Delft University of Technology
Faculty of Electrical Engineering, Mathematics
and Computer Science
Delft Institute of Applied Mathematics

'Modelling the movements of a counterweight trebuchet while firing'

Report for the purpose of the
Delft Institute of Applied Mathematics
as part of obtaining the degree of

BACHELOR OF SCIENCE
in
APPLIED MATHEMATICS

by

ROBIN DE JONG

Delft, The Netherlands
January 2020

BSc report APPLIED MATHEMATICS

"Modelling the movements of a counterweight trebuchet while firing"

ROBIN DE JONG

Delft University of Technology

Supervisor

Dr. B.J. Meulenbroek

Other committee members

Dr.ir. F.J. Vermolen

Dr. F.G. Spandaw

Dr. P.M. Visser

January, 2020 Delft

Contents

1 Introduction

The counterweight trebuchet is a type of catapult from the late middle ages. This siege engine uses a swinging arm with a sling attached to throw a projectile. After lifting the heavy counterweight with manpower, all that potential energy can be released within a second. This made it possible to fire a projectile with enormous force. In figure 1 we see a drawing of a counterweight trebuchet.

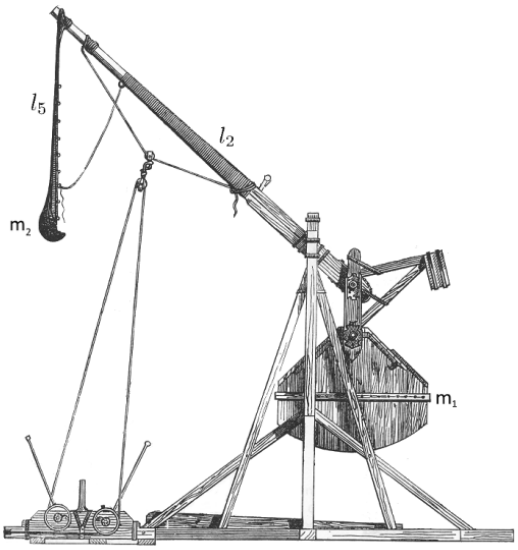


Figure 1: In this drawing of a counterweight trebuchet we can see the counterweight m_1, the projectile m_2, the throwing arm l_2 and the sling l_5. l_1, l_3 and l_4 are given in figure 2.

1.1 Problem statement

In this project we are going to model the movements of a counterweight trebuchet while firing. We want to develop a simple, yet realistic model.

1.2 Historical background

Late 12^{th} century the counterweight trebuchet was invented in the Mediterranean area. It is not clear when and were exactly. In part the counterweight trebuchet replaced the traction trebuchet. With the traction trebuchet up to hundreds of people would pull on ropes as hard as possible to launch a projectile. Because the range depended on how hard the men would pull, it was not very consistent. The counterweight trebuchet on the other hand was very precise. It could also shoot much heavier projectiles over a longer distance. The counterweight trebuchet did however need an expert to operate. It was also a lot harder to transport, because of this it was often build close to the target. The large forces on the counterweight trebuchet meant that you also needed stronger materials that with the traction trebuchet. Good timber for the main beam was hard to find. Also the projectile stones needed to be strong enough. This all resulted in higher costs for the counterweight trebuchet. The big advantage was that you could actually bring down defensive walls with the counterweight trebuchet relative to the traction trebuchet which could not. However the traction trebuchet was still widely used to attack personnel weaker defence like buildings and to launch projectiles over walls. Around the 1500s the cannon replaced the trebuchet [Purton, 2009].

1.3 Literature review

The first inspiration came from a project were they used the Euler-Lagrange method to model a counterweight trebuchet and build it up angle by angle. They started with a see-saw model, our one angle model, where both the counterweight and the projectile are fixed to the rotating main beam. Then they went to the hinged counterweight model, our two angle model, where they added an angle between the main beam and the counterweight. Finally they ended with a trebuchet with a hinged counterweight and sling, our three angle model, where they added the sling. We see the same method being used for this problem in most other literature as well. In this project the Lagrangian for the one and three angle models are the same as ours, but with the two angle model there is a small calculation error when computing the speed of the counterweight [Rutan and Wieczorek, 2005]. Modeling a counterweight trebuchet is done a few times, but often with small calculation errors or by letting the computer do most of the calculations. We do not exactly know how this is numerically implemented, so we call it a black box. We want to make sure our calculations are correct and do as much as possible by hand. We also used a project that takes the beam weight into account, but uses a lot of black boxes. We use the results to check our equations, but we want to try and see if it is necessary to include the beam weight. Note that there is a typo in the very last step of the one angle model. [Siano, 2013] Our two angle model looks a good deal like a double pendulum. A double pendulum is done quite often, so we can be fairly sure that these calculations are correct [Altic, 2008]. For the three angle model we used a simulation to check the Lagrangian [Mahieu and Brandhuber, 2012]. From a 1992 reconstruction of a counterweight trebuchet in Denmark we know how far it will realistically shoot. We derived the dimensions from this reconstruction from the blueprint and use these for our model. They released at $\frac{1}{9}\pi$ rad and the projectile went 80 meters high and 168 meters far [Hansen, 1992]. Another reconstruction with the same blue prints was build at Warwick Castle and is still in use for museum visitors.

1.4 Methodology

For this model we use the Euler-Lagrange method. This method is a reformulation of the Newton's equations of motion. The Euler-Lagrange method provides a systematic way to determine the equations of motion in any coordinate system. For this method we use the Euler-Lagrange equation

$$\frac{\partial}{\partial t}\Big(\frac{\partial L}{\partial x_i}\Big) - \frac{\partial L}{\partial \dot{x}_i} = 0$$

where L is the Lagrangian depending on $x_1, \ldots, x_n$. In our case the x_i's are the angles θ, ϕ and ψ. A Lagrangian for a certain system is not unique, but we can always find a valid Lagrangian by using $L(x_1, \ldots, x_n) = E_k - E_p$. Even though the Lagrangian is not unique, every valid Lagrangian will result in the same set of equations of motion [Barger and Olsson, 1995].

1.5 General problem

We have a stand l_3, that we see as a line perpendicular to the ground. The stand does not move in our model. The point where the stand touches the ground is our origin $(0, 0)$. The main beam is attached at the top of l_3. The main beam can rotate around the top of l_3 with angle θ. On one side there is a short part of the main beam, l_1, on the other side is a long part of the main beam, l_2. At the end of l_1, there is a counterweight beam l_4. The counterweight beam l_4 can rotate around the end of l_1 with angle ϕ. A large counterweight m_1 is attached to this counterweight beam l_4. At the end of l_2 there is a projectile beam l_5. The projectile beam l_5 can rotate around the end of l_2 with angle ψ. A projectile m_2 is connected to the projectile beam l_5. This is all schematically laid out in figure 2 which is based on figure 1. We neglect the loss of energy due to friction. This involves both friction

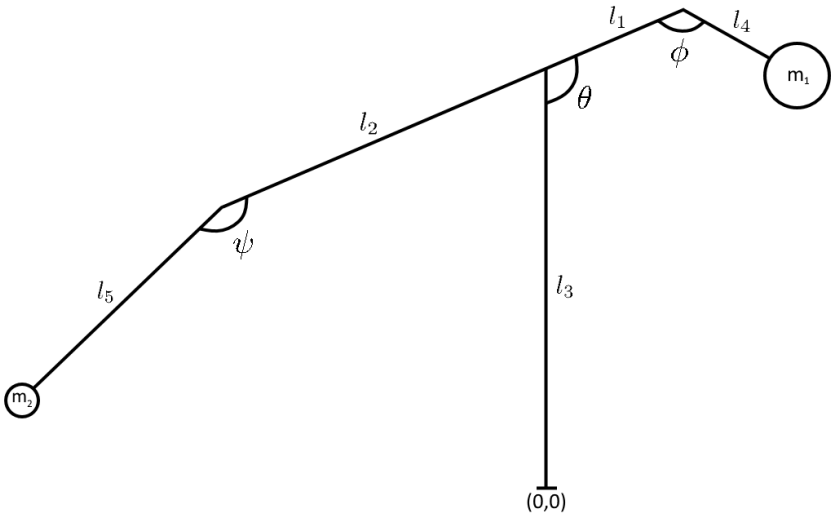


Figure 2: A schematic representation of the general model.

at the three rotation points and air resistance. We also neglect the weight and volume of the device except m_1 and m_2. We expect this to be possible since we choose m_1 quite large. The ground does not play a role in our model. We see m_1 and m_2 as point masses and we see l_1, l_2, l_3, l_4 and l_5 as lines. We assume all the beams to be rigid. The units that we work with are meters, seconds, kilos and radians.

1.6 Variables and parameters

Independent variable

t	time	s

Table 1: The independent variable of our model

Dependent variables

θ	angle between the main beam and the stand	rad
ϕ	angle between the main beam and the counterweight beam	rad
ψ	angle between the main beam and the projectile beam	rad

Table 2: The Dependent variables of our model are the angles, see figure 2.

Parameters

m_1	mass of the counterweight	$2000\,kg$
m_2	mass of the projectile	$15\,kg$
l_1	short part of the main beam, between the θ-joint and the ϕ-joint	$1.2\,m$
l_2	long part of the main beam, between the θ-joint and the ψ-joint	$5.7\,m$
l_3	stand, between the ground and the θ-joint	$3.2\,m$
l_4	counterweight beam, between the ϕ-joint and the counterweight	$1.4\,m$
l_5	projectile beam, between the ψ-joint and the projectile	$5\,m$
g	gravitation constant	$9.81\,m/s^2$
θ_0	θ at $t = 0$	$0.7\pi\,rad$
$\dot{\theta}_0$	$\dot{\theta}$ at $t = 0$	$0\,rad/s$
ϕ_0	ϕ at $t = 0$	$\pi - \theta_0\,rad$
$\dot{\phi}_0$	$\dot{\phi}$ at $t = 0$	$0\,rad/s$
ψ_0	ψ at $t = 0$	$\theta_0 - 0.5\pi\,rad$
$\dot{\psi}_0$	$\dot{\psi}$ at $t = 0$	$0\,rad/s$

Table 3: The parameters of our model derived from the reconstruction in Denmark [Hansen, 1992].

