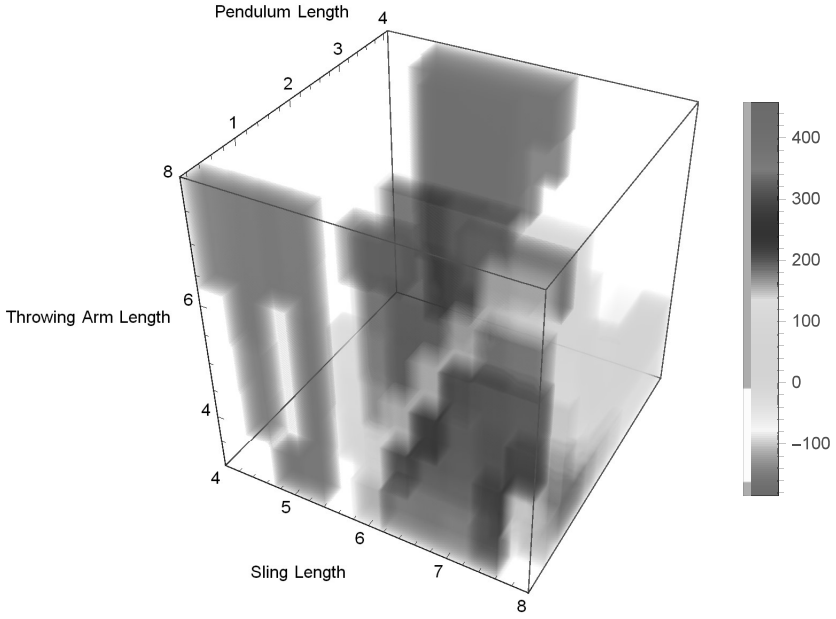
$\theta \to$ InterpolatingFunction[Domain: {{0., 2.}} Output: scalar]$\}\}$

```
data3dr = Table[range /. RangeOfProjectile[z, b, 0.4, d, x, m1, m2, y, 1],
  {z, 3, 6, 0.75}, {y, 2.8, 3.4, 0.15}, {x, 3, 6, 0.75}]
ListContourPlot3D[data3dr, DataRange → {{3, 6}, {2.8, 3.4}, {3, 6}}, Contours → 5]
```

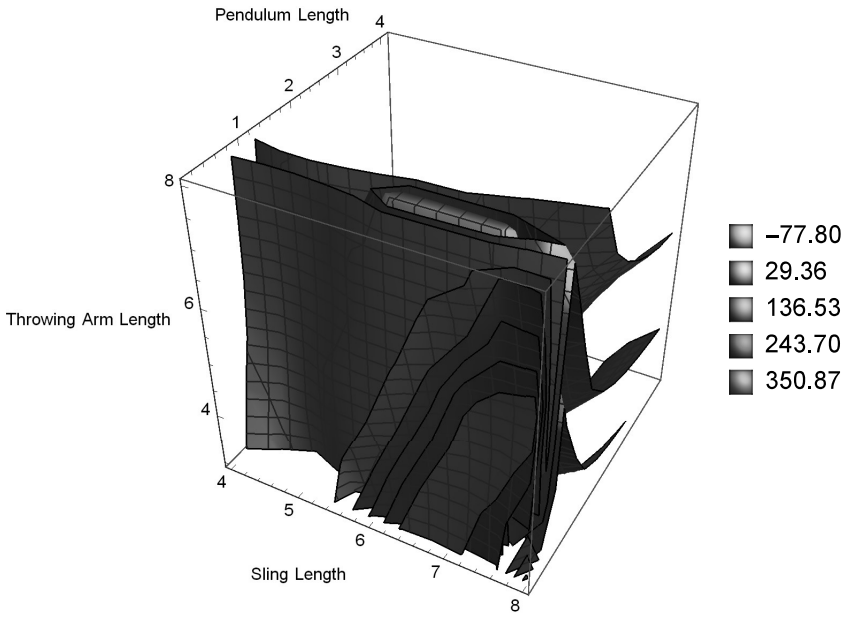



```
Labeled[ListDensityPlot3D[data3dr2[[All, All, All, 1]],
  DataRange → {{4, 8}, {0.1, 4.1}, {3, 8}}, ColorFunction → Hue, PlotLegends → Automatic,
  AxesLabel → {"Sling Length", "Pendulum Length", "Throwing Arm Length"}],
 "Density Plot – Range of projectile at release angle 2.75", Top]
```

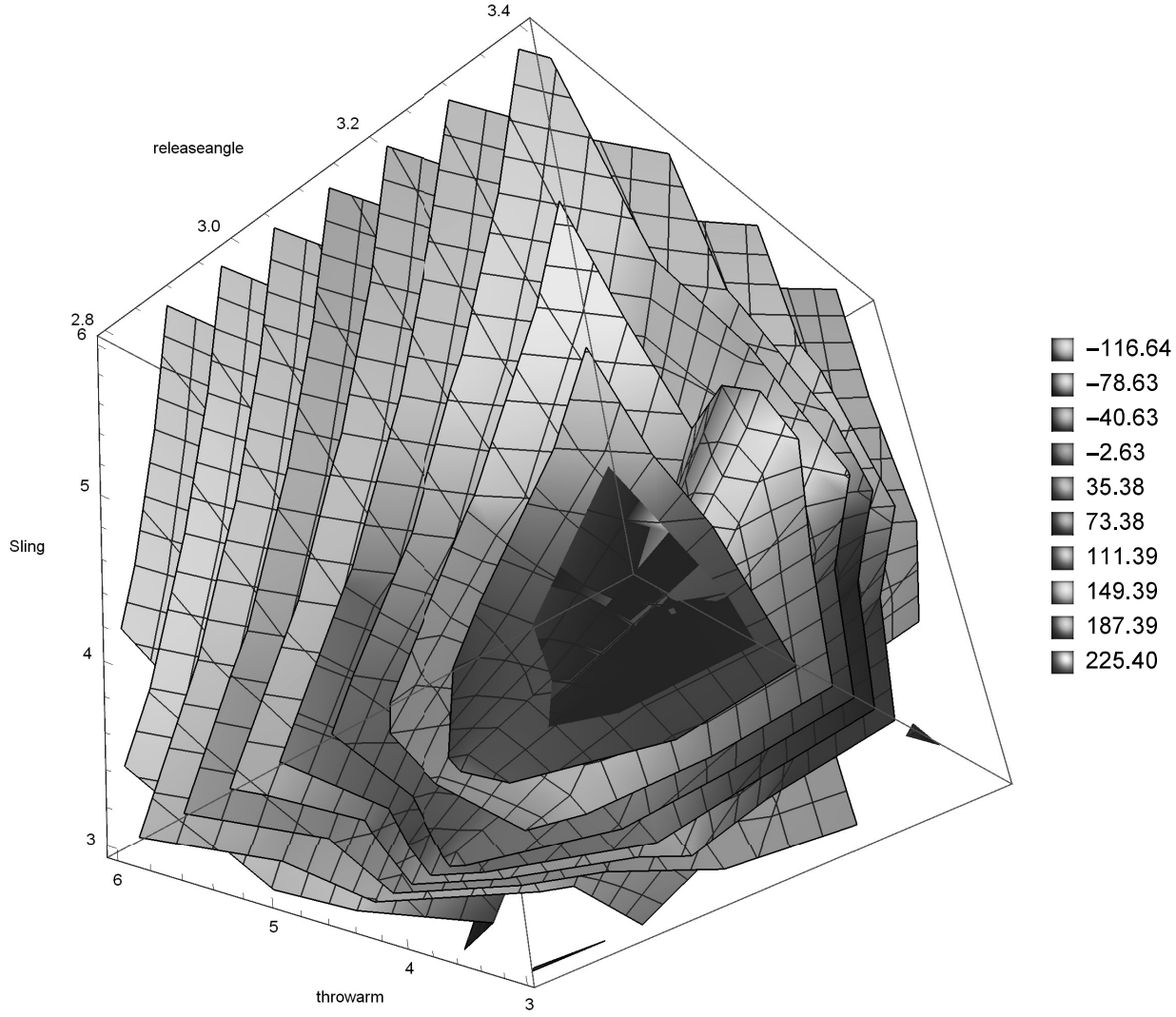
Density Plot – Range of projectile at release angle 2.75



```
Labeled[ListContourPlot3D[data3dr2[[All, All, All, 1]],
  DataRange → {{4, 8}, {0.1, 4.1}, {3, 8}}, Contours → 5, PlotLegends → "Expressions",
  AxesLabel → {"Sling Length", "Pendulum Length", "Throwing Arm Length"}],
 "Contour Plot – Range of projectile at release angle 2.75", Top]
```



```
ListContourPlot3D[data3dr, DataRange → {{3, 6}, {2.8, 3.4}, {3, 6}}, Contours → 10,
 PlotLegends → "Expressions" , AxesLabel → {"throwarm", "releaseangle", "Sling"}]
```




```
Export["test3.dat", data3dr2]
```