2 The one angle model

For the first model we take our general problem with l_4 and l_5 equal to zero. Only angle θ plays a role in this model, so we call it the one angle model. This is shown schematically in figure 3. For the next model we will add l_4. This means that ϕ plays a role again, so this is the two angle model.

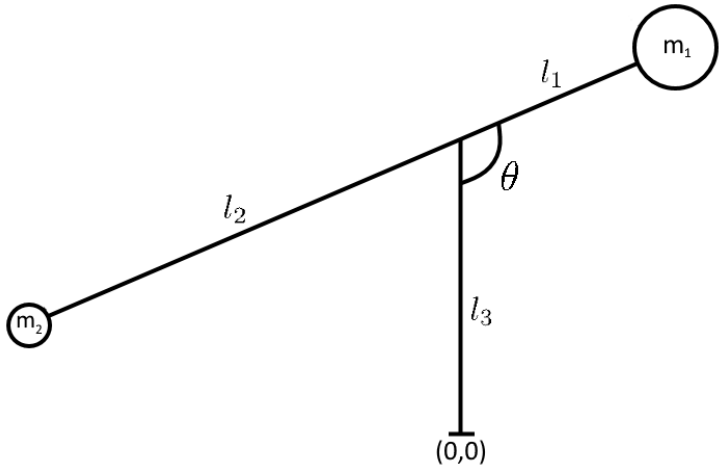


Figure 3: Schematic representation of the one angle model

2.1 Mathematical model

The position of the counterweight is given by r_{1a} and the position of the projectile is given by r_{2a}, see figure 4.

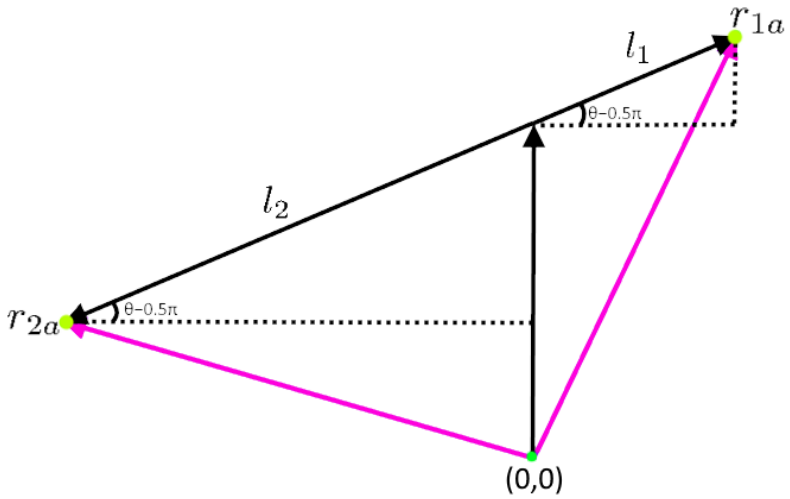


Figure 4: Vector representation of the one angle model

From figure 4 we observe

$$r_{1a} = \begin{bmatrix} 0 \\ l_3 \end{bmatrix} + \begin{bmatrix} l_1 \cos(\theta - \frac{1}{2}\pi) \\ l_1 \sin(\theta - \frac{1}{2}\pi) \end{bmatrix} = \begin{bmatrix} l_1 \sin(\theta) \\ -l_1 \cos(\theta) + l_3 \end{bmatrix} \tag{1}$$

$$r_{2a} = \begin{bmatrix} 0 \\ l_3 \end{bmatrix} - \begin{bmatrix} l_2 \cos(\theta - \frac{1}{2}\pi) \\ l_2 \sin(\theta - \frac{1}{2}\pi) \end{bmatrix} = \begin{bmatrix} -l_2 \sin(\theta) \\ l_2 \cos(\theta) + l_3 \end{bmatrix} \tag{2}$$

With the y coördinates of the masses we can compute the potential energy of the counterweight and projectile.

$$E_{p1a} = m_1 g(-l_1 \cos(\theta) + l_3) \qquad E_{p2a} = m_2 g(l_2 \cos(\theta) + l_3) \tag{3}$$

Adding E_{p1a} and E_{p2a} we find

$$E_{pa} = -g(m_1 l_1 - m_2 l_2) f \cos(\theta) + g l_3 (m_1 + m_2) \tag{4}$$

To compute the kinetic energy of the masses we need the speed of the masses. Using (1) and (2) we find

$$v_{1a}^2 = (l_1 \cos(\theta)\dot{\theta})^2 + (l_1 \sin(\theta)\dot{\theta})^2 = l_1 \dot{\theta} \qquad v_{2a}^2 = (-l_2 \cos(\theta)\dot{\theta})^2 + (-l_2 \sin(\theta)\dot{\theta})^2 = l_2 \dot{\theta}$$

With these speeds we can compute the kinetic energy for the counterweight and the projectile.

$$E_{k1a} = \frac{1}{2} m_1 l_1^2 \dot{\theta}^2 \qquad E_{k2a} = \frac{1}{2} m_2 l_2^2 \dot{\theta}^2 \tag{5}$$

Adding E_{k1a} and E_{k2a} gives us the total kinetic energy

$$E_{ka} = \frac{1}{2}(m_1 l_1^2 + m_2 l_2^2)\dot{\theta}^2 \tag{6}$$

We find our Lagrangian, $L_a = E_{ka} - E_{pa}$, by using equation (4) and (6).

$$L_a = \frac{1}{2}(m_1 l_1^2 + m_2 l_2^2)\dot{\theta}^2 + g(m_1 l_1 - m_2 l_2) f \cos(\theta) - g l_3 (m_1 + m_2) \tag{7}$$

In order to use the Euler-Lagrange equation for θ, $\frac{\partial L_a}{\partial \theta} - \frac{\partial}{\partial t}\left(\frac{\partial L_a}{\partial \dot{\theta}}\right) = 0$, we need to compute $\frac{\partial L_a}{\partial \theta}$ and $\frac{\partial}{\partial t}\left(\frac{\partial L_a}{\partial \dot{\theta}}\right)$. By differentiating the Lagrangian, equation (7), with respect to θ we find

$$\frac{\partial L_c}{\partial \theta} = -g(m_1 l_1 - m_2 l_2)\sin(\theta) \tag{8}$$

Differentiating the Lagrangian, equation (7), with respect to $\dot{\theta}$ yields

$$\frac{\partial L_c}{\partial \dot{\theta}} = (m_1 l_1^2 + m_2 l_2^2) \tag{9}$$

Differentiating equation (9) with respect to t, gives us

$$\frac{\partial}{\partial t}\left(\frac{\partial L_a}{\partial \dot{\theta}}\right) = (m_1 l_1^2 + m_2 l_2^2)\ddot{\theta} \tag{10}$$

By subtracting (10) from (8), we finally obtain

$$(m_1 l_1^2 + m_2 l_2^2)\ddot{\theta} + g(m_1 l_1 - m_2 l_2)\sin(\theta) = 0$$

Solving for $\ddot{\theta}$ gives

$$\ddot{\theta} = -\frac{g(m_1 l_1 - m_2 l_2)}{(m_1 l_1^2 + m_2 l_2^2)} \sin(\theta) \tag{11}$$

Note that masses m_1 and m_2 and lengths l_1 and l_2 are nonzero by assumption, which means that denominator never becomes zero. Furthermore we set

$$\theta(t=0) = \theta_0 \qquad \dot{\theta}(t=0) = \dot{\theta}_0 \tag{12}$$

as initial conditions, which means that equations (11) and (12) fully specify our model. In the literature we see the same equation for $\ddot{\theta}$ [Rutan and Wieczorek, 2005, 8][Siano, 2013, 9].

2.2 Analytic solution

Our numerical solution is dependent on the time step size we take. To see how accurate our numerical solution is, we can try to get an analytic solution and compare these solutions. Since our expression for $\ddot{\theta}$, equation (11), is quite simple in this case, we can try to express $\dot{\theta}$ in terms of θ.

$$\ddot{\theta} = -\frac{g(m_1 l_1 - m_2 l_2)}{(m_1 l_1^2 + m_2 l_2^2)} \sin(\theta)$$

Multiplying both sides with $\dot{\theta}$ yields

$$\ddot{\theta}\dot{\theta} = -\frac{g(m_1 l_1 - m_2 l_2)}{(m_1 l_1^2 + m_2 l_2^2)} \sin(\theta)\dot{\theta}$$

Integrating both sides with respect to t gives

$$\frac{1}{2}\dot{\theta}^2 = \frac{g(m_1 l_1 - m_2 l_2)}{(m_1 l_1^2 + m_2 l_2^2)} \cos(\theta) + \alpha \tag{13}$$

We use the initial conditions, see equation (12), to solve for α.

$$\alpha = \frac{1}{2}\dot{\theta}_0^2 - \frac{g(m_1 l_1 - m_2 l_2)}{(m_1 l_1^2 + m_2 l_2^2)} \cos(\theta_0) \tag{14}$$

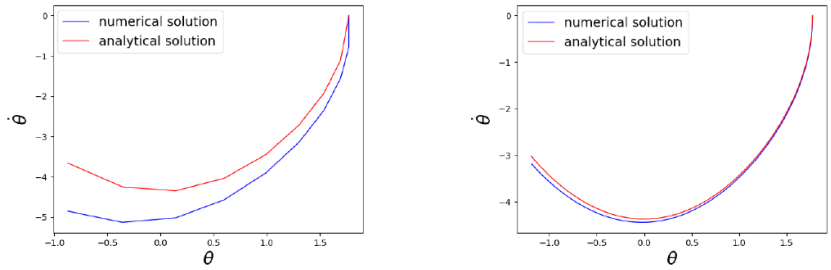
Combining equations (13) and (14) and solving for $\dot{\theta}$ we obtain

$$\dot{\theta}(\theta) = \begin{cases} -\sqrt{2\frac{g(m_1 l_1 - m_2 l_2)}{(m_1 l_1^2 + m_2 l_2^2)}[\cos(\theta) - \cos(\theta_0)]} + \dot{\theta}_0 & 0 < \theta < \pi \\ \\ \sqrt{2\frac{g(m_1 l_1 - m_2 l_2)}{(m_1 l_1^2 + m_2 l_2^2)}[\cos(\theta) - \cos(\theta_0)]} + \dot{\theta}_0 & -\pi < \theta < 0 \end{cases} \tag{15}$$

If $0 < \theta < \pi$ we get a negative sign, since our $\ddot{\theta}$ is negative and vice versa for $-\pi < \theta < 0$.

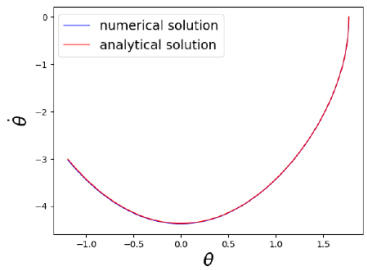
2.3 Numerical solution

We choose our $\theta_0 = 0.8\pi\, rad$ and $\dot{\theta}_0 = 0\, rad/s$. Together with equation (11) for $\ddot{\theta}$ our model is now fully specified, but we still have to find a method to compute the results. Because we have an equation for $\dot{\theta}$, see equation (15), we can plot a phase space for different methods. We do this for the most simplistic method, the Euler forward method, and the most standard method when using Python, scipy.integrate.odeint. Taking $0 \leq t \leq 1$ and using the Euler forward method gives us

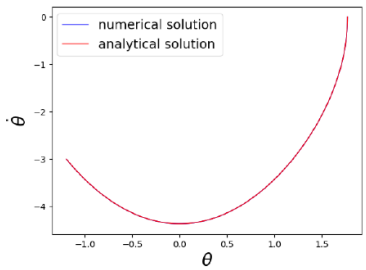


(a) $\Delta t = 0.1$

(b) $\Delta t = 0.01$



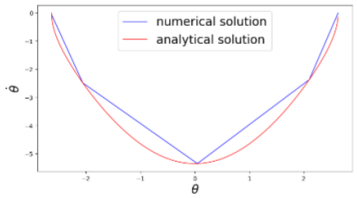
(c) $\Delta t = 0.001$



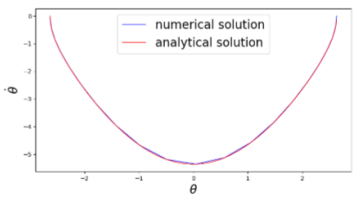
(d) $\Delta t = 0.0001$

Figure 5: Phase spaces using the Euler forward method with different time steps sizes Δt

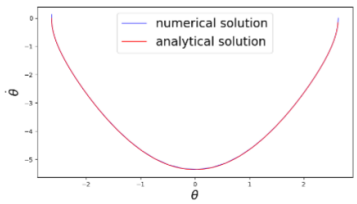
To get accurate result with the Euler forward method, we should choose our time step Δt at least 0.001. The Python function initially uses the Adams–Bashforth method, but switches to the Backward differentiation formula if the differential equation turns out to be stiff. Taking $0 \leq t \leq 2$ and using the Python function gives us



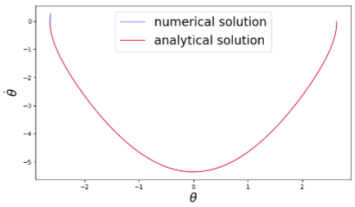
(a) $\Delta t = 0.5$



(b) $\Delta t = 0.1$



(c) $\Delta t = 0.05$



(d) $\Delta t = 0.01$

Figure 6: Phase spaces using the Python function with different time steps sizes Δt

With the Python function scipy.integrate.odeint, we get accurate results from step size $\Delta t = 0.01$. This is a tenth of the step size of the Euler forward method. So we will use the Python function cipy.integrate.odeint from now on. For the one angle model is time step size $\Delta t = 0.01$ enough, but when we make our model more complicated, we have to check if this is small enough.

When we use this to plot our angle and angular velocity as functions of time, we get a feeling for what our one angle trebuchet is doing.

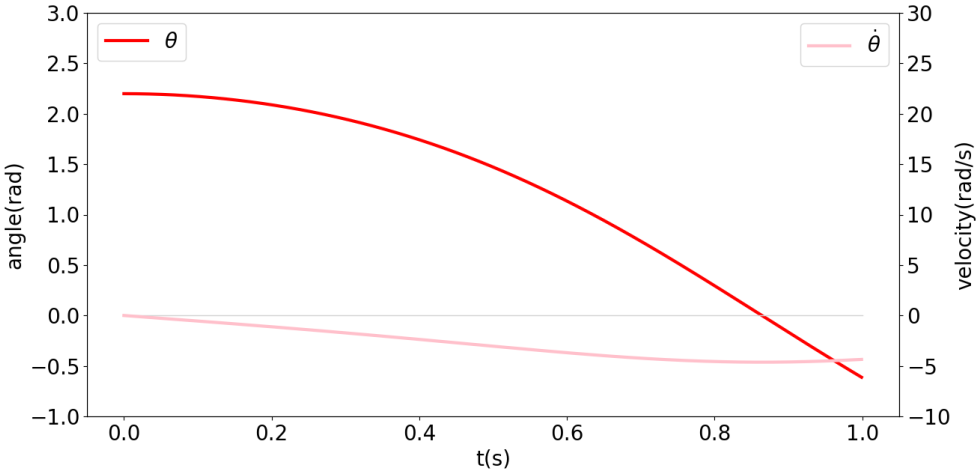


Figure 7: In this plot we see θ and $\dot{\theta}$ as a function of time. We used the one angle model with $\Delta t = 10^{-2}$ and $T = 1$. We see θ accelerating at first which is what we would expect and what we need in order to fire something.

We can also look at at longer period of time.

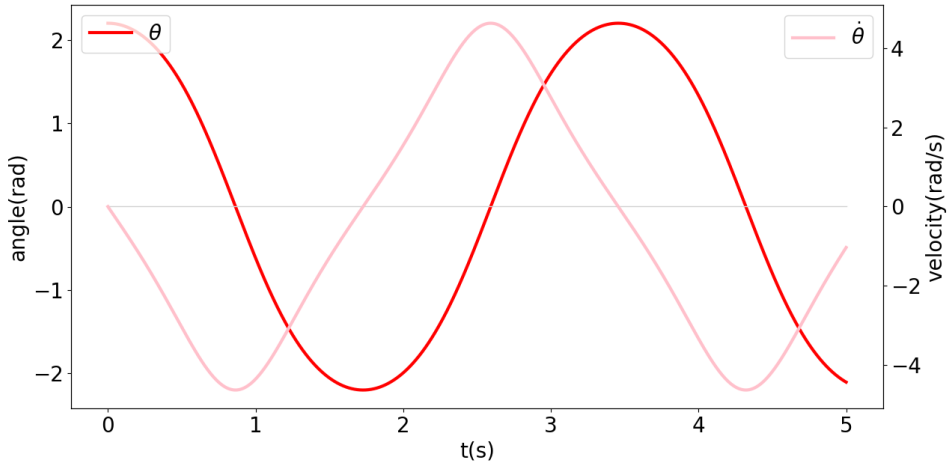


Figure 8: In this plot we see θ and $\dot{\theta}$ as a function of time. We used the one angle model with $\Delta t = 10^{-2}$ and $T = 5$. We see that the solution is periodic and looks like a cosine function for $\theta(t)$, which is to be expected when we look at the analytic solution (15).

2.4 Firing

We call the moment when we release the projectile t_R with corresponding release angle θ_R. For $t \leq t_R$ we know the position of the projectile r_{2a}, see equation (2).

$$r_{2a}(t) = \begin{bmatrix} -l_2 \sin(\theta) \\ l_2 \cos(\theta) + l_3 \end{bmatrix} \tag{16}$$

We can differentiate r_{2a} with respect to t.

$$\dot{r}_{2a}(t) = \begin{bmatrix} \cos(\theta) \cdot (-\dot{\theta} l_2) \\ \sin(\theta) \cdot (-\dot{\theta} l_2) \end{bmatrix} \tag{17}$$

For $t \geq t_R$ we know the acceleration of the projectile. Since we don't take friction into account, $\ddot{r}_2(t)$ reduces to

$$\ddot{r}_2(t) = \begin{bmatrix} 0 \\ -g \end{bmatrix}$$

We can integrate $\ddot{r}_2(t)$ with respect to t.

$$\dot{r}_2(t) = \begin{bmatrix} \beta \\ -g \cdot (t - t_R) + \gamma \end{bmatrix} \tag{18}$$

We can again integrate $\dot{r}_2(t)$ with respect to t.

$$r_2(t) = \begin{bmatrix} \beta \cdot (t - t_R) + \delta \\ -\frac{1}{2} g \cdot (t - t_R)^2 + \gamma \cdot (t - t_R) + \epsilon \end{bmatrix} \tag{19}$$

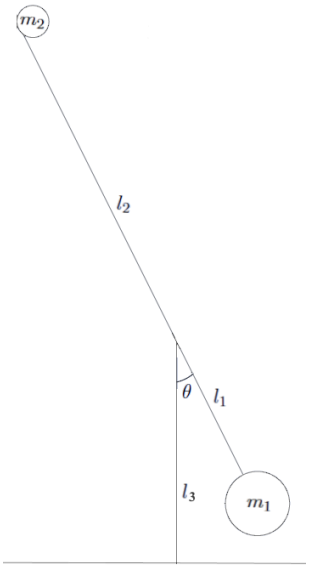


Figure 9: Schematic representation of the one angle trebuchet at the release moment

At $t = t_R$ we know that the speed and position of the projectile should be the same. So $\dot{r}_{2a}(t_R)$, see equation (17), and $\dot{r}_2(t_R)$, see equation (19), should be equal to each other. Also $r_{2a}(t_R)$, see equation (16), and $r_2(t_R)$, see equation (18), should be equal. From $\dot{r}_{2a}(t_R) = \dot{r}_2(t_R)$ follows

$$\beta(\theta_R) = \cos(\theta_R) \cdot (-\dot{\theta}_R l_2)$$
$$\gamma(\theta_R) = \sin(\theta_R) \cdot (-\dot{\theta}_R l_2)$$

From $r_{2a}(t_R) = r_2(t_R)$ follows

$$\delta(\theta_R) = -l_2 \sin(\theta_R)$$
$$\epsilon(\theta_R) = l_2 \cos(\theta_R) + l_3$$

We call the moment that the projectile hits the ground t_E. At t_E we know that the height is zero, so we can calculate t_E

$$t_E(\theta_R) = \frac{\gamma(\theta_R) + \sqrt{\gamma(\theta_R)^2 + 2g\epsilon(\theta_R)}}{g} + t_R \tag{20}$$

With t_E we can calculate the range and impact with a certain release angle.

$$s(\theta_R) = \beta(\theta_R)(t_E(\theta_R) - t_R) \tag{21}$$

$$E(\theta_R) = \frac{1}{2} m_2 (\beta(\theta_R)^2 + (-g t_E(\theta_R) \gamma(\theta_R))^2) \tag{22}$$

When we variate θ_R, the range and impact changes.