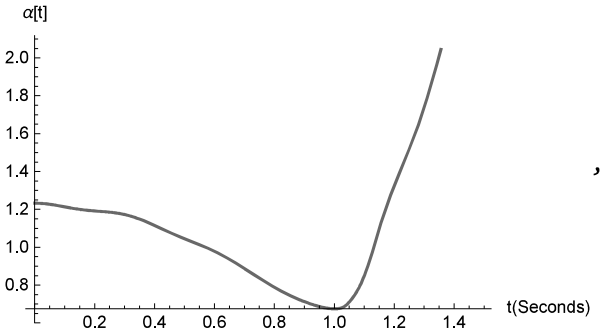
```
test3.dat
```

```
RangeOfProjectile[6.4, b, 0.1, d, 8, m1, m2, 2.83]
```

```
{solution → {{θ → InterpolatingFunction[ Domain: {{0., 1.49}}
                                         Output: scalar ],
  γ → InterpolatingFunction[ Domain: {{0., 1.49}}
                             Output: scalar ],
  α → InterpolatingFunction[ Domain: {{0., 1.49}}
                             Output: scalar ]}},
```

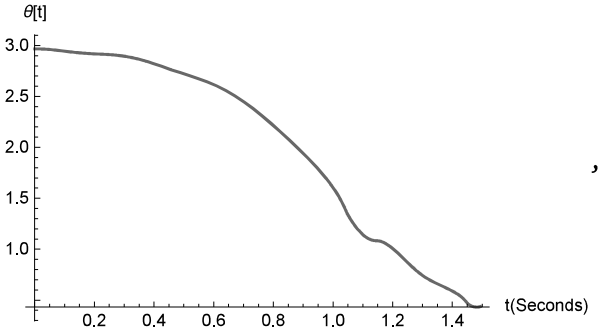
range → 458.928, releaseangle → 45.0032, relVel → 67.0976,