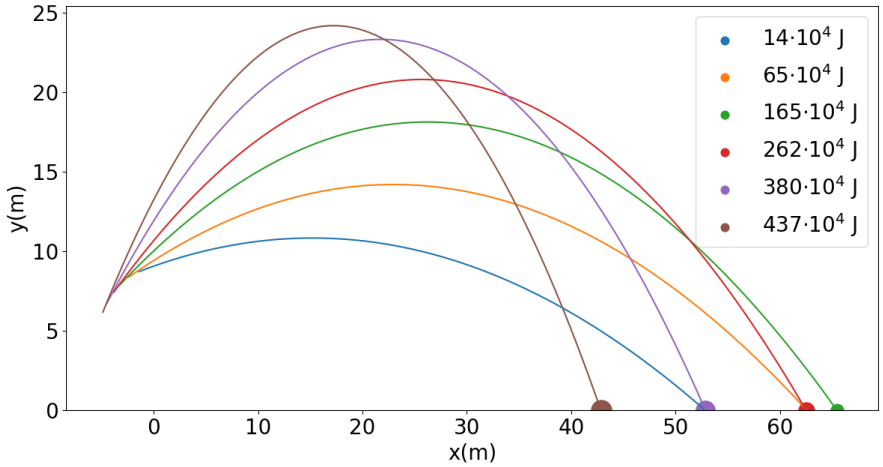


Figure 10: In this plot we see the path of the projectile after firing with the one angle trebuchet and different release angles θ_R. Besides the range, we can see the impact on the ground in the legend. A bigger circle at $y = 0$ means a bigger impact.

Now we compute the optimal θ_R for range and impact.

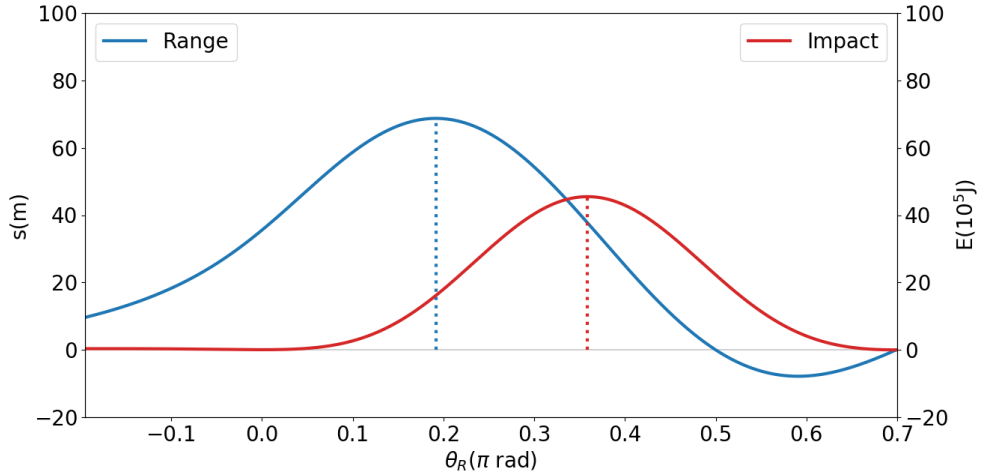


Figure 11: In this plot we see the range and impact of the one angle trebuchet with different release angles θ_R. The optimal range of 68.7 m is reached with release angle $\theta_R = 0.19\pi$ rad and the optimal impact of $45.5 \cdot 10^5$ J is reached with release angle $\theta_R = 0.36\pi$ rad.

2.5 Observations

The optimal release angle for range and impact are very different. Depending on what your goal is, we can adjust the model to optimize that. With the one angle trebuchet we can shoot up to 68.6 meters and have a maximum impact of $45.5 \cdot 10^5 J$.

3 The two angle model

For the second model we take our general problem with l_5 equal to zero. Only the angles θ and ϕ play a role in this model, so we call it the two angle model. This is shown schematically in figure 12. For the next model we will add l_5.This means that ψ plays a role again, so this is the three angle model.

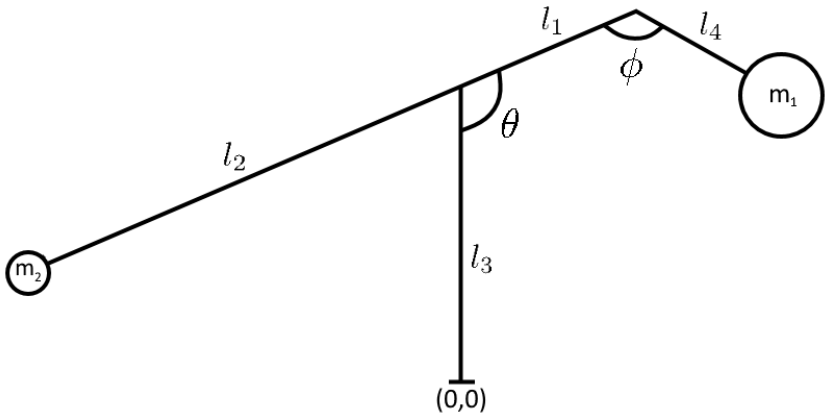


Figure 12: Schematic representation of the two angle model

3.1 Mathematical model

The position of the counterweight is given by r_{1b}, see figure 13.

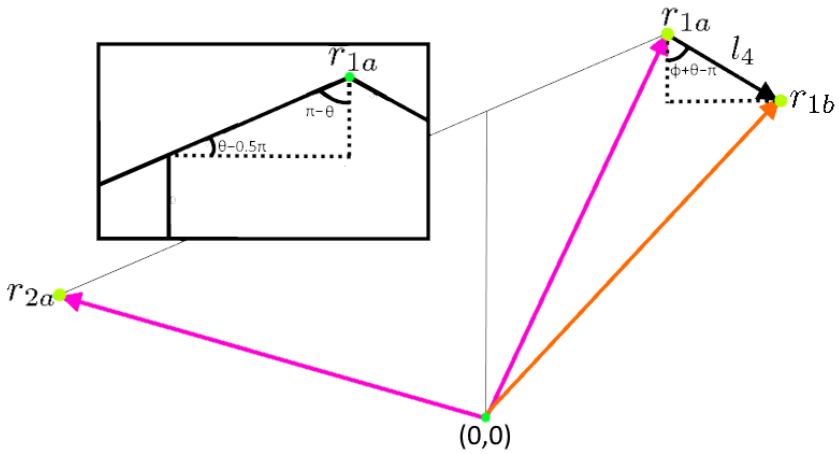


Figure 13: Vector representation of the two angle model

From figure 13 we observe

$$r_{1b} = r_{1a} + \begin{bmatrix} l_4 \sin(\theta + \phi - \pi) \\ -l_4 \cos(\theta + \phi - \pi) \end{bmatrix} = \begin{bmatrix} l_1 \sin(\theta) - l_4 \sin(\theta + \phi) \\ -l_1 \cos(\theta) + l_3 + l_4 \cos(\theta + \phi) \end{bmatrix} \tag{23}$$

where r_{1a} is the position of the projectile from the one angle model, see equation (1). The position of the projectile is denoted by r_{2b} and is not changed with respect to r_{2a} of the one angle model, see equation (2).

$$r_{2b} = r_{2a} = \begin{bmatrix} -l_2 \sin(\theta) \\ l_2 \cos(\theta) + l_3 \end{bmatrix}$$

We have a new equation for the height of the counterweight, so our potential energy of the counterweight changes as well.

$$E_{p1b} = gm_1(-l_1\cos(\theta) + l_4\cos(\theta+\phi) + l_3) \tag{24}$$

Since r_{2b} is the same as r_{2a}, the potential energy of the projectile is the same as that of the one angle model, $E_{p2b} = E_{p2a} = gm_2(l_2\cos(\theta) + l_3)$, see equation (3). Adding E_{p1b} and E_{p2b} yields the total potential energy

$$E_{pb} = -g(m_1l_1 - m_2l_2)\cos(\theta) + gm_1l_4\cos(\theta+\phi) + gl_3(m_1+m_2) \tag{25}$$

The equation for the position of the counterweight has changed, so the speed of the counterweight changes as well.

$$v_{1b}^2 = l_1^2\dot{\theta}^2 + l_4^2(\dot{\theta}+\dot{\phi})^2 - 2l_1l_4\dot{\theta}(\dot{\theta}+\dot{\phi})\cos(\phi)$$

Hence the kinetic energy of the counterweight becomes

$$E_{k1b} = \frac{1}{2}m_1(l_1^2\dot{\theta}^2 + l_4^2(\dot{\theta}+\dot{\phi})^2 - 2l_1l_4\dot{\theta}(\dot{\theta}+\dot{\phi})\cos(\phi)) \tag{26}$$