Plot_α →



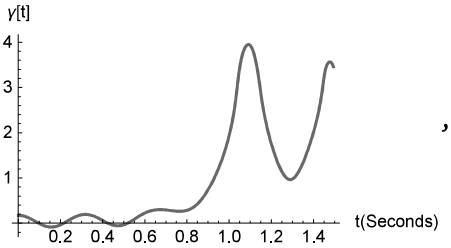
,

Plot_θ →



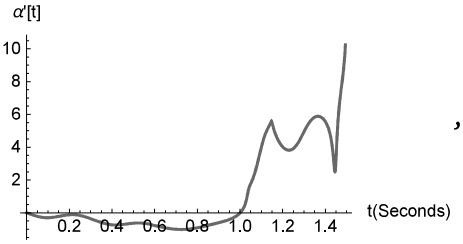
,

Plot_γ →



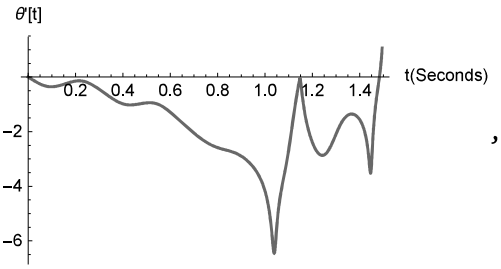
,

Plot_dα →



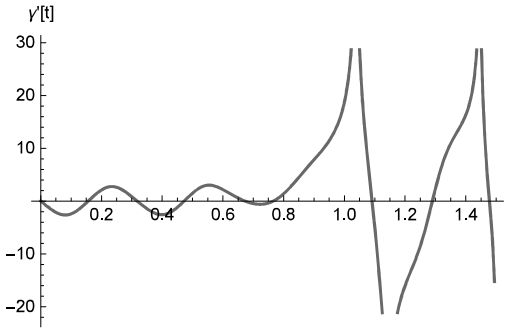
,

Plot_dθ →



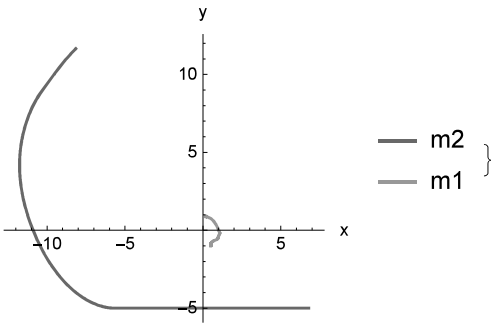
,

Plot_dγ →



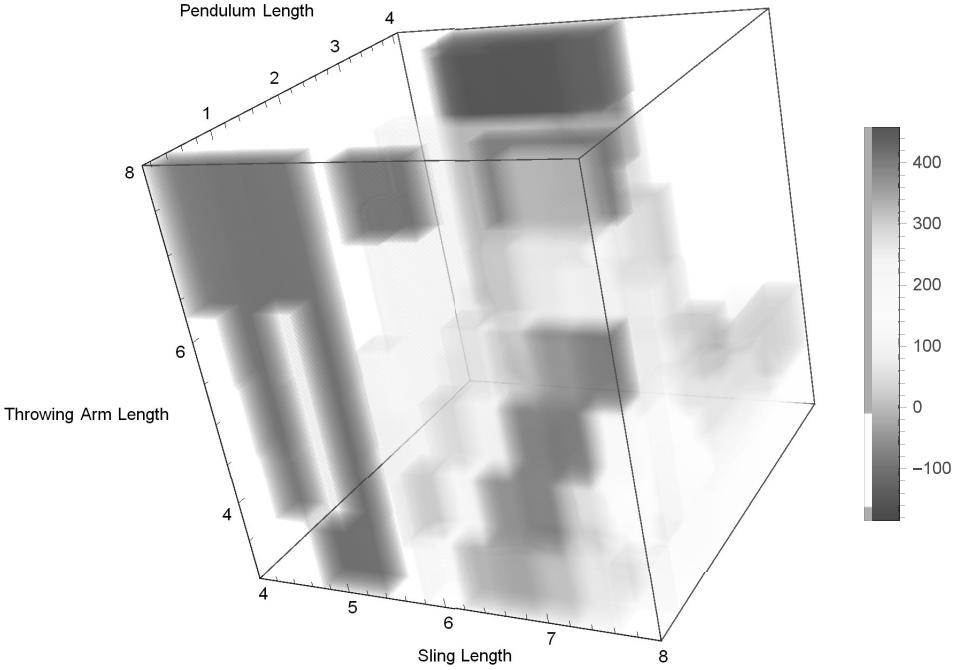
,

Para_P →

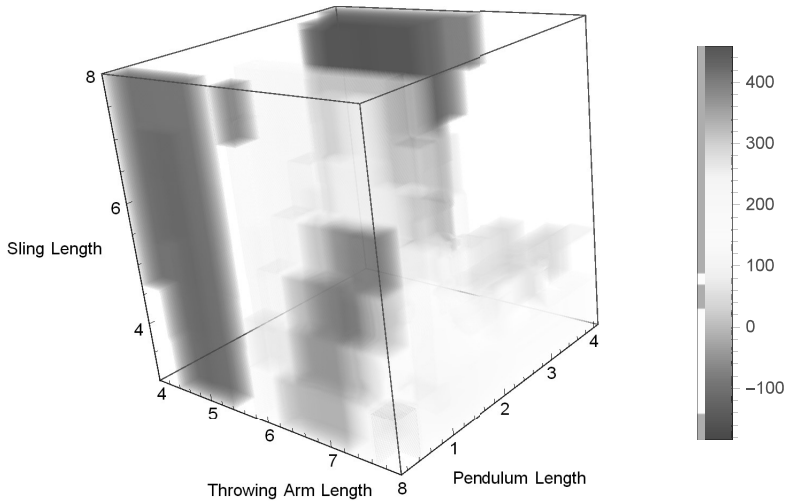


```
Labeled[
 ListDensityPlot3D[data3dr2[[All, All, All, 1]], DataRange → {{4, 8}, {0.1, 4.1}, {3, 8}},
  ColorFunction → "TemperatureMap", PlotLegends → Automatic,
  AxesLabel → {"Sling Length", "Pendulum Length", "Throwing Arm Length"}],
 "Range of projectile at release angle 2.75", Top]
Labeled[ListDensityPlot3D[data3dr2[[All, All, All, 2]],
  DataRange → {{4, 8}, {0.1, 4.1}, {3, 8}},
  PlotLegends → Automatic, ColorFunction → "TemperatureMap",
  AxesLabel → {"Throwing Arm Length", "Pendulum Length", "Sling Length"}],
 "Range of projectile at release angle 2.83", Top]
Labeled[ListDensityPlot3D[data3dr2[[All, All, All, 3]],
  DataRange → {{4, 8}, {0.1, 4.1}, {3, 8}},
  PlotLegends → Automatic, ColorFunction → "TemperatureMap",