Since r_{2b} did not change with respect to r_{2a}, the speed of the counterweight will stay the same as the one angle model, $E_{k2b} = E_{k2a} = \frac{1}{2}m_2l_2^2\dot{\theta}^2$, see equation (5). Adding E_{k1b} and E_{k2b} equals the total kinetic energy

$$E_{kb} = \frac{1}{2}(m_1l_1^2 + m_2l_2^2)\dot{\theta}^2 + \frac{1}{2}m_1l_4^2(\dot{\theta}+\dot{\phi})^2 - m_1l_1l_4\dot{\theta}(\dot{\theta}+\dot{\phi})\cos(\phi) \tag{27}$$

This means that we find our Lagrangian, $L_b = E_{kb} - E_{pb}$, by using equation (25) and (27).

$$L_b = \frac{1}{2}(m_1l_1^2 + m_2l_2^2)\dot{\theta}^2 + \frac{1}{2}m_1l_4^2(\dot{\theta}+\dot{\phi})^2 - m_1l_1l_4\dot{\theta}(\dot{\theta}+\dot{\phi})\cos(\phi) \tag{28}$$

$$+ g(m_1l_1 - m_2l_2)\cos(\theta) - gm_1l_4\cos(\theta+\phi) - gl_3(m_1+m_2) \tag{29}$$

In order to use the Euler-Lagrange equation for θ, $\frac{\partial L_b}{\partial\theta} - \frac{\partial}{\partial t}\left(\frac{\partial L_b}{\partial\dot{\theta}}\right) = 0$, we need to compute $\frac{\partial L_b}{\partial\theta}$ and $\frac{\partial}{\partial t}\left(\frac{\partial L_b}{\partial\dot{\theta}}\right)$. By differentiating the Lagrangian, equation (29), with respect to θ we find

$$\frac{\partial L_b}{\partial\theta} = -g(m_1l_1 - m_2l_2)\sin(\theta) + gm_1l_4\sin(\theta+\phi) \tag{30}$$

Differentiating the Lagrangian, equation (29), with respect to $\dot{\theta}$ yields

$$\frac{\partial L_c}{\partial\dot{\theta}} = (m_1l_1^2 + m_2l_2^2)\dot{\theta} + m_1l_4^2(\dot{\theta}+\dot{\phi}) - m_1l_1l_4\cos(\phi)(2\dot{\theta}+\dot{\phi}) \tag{31}$$

Differentiating equation (31) with respect to t, gives us

$$\frac{\partial}{\partial t}\left(\frac{\partial L_b}{\partial\dot{\theta}}\right) = (m_1l_1^2 + m_2l_2^2)\ddot{\theta} + m_1l_4^2(\ddot{\phi}+\ddot{\theta}) + m_1l_1l_4[\sin(\phi)(2\dot{\theta}+\dot{\phi})\dot{\phi} - \cos(\phi)(2\ddot{\theta}+\ddot{\phi})] \tag{32}$$

By subtracting (32) from (30), we finally obtain

$$b_{11}\ddot{\theta} + b_{12}\ddot{\phi} - v_1 = 0 \tag{33}$$

where

$$
\begin{aligned}
b_{11}(\phi) &= -l_1^2 m_1 + 2 l_1 l_4 m_1 \cos(\phi) - l_2^2 m_2 - l_4^2 m_1 \\
b_{12}(\phi) &= l_4 m_1 (l_1 \cos(\phi) - l_4) \\
v_1(\theta, \phi, \dot{\theta}, \dot{\phi}) &= -g(l_2 m_2 \sin(\theta) - m_1(l_1 \sin(\theta) - l_4 \sin(\phi + \theta))) + l_1 l_4 m_1 (\dot{\phi} + 2\dot{\theta}) \sin(\phi) \dot{\phi}
\end{aligned}
$$

We added the angle ϕ, which means that we need to use the Euler-Lagrange equation for ϕ as well, $\frac{\partial L_b}{\partial \phi} - \frac{\partial}{\partial t}\left(\frac{\partial L_b}{\partial \dot{\phi}}\right) = 0$. First we differentiate the Lagrangian, equation (29), with respect to ϕ.

$$
\frac{\partial L_b}{\partial \phi} = -m_1 l_1 l_4 \dot{\theta}(\dot{\theta} + \dot{\phi}) \sin(\phi) + g m_1 l_4 \sin(\theta + \phi) \tag{34}
$$

Differentiating equation (29) with respect to $\dot{\phi}$ gives us

$$
\frac{\partial L_b}{\partial \dot{\phi}} = m_1 l_4^2 (\dot{\theta} + \dot{\phi}) + m_1 l_1 l_4 \dot{\theta} \cos(\phi) \tag{35}
$$

Differentiating equation (35) with respect to t yields

$$
\frac{\partial}{\partial t}\left(\frac{\partial L_b}{\partial \dot{\phi}}\right) = m_1 l_4^2 (\ddot{\theta} + \ddot{\phi}) + m_1 l_1 l_4 \ddot{\theta} \cos(\phi) - m_1 l_1 l_4 \dot{\theta} \sin(\phi) \dot{\phi} \tag{36}
$$

When we subtract (36) from (34), we obtain

$$
b_{21} \ddot{\theta} + b_{22} \ddot{\phi} - v_2 = 0 \tag{37}
$$

where

$$
\begin{aligned}
b_{21}(\phi) &= l_4 m_1 (l_1 \cos(\phi) - l_4) \\
&= a_{12}(\phi) \\
b_{22} &= -l_4^2 m_1 \\
v_2(\theta, \phi, \dot{\theta}) &= -g m_1 l_4 \sin(\theta + \phi) - m_1 l_1 l_4 \dot{\theta}^2 \sin(\phi)
\end{aligned}
$$

We can rewrite equations (33) and (37) as a matrix equation

$$
B \begin{bmatrix} \ddot{\theta} \\ \ddot{\phi} \end{bmatrix} = \vec{v} \text{ with } B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \text{ and } \vec{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, \text{ in this case } \begin{bmatrix} b_{11}(\phi) & b_{12}(\phi) \\ b_{12}(\phi) & b_{22} \end{bmatrix} \begin{bmatrix} \ddot{\theta} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} v_1(\theta, \phi, \dot{\theta}, \dot{\phi}) \\ v_2(\theta, \phi, \dot{\theta}) \end{bmatrix} \tag{38}
$$

We can then calculate $\ddot{\theta}$ and $\ddot{\phi}$ with $[\ddot{\theta}\ \ddot{\phi}]^T = B^{-1}\vec{v}$ provided B is invertible. Note that masses m_1 and m_2 and lengths l_1, l_2 and l_4 are nonzero by assumption. The determinant of B becomes

$$
\det(B) = l_4^2 m_1 (l_1^2 m_1 \sin^2(\phi) + l_2^2 m_2) \neq 0
$$

which means that the inverse of B always exists.

3.2 Checking the Lagrangian

Since our model get a little more complicated and the literature makes a lot of calculation errors as well, we use some checks to see if our Lagrangian is correct. First we see if the energy stays constant in the system, second we look at the Lagrangian of a double pendulum. To see if the energy stays constant we use the Hamiltonian.

$$E = \frac{\partial L_b}{\partial \dot{\theta}}\dot{\theta} + \frac{\partial L_b}{\partial \dot{\phi}}\dot{\phi} - L_b$$

If we calculate the energy with every step, it stays indeed constant.

Since modeling a double pendulum is done quite often in the literature and is almost always the same, we can assume that the Lagrangian of the double pendulum is correct [Altic, 2008, 7]. We can translate the constants of the Lagrangian of a double pendulum to our constants.

$$\begin{aligned} L = &\frac{1}{2}(m_1 + m_3)l_1^2\dot{\theta}^2 + \frac{1}{2}m_1 l_4^2(\dot{\phi} + \dot{\theta})^2 - m_1 l_1 l_4 \dot{\theta}(\dot{\theta} + \dot{\phi})\cos(\phi) \\ &+ (m_1 + m_3)g l_1 \cos(\theta) - m_1 l_4 g \cos(\phi + \theta) \end{aligned}$$

m_3 is a mass at the ϕ-joint, we take $m_3 = 0$ to get our model. If we take $m_2 = 0$ and $l_2 = 0$ in our Lagrangian, we are left with the Lagrangian of a double pendulum with $m_3 = 0$.

$$\begin{aligned} L = &\frac{1}{2}m_1 l_1^2\dot{\theta}^2 + \frac{1}{2}m_1 l_4^2(\dot{\theta} + \dot{\phi})^2 - m_1 l_1 l_4 \dot{\theta}(\dot{\theta} + \dot{\phi})\cos(\phi) \\ &+ g m_1 l_1 \cos(\theta) - g m_1 l_4 \cos(\theta + \phi) - g l_3 m_1 \end{aligned}$$

The only difference is the last part with the of the Lagrangian. Since this part does not have any relation to θ or ϕ, we can ignore it. When we take the derivatives, that part will be eliminated. In other literature we see the same set of equations as (38) [Siano, 2013, 11].

3.3 Limit to the last model

When we take $l_4 = 0$ in our matrix B and vector $\vec{v}$, we get the same equation for $\ddot{\theta}$ from the one angle model, see equation (11). This is exactly what we would expect. However if we take the limit $l_4 \to 0$, we do not get the same result. This is because of singular perturbation. The same thing happens with a double pendulum, what looks a lot like our situation. With singular perturbation, we can not approximate what happens with a really small value for the parameter, l_4 in our case, by setting the parameter value to zero.

3.4 Numerical solution

To see if the time step is still small enough, we ran the model for different step sizes until we observed no difference. To get smooth plots with the two angle model, we need $\Delta t = 10^{-3}$.