  AxesLabel → {"Sling Length", "Throwing Arm Length", "Pendulum Length"}],
 "Range of projectile at release angle 2.91", Top]
Labeled[ListDensityPlot3D[data3dr2[[All, All, All, 4]],
  DataRange → {{4, 8}, {0.1, 4.1}, {3, 8}},
  PlotLegends → Automatic, ColorFunction → "TemperatureMap",
  AxesLabel → {"Throwing Arm Length", "Pendulum Length", "Sling Length"}],
 "Range of projectile at release angle 2.99", Top]
Labeled[ListDensityPlot3D[data3dr2[[All, All, All, 5]],
  DataRange → {{4, 8}, {0.1, 4.1}, {3, 8}},
  PlotLegends → Automatic, ColorFunction → "TemperatureMap",
  AxesLabel → {"Throwing Arm Length", "Pendulum Length", "Sling Length"}],
 "Range of projectile at release angle 3.07", Top]
Labeled[ListDensityPlot3D[data3dr2[[All, All, All, 6]],
  DataRange → {{4, 8}, {0.1, 4.1}, {3, 8}},
  PlotLegends → Automatic, ColorFunction → "TemperatureMap",
  AxesLabel → {"Throwing Arm Length", "Pendulum Length", "Sling Length"}],
 "Range of projectile at release angle 3.15", Top]
```

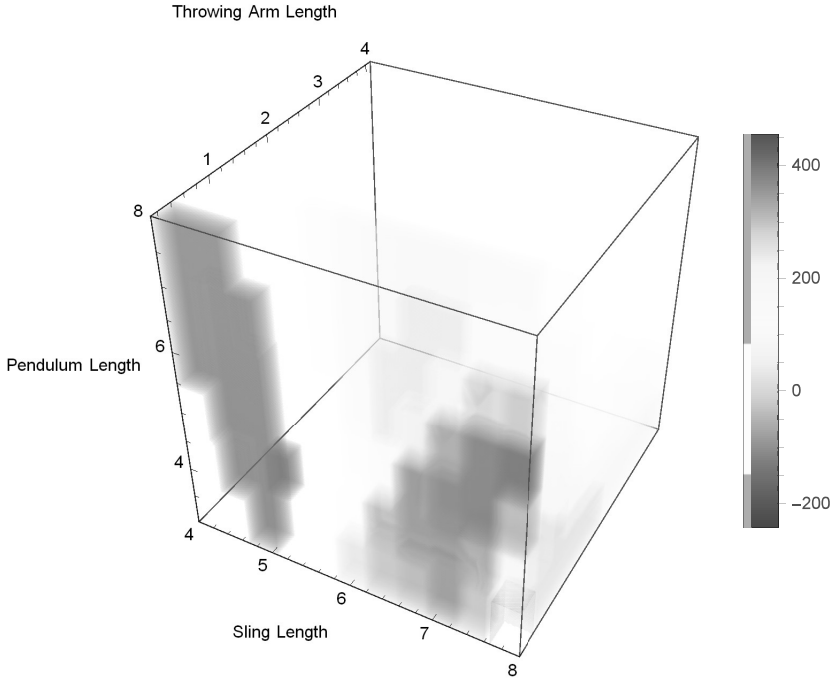
"Range of projectile at release angle 2.75"