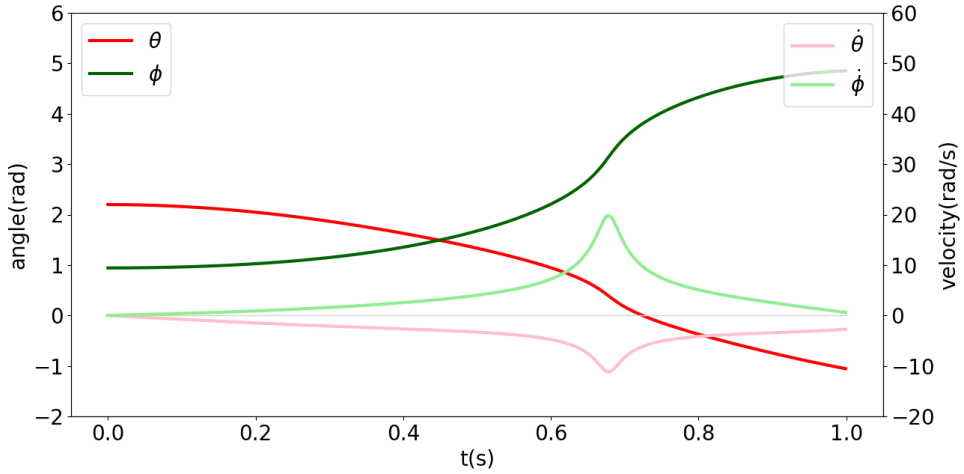


Figure 14: In this plot we see $\theta, \phi, \dot{\theta}$ and $\dot{\phi}$ as a function of time. We used the two angle model with $\Delta t = 10^{-3}$ and $T = 1$. We see $\dot{\theta}$ has a negative peak, which is the optimal moment to fire.

We can also see what happens if we look at a longer time period.

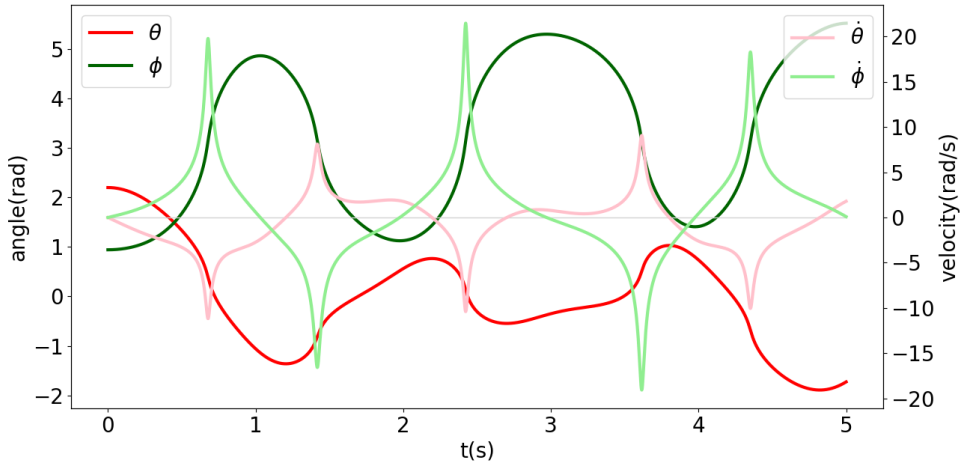


Figure 15: In this plot we see $\theta, \phi, \dot{\theta}$ and $\dot{\phi}$ as a function of time. We used the two angle model with $\Delta t = 10^{-3}$ and $T = 5$. We see that the solution is not periodic. This is to be expected since our two angle model is very similar to a double pendulum and a double pendulum is known to be chaotic. Note that the high peaks are not singularities, see figure 14.

3.5 Firing

Since the equations for the position of the projectile r_{2b} is the same as that of the one angle model r_{2a}, see equation (2), we can use the same equations for firing, see section 2.4. The position of the counterweight has changed, so the speed of the projectile at different release angles θ_R has changed.

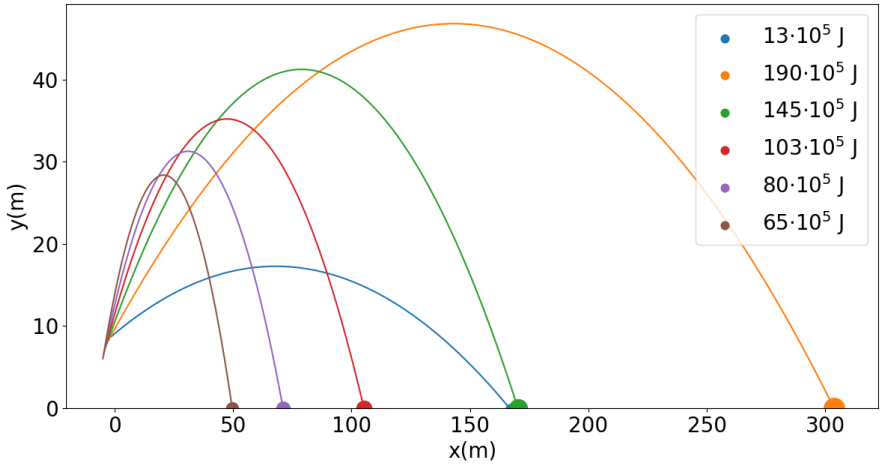


Figure 16: In this plot we see the path of the projectile after firing with the two angle trebuchet and different release angles θ_R. Besides the range, we can see the impact on the ground in the legend. A bigger circle at $y = 0$ means a bigger impact.

Now we compute the optimal θ_R for range and impact.

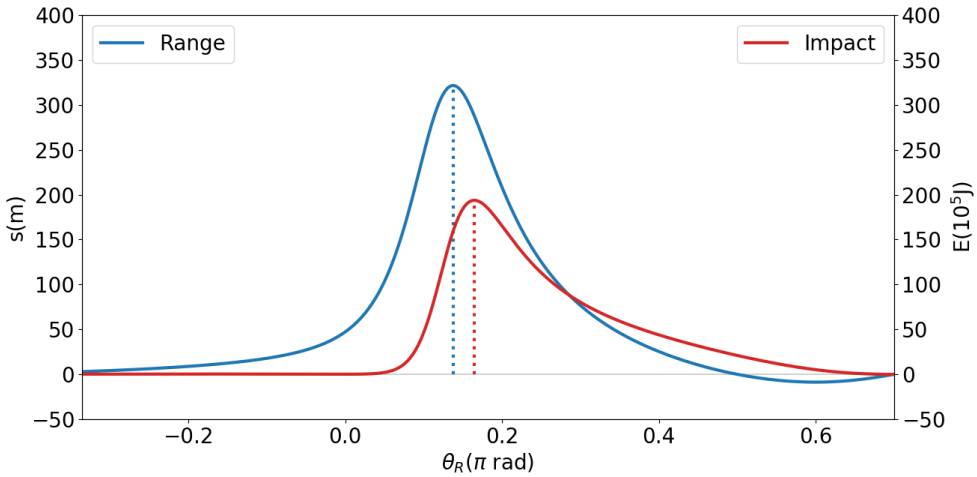


Figure 17: In this plot we see the range and impact of the two angle trebuchet with different release angles θ_R. The optimal range of 321.6 m is reached with release angle $\theta_R = 0.14\pi$ rad and the optimal impact of $19.4 \cdot 10^6$ J is reached with release angle $\theta_R = 0.16\pi$ rad.

3.6 Observations

The optimal release angle for range and impact are closer together than with the one angle model. With the two angle trebuchet we can shoot up to 321.6 meters, which is 4.7 times the range of the one angle trebuchet. We have a maximum impact of $19.4 \cdot 10^6 J$, which is 4.3 times the one angle trebuchet. This is because the angular velocities have extreme peaks.

4 The three angle model

For the third model we take our general problem, see section 1.5. All three angle play a role in this model, so we call it the three angle model. This is shown schematically in figure 18.

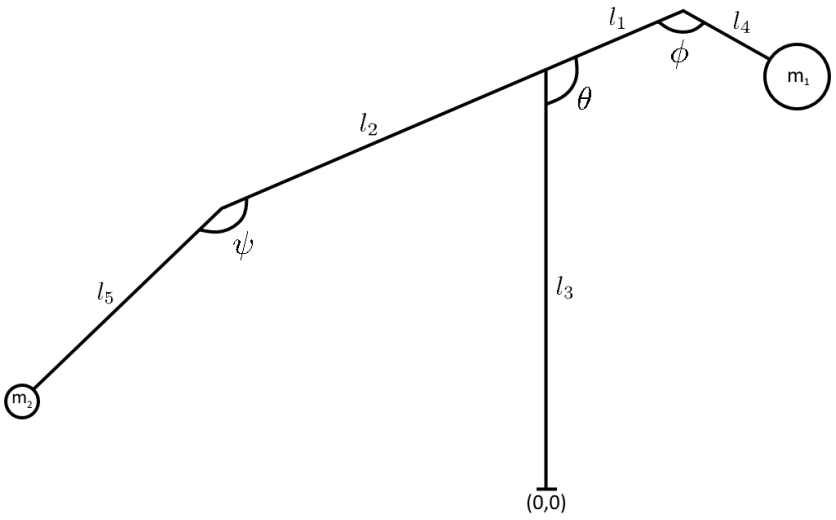


Figure 18: Schematic representation of the three angle model

4.1 Mathematical model

The position of the projectile is given by r_{2c}, see figure 19.

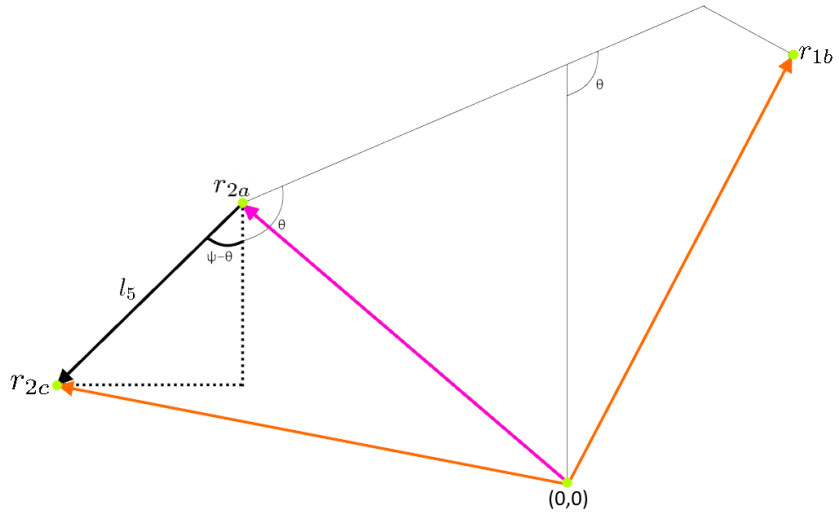


Figure 19: Vector representation of the three angle model

From figure 19 we observe

$$r_{2c} = r_{2a} - \begin{bmatrix} l_5 \sin(\psi - \theta) \\ l_5 \cos(\psi - \theta) \end{bmatrix} = \begin{bmatrix} -l_2 \sin(\theta) - l_5 \sin(\psi - \theta) \\ l_2 \cos(\theta) + l_3 - l_5 \cos(\psi - \theta) \end{bmatrix} \tag{39}$$

where r_{2a} is the position of the projectile from the one angle model, see equation (2). The position of the counterweight is denoted by r_{1c} and is not changed with respect to r_{1b} of the two angle model, see equation (23).

$$r_{1c} = r_{1b} = \begin{bmatrix} l_1 \sin(\theta) - l_4 \sin(\theta + \phi) \\ -l_1 \cos(\theta) + l_3 + l_4 \cos(\theta + \phi) \end{bmatrix}$$

We have a new equation for the height of the projectile, so our potential energy of the projectile changes as well. Filling in the new height yields

$$E_{p2c} = gm_2(l_2 \cos(\theta) + l_3 - l_5 \cos(\psi - \theta))$$

Since r_{1c} is the same as r_{1b}, the potential energy of the counterweight is the same as that of the two angle model, $E_{p1c} = E_{p1b} = gm_1(-l_1 \cos(\theta) + l_4 \cos(\theta + \phi) + l_3)$, see equation (24). Adding E_{p1c} and E_{p2c} yields the total potential energy

$$E_{pc} = -g(m_1 l_1 - m_2 l_2) \cos(\theta) + gl_3(m_1 + m_2) + gm_1 l_4 \cos(\theta + \phi) - gm_2 l_5 \cos(\psi - \theta) \tag{40}$$

The equation for the position of the projectile has changed, so the speed of the projectile changes as well. The new speed of the projectile is given by

$$v_{2c}^2 = l_2^2 \dot{\theta}^2 + l_5^2(\dot{\psi} - \dot{\theta})^2 + 2l_2 l_5 \dot{\theta}(\dot{\psi} - \dot{\theta}) cos(\psi)$$

Hence the kinetic energy of the projectile becomes

$$E_{k2c} = \frac{1}{2} m_2 (l_2^2 \dot{\theta}^2 + l_5^2(\dot{\psi} - \dot{\theta})^2 + 2l_2 l_5 \dot{\theta}(\dot{\psi} - \dot{\theta}) cos(\psi))$$

Since r_{1c} did not change with respect to r_{1b}, the speed of the counterweight will stay the same as the two angle model, $E_{k1c} = E_{k1b} = \frac{1}{2} m_1 (l_1^2 \dot{\theta}^2 + l_4^2(\dot{\theta} + \dot{\phi})^2 - 2l_1 l_4 \dot{\theta}(\dot{\theta} + \dot{\phi}) \cos(\phi))$, see equation (26). Adding E_{k1c} and E_{k2c} yields the total kinetic energy

$$\begin{aligned} E_{kc} = {} & \frac{1}{2} m_1 l_1^2 \dot{\theta}^2 + \frac{1}{2} m_1 l_4^2 (\dot{\theta} + \dot{\phi})^2 - m_1 l_1 l_4 \dot{\theta} \cos(\phi)(\dot{\phi} + \dot{\theta}) \\ & + \frac{1}{2} m_2 l_2^2 \dot{\theta}^2 + \frac{1}{2} m_2 l_5^2 (\dot{\psi} - \dot{\theta})^2 + m_2 l_2 l_5 \dot{\theta} \cos(\psi)(\dot{\psi} - \dot{\theta}) \end{aligned} \tag{41}$$

This means that we find our Lagrangian, $L_c = E_{kc} - E_{pc}$, by using equation (40) and (41).

$$\begin{aligned} L_c = {} & \frac{1}{2}(m_1 l_1^2 + m_2 l_2^2)\dot{\theta}^2 - gl_3(m_1 + m_2) + g(m_1 l_1 - m_2 l_2) \cos(\theta) \\ & + \frac{1}{2} m_1 l_4^2 (\dot{\theta} + \dot{\phi})^2 - m_1 l_1 l_4 \dot{\theta}(\dot{\theta} + \dot{\phi}) \cos(\phi) - gm_1 l_4 \cos(\theta + \phi) \\ & + \frac{1}{2} m_2 l_5^2 (\dot{\psi} - \dot{\theta})^2 + m_2 l_2 l_5 \dot{\theta}(\dot{\psi} - \dot{\theta}) \cos(\psi) + gm_2 l_5 \cos(\psi - \theta) \end{aligned} \tag{42}$$

In order to use the Euler-Lagrange equation for θ, $\frac{\partial L_c}{\partial \theta} - \frac{\partial}{\partial t}\left(\frac{\partial L_c}{\partial \dot{\theta}}\right) = 0$, we need to compute $\frac{\partial L_c}{\partial \theta}$ and $\frac{\partial}{\partial t}\left(\frac{\partial L_c}{\partial \dot{\theta}}\right)$. By differentiating the Lagrangian, equation (42), with respect to θ we find

$$\frac{\partial L_c}{\partial \theta} = -g(m_1 l_1 - m_2 l_2) \sin(\theta) + gm_1 l_4 \sin(\theta + \phi) + gm_2 l_5 \sin(\psi - \theta) \tag{43}$$

Differentiating the Lagrangian, equation (42), with respect to $\dot{\theta}$ yields

$$\begin{aligned} \frac{\partial L_c}{\partial \dot{\theta}} = {} & (m_1 l_1^2 + m_2 l_2^2)\dot{\theta} \\ & + m_1 l_4^2 (\dot{\theta} + \dot{\phi}) - m_1 l_1 l_4 \cos(\phi)(2\dot{\theta} + \dot{\phi}) \\ & - m_2 l_5^2 (\dot{\psi} - \dot{\theta}) - m_2 l_2 l_5 \cos(\psi)(2\dot{\theta} - \dot{\psi}) \end{aligned} \tag{44}$$

Differentiating equation (44) with respect to t, gives us

$$\begin{aligned}\frac{\partial}{\partial t}\Big(\frac{\partial L_c}{\partial \dot{\theta}}\Big) &= (m_1 l_1^2 + m_2 l_2^2)\ddot{\theta} \\ &+ m_1 l_4^2(\ddot{\theta}+\ddot{\phi}) + m_1 l_1 l_4\Big[\sin(\phi)\dot{\phi}(2\dot{\theta}+\dot{\phi}) - \cos(\phi)(2\ddot{\theta}+\ddot{\phi})\Big] \\ &- m_2 l_5^2(\ddot{\psi}-\ddot{\theta}) + m_2 l_2 l_5\Big[\sin(\psi)\dot{\psi}(2\dot{\theta}-\dot{\psi}) - \cos(\psi)(2\ddot{\theta}-\ddot{\psi})\Big]\end{aligned} \tag{45}$$

By subtracting (45) from (43), we finally obtain

$$c_{11}\ddot{\theta} + c_{12}\ddot{\phi} + c_{13}\ddot{\psi} - w_1 = 0 \tag{46}$$

where

$$\begin{aligned}c_{11}(\phi,\psi) &= -l_1^2 m_1 + 2l_1 l_4 m_1\cos(\phi) - l_4^2 m_1 - l_2^2 m_2 + 2l_2 l_5 m_2\cos(\psi) - l_5^2 m_2 \\ c_{12}(\phi) &= l_4 m_1(l_1\cos(\phi) - l_4) \\ c_{13}(\psi) &= -l_5 m_2(l_2\cos(\psi) - l_5) \\ w_1(\theta,\phi,\psi,\dot{\theta},\dot{\phi},\dot{\psi}) &= g(m_1(l_1\sin(\theta) - l_4\sin(\phi+\theta)) - m_2(l_2\sin(\theta) + l_5\sin(\psi-\theta))) \\ &+ l_1 l_4 m_1(\dot{\phi}+2\dot{\theta})\sin(\phi)\dot{\phi} + l_2 l_5 m_2(-\dot{\psi}+2\dot{\theta})\sin(\psi)\dot{\psi}\end{aligned}$$

When we compare the Lagrangian of the two angle model, see equation (29), with the Lagrangian of the three angle model, see equation (42), we see that there are no terms with ϕ changed or added. Hence the Euler-Lagrange expression for ϕ stays the same, see equation (37).

$$c_{21}\ddot{\theta} + c_{22}\ddot{\phi} + c_{23}\ddot{\psi} = w_2 \tag{47}$$

where

$$\begin{aligned}c_{21}(\phi) = b_{21} &= l_4 m_1(l_1\cos(\phi) - l_4) \\ &= c_{12}(\phi) \\ c_{22} = b_{22} &= -l_4^2 m_1 \\ c_{23} &= 0 \\ w_2(\theta,\phi,\dot{\theta}) = v_2 &= l_4 m_1(g\sin(\phi-\theta) - l_1\sin(\psi)\dot{\theta}^2)\end{aligned}$$

We added the angle ψ, which means that we need to use the Euler-Lagrange equation for ψ as well, $\frac{\partial L_c}{\partial \psi} - \frac{\partial}{\partial t}\Big(\frac{\partial L_c}{\partial \dot{\psi}}\Big) = 0$. First we differentiate the Lagrangian, equation (42), with respect to ψ.

$$\frac{\partial L_c}{\partial \psi} = -g m_2 l_5\sin(\psi-\theta) - m_2 l_2 l_5\dot{\theta}\sin(\psi)(\dot{\psi}-\dot{\theta}) \tag{48}$$

Differentiating equation (42) with respect to $\dot{\psi}$ gives us

$$\frac{\partial L_c}{\partial \dot{\psi}} = m_2 l_5^2(\dot{\psi}-\dot{\theta}) + m_2 l_2 l_5\dot{\theta}\cos(\psi) \tag{49}$$