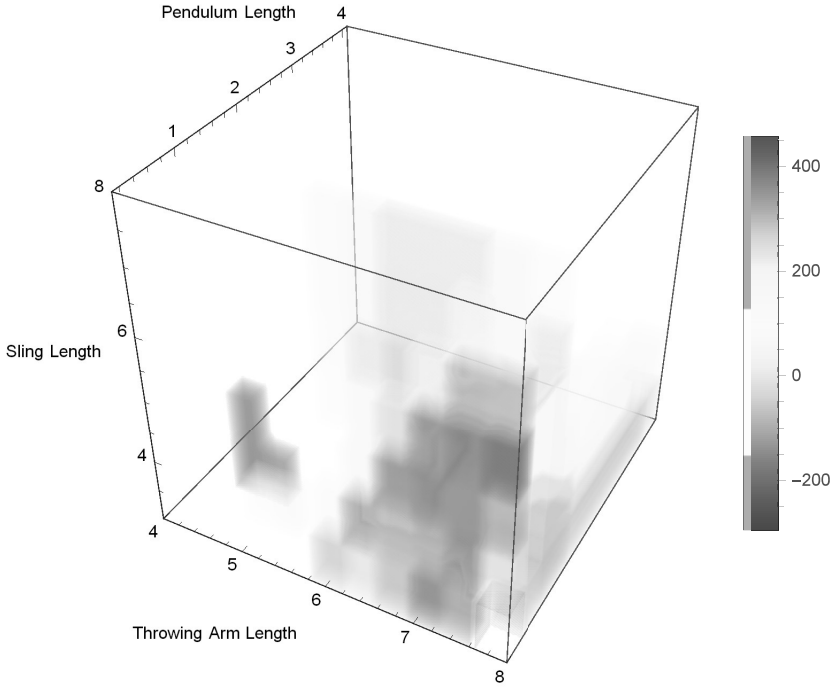
"Range of projectile at release angle 2.83"