Differentiating equation (49) with respect to t yields

$$\frac{\partial}{\partial t}\Big(\frac{\partial L_c}{\partial \dot{\psi}}\Big) = m_2 l_5^2(\ddot{\psi}-\ddot{\theta}) + m_2 l_2 l_5(-\dot{\theta}\sin(\psi)\dot{\psi} + \ddot{\theta}\cos(\psi)) \tag{50}$$

When we subtract (50) from (49), we obtain

$$c_{31}\ddot{\theta} + c_{32}\ddot{\phi} + c_{33}\ddot{\psi} - w_3 = 0 \tag{51}$$

where

$$\begin{aligned} c_{31}(\psi) &= l_5 m_2(-l_2\cos(\psi) + l_5) \\ &= c_{13}(\psi) \\ c_{32} &= 0 \\ c_{33} &= -l_5^2 m_2 \\ w_3(\theta, \psi, \dot{\theta}) &= l_5 m_2(g\sin(\psi - \theta) - l_2\sin(\psi)\dot{\theta}^2) \end{aligned}$$

Combining equations (46), (47) and (51) and rewriting them as a matrix equation reduces the problem to

$$\begin{bmatrix} c_{11}(\phi,\psi) & c_{12}(\phi) & c_{13}(\psi) \\ c_{12}(\phi) & c_{22} & 0 \\ c_{13}(\psi) & 0 & c_{33} \end{bmatrix} \begin{bmatrix} \ddot{\theta} \\ \ddot{\phi} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} w_1(\theta,\phi,\psi,\dot{\theta},\dot{\phi},\dot{\psi}) \\ w_2(\theta,\phi,\dot{\theta}) \\ w_3(\theta,\psi,\dot{\theta}) \end{bmatrix}$$

where we can find our double derivatives by inverting the matrix C. To make sure that this is always possible, we check that the determinant of C is nonzero. Assume that l_1, l_2, l_4, l_5, m_1 and m_2 are nonzero, then the determinant of C becomes

$$\det(C) = -l_4^2 l_5^2 m_1 m_2 (l_1^2 m_1 \sin^2(\phi) + l_2^2 m_2 \sin^2(\psi))$$

We see that we get a problem in the special case where $\sin(\phi)$ and $\sin(\psi)$ are zero at the exact same moment. We build in a check to tell us if $l_1^2 m_1 \sin^2(\phi) + l_2^2 m_2 \sin^2(\psi) < 0.1$. Since this never happens, we can assume that no problems occur with inverting the matrix C.

4.2 Checking the Lagrangian

Our model gets even more complicated. So we again use some checks to see if our Lagrangian is correct. First we see if the energy stays constant in the system, second we compare our Lagrangian to the literature. To see if the energy stays constant we use the Hamiltonian.

$$E = \frac{\partial L_c}{\partial \dot{\theta}}\dot{\theta} + \frac{\partial L_c}{\partial \dot{\phi}}\dot{\phi} + \frac{\partial L_c}{\partial \dot{\psi}}\dot{\psi} - L_c$$

If we calculate the energy with every step, it stays indeed constant.
In the literature we see the same Lagrangian [Rutan and Wieczorek, 2005, 27]. We also compare our Lagrangian to that of a simulation [Mahieu and Brandhuber, 2012]. We use this source because it is a blog like post, were a lot of people interested in this field comment. So the chances for errors gone unnoticed are very slim. The Lagrangian used in the simulation is

$$\begin{aligned} L = \frac{1}{2}\Big[& 2g\Big(cos(\theta_w)(L_1M_1 - L_2M_2) + L_3M_2cos(\phi_w) + L_4M_1cos(\psi_w)\Big) + \dot{\theta}_w^2(L_1^2M_1 + L_2^2M_2) \\ & + \dot{\theta}_w\Big(2L_1L_4M_1\dot{\psi}_w cos(\theta_w - \psi_w) - 2L_2L_3M_2\dot{\phi}_w cos(\theta_w - \phi_w)\Big) + L_3^2M_2\dot{\phi}_w^2 + L_4^2M_1\dot{\psi}_w^2\Big] \end{aligned}$$

We can translate the constants to our constants with $L_1 = l_2, L_2 = l_1, L_3 = l_4, L_4 = l_5, M_1 = m_2, M_2 = m_1, \theta_w = \theta + \pi, \phi_w = \theta + \phi + \pi$ and $\psi_w = \theta - \psi$.

$$\begin{aligned} L = \frac{1}{2}\Big[& 2g\Big(cos(\theta + \pi)(l_2m_2 - l_1m_1) + l_4m_1cos(\theta + \phi + \pi) + l_5m_2cos(\theta - \psi)\Big) + \dot{\theta}^2(l_2^2m_2 + l_1^2m_1) \\ & + \dot{\theta}\Big(2l_2l_5m_2(\dot{\theta} - \dot{\psi})cos(\psi + \pi) - 2l_1l_4m_1(\dot{\theta} + \dot{\phi})cos(-\phi)\Big) + l_4^2m_1(\dot{\theta} + \dot{\phi})^2 + l_5^2m_2(\dot{\theta} - \dot{\psi})^2\Big] \end{aligned}$$

This is equal to our Lagrangian without the constant part, see equation (42). This is no problem, since the constant part will be eliminated when we differentiate.

$$
\begin{aligned}
L_c = &\frac{1}{2}(m_1 l_1^2 + m_2 l_2^2)\dot{\theta}^2 + g(m_1 l_1 - m_2 l_2)\cos(\theta) \\
&+\frac{1}{2} m_1 l_4^2 (\dot{\theta} + \dot{\phi})^2 - m_1 l_1 l_4 \dot{\theta}(\dot{\theta} + \dot{\phi})\cos(\phi) - g m_1 l_4 \cos(\theta + \phi) \\
&+\frac{1}{2} m_2 l_5^2 (\dot{\psi} - \dot{\theta})^2 + m_2 l_2 l_5 \dot{\theta}(\dot{\psi} - \dot{\theta})\cos(\psi) + g m_2 l_5 \cos(\psi - \theta)
\end{aligned}
$$

4.3 Limit to the last model

When we take $l_5 = 0$ matrix C and vector $\vec{w}$ we get the matrix B and vector $\vec{v}$ from the two angle model, see equation (38). However the same thing goes wrong when we take the limit $l_5 \to 0$, see section 3.3.

4.4 Numerical solution

To see if the time step is still small enough, we ran the model for a smaller time step $\Delta t = 10^{-4}$. We observed no difference, so time step size $\Delta t = 10^{-3}$ is small enough for the three angle model.

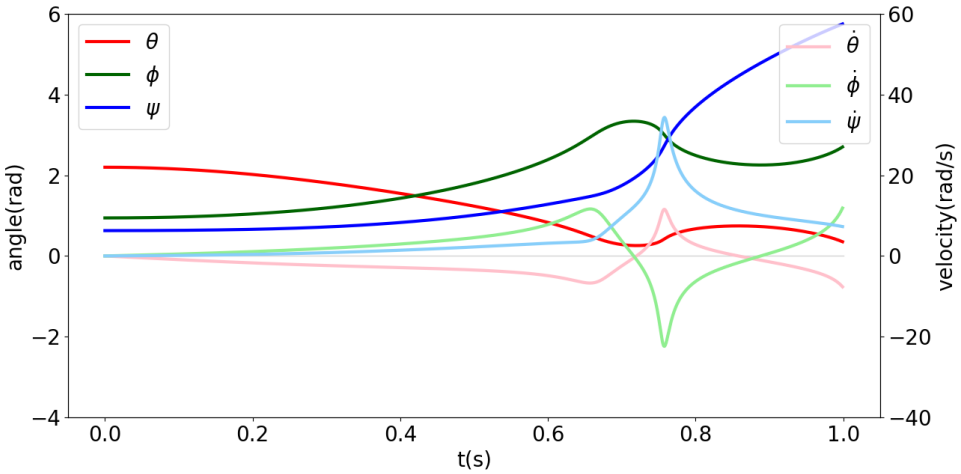


Figure 20: In this plot we see $\theta, \phi, \psi, \dot{\theta}, \dot{\phi}$ and $\dot{\psi}$ as a function of time. We used the three angle model with $\Delta t = 10^{-3}$ and $T = 1$. To fire as far as possible, a most negative $\dot{\theta}$ and most positive $\dot{\psi}$ are ideal. We see that $\dot{\theta}$ does not really have a negative peak this time, but $\dot{\psi}$ does have a big positive peak.

We can also see what happens if we look at a longer time period.

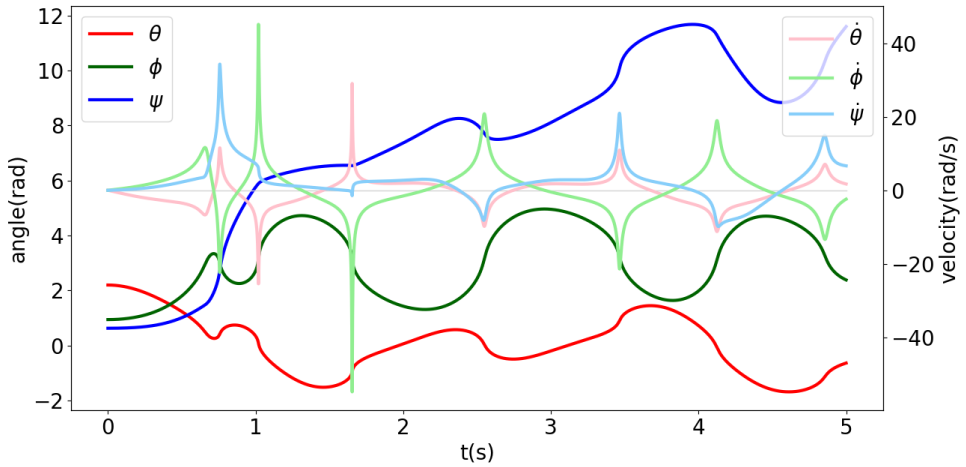


Figure 21: In this plot we see $\theta, \phi, \psi, \dot{\theta}, \dot{\psi}$ and $\dot{\phi}$ as a function of time. We used the three angle model with $\Delta t = 