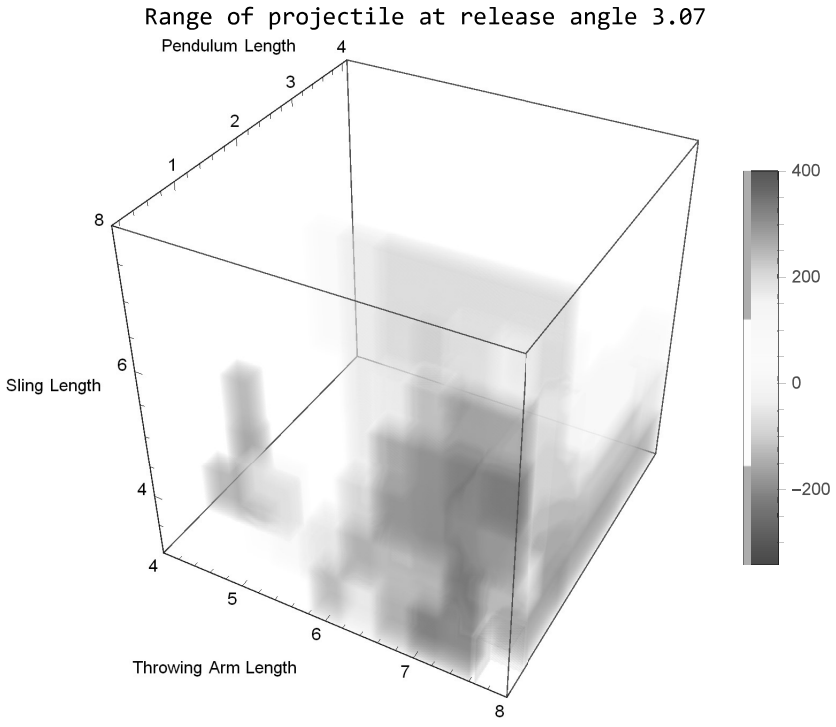


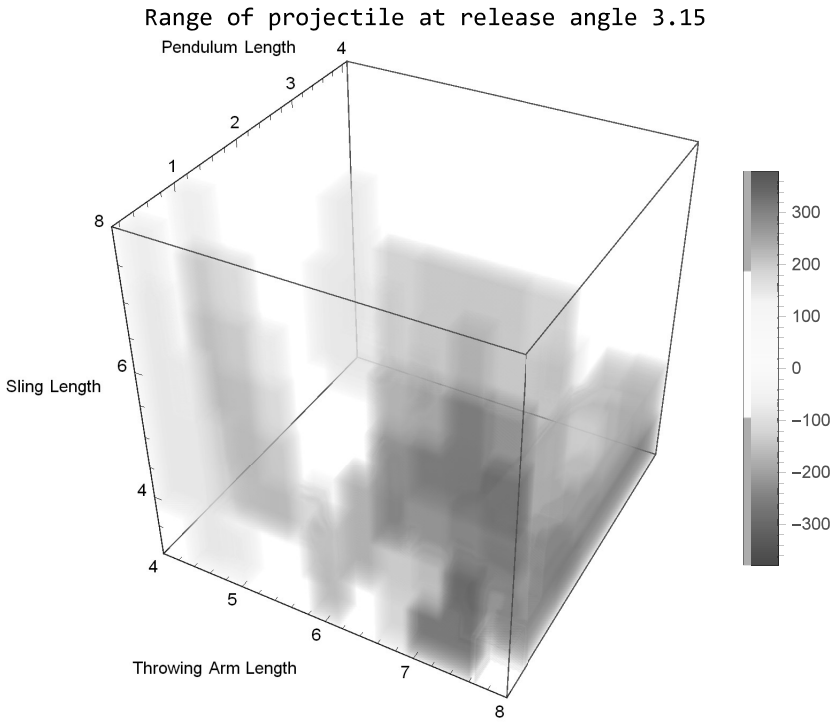
Range of projectile at release angle 2.91



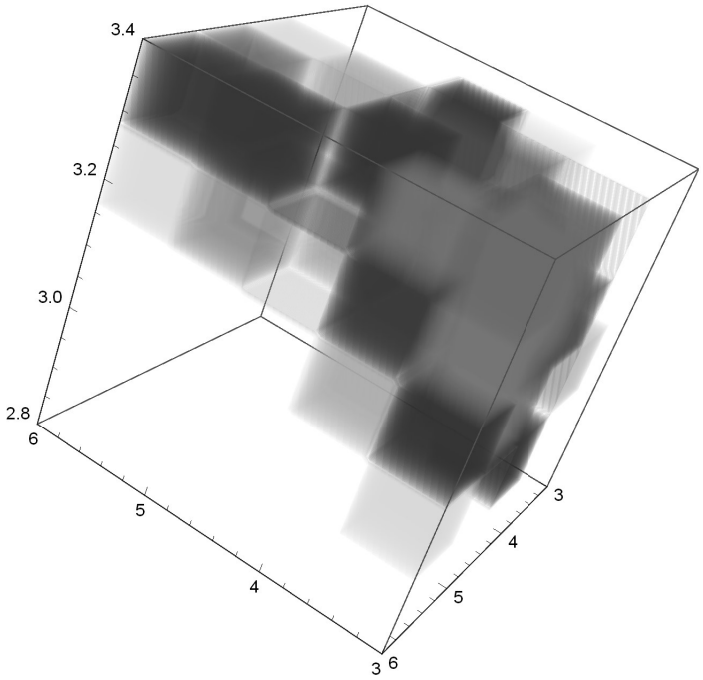
Range of projectile at release angle 2.99



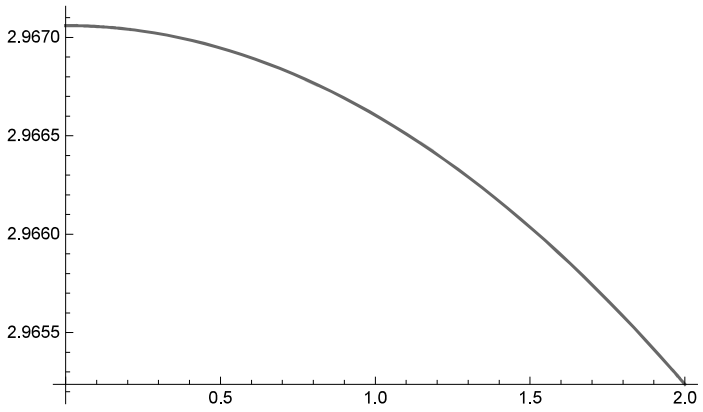

Range of projectile at release angle 3.07
Pendulum Length
Sling Length
Throwing Arm Length
400
200
0
−200


Range of projectile at release angle 3.15
Pendulum Length
Sling Length
Throwing Arm Length
300
200
100
0
−100
−200
−300

```
ListDensityPlot3D[data3dr, DataRange → {{, 6}, {2.8, 3.4}, {3, 6}},
 ColorFunction → Hue, PlotLegends → "Expressions"]
```



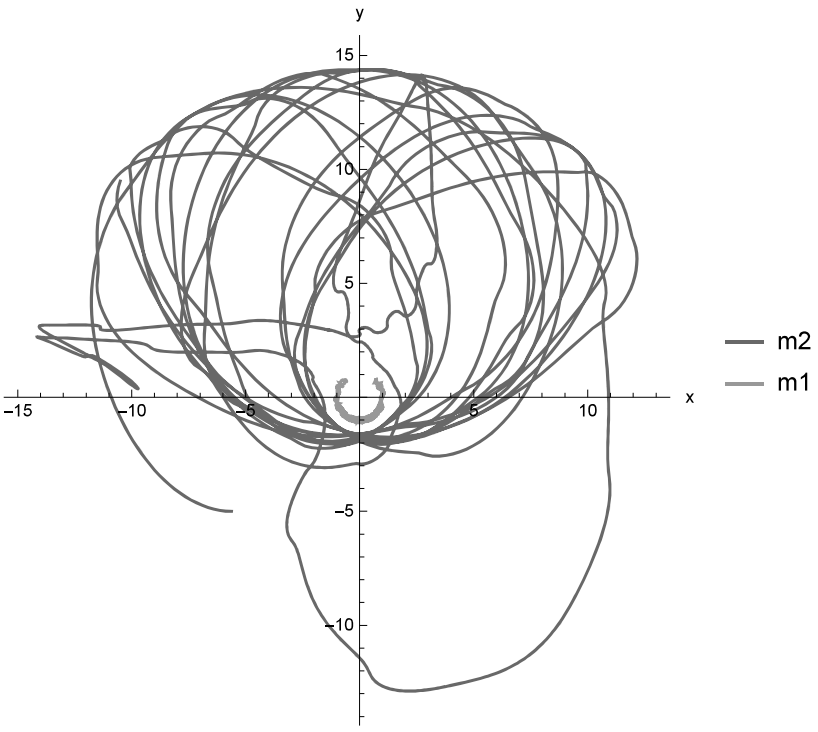

```
Plot[θ[t] /. sol1, {t, 0, 2}]
```




```
RangeOfProjectile[6.4, b, 0.1, d, 8, m1, m2, 2.83]
```

```
{{γ → InterpolatingFunction[ Domain: {{1.15, 20.}}
                             Output: scalar ],
 θ → InterpolatingFunction[ Domain: {{1.15, 20.}}
                             Output: scalar ],
 α → InterpolatingFunction[ Domain: {{1.15, 20.}}
                             Output: scalar ]}}
```

```
ParametricPlot[{{First[x2[t] /. sol2], First[y2[t] /. sol2]},
  {First[x1[t] /. sol2], First[y1[t] /. sol2]}}, {t, end, 20},
 AxesLabel → {"x", "y"}, PlotLegends → {"m2", "m1"}, PlotRange → Full]
```




```
ListDensityPlot3D[data3dr, DataRange → {{, 6}, {2.8, 3.4}, {3, 6}},
 ColorFunction → Hue, PlotLegends → "Expressions"]
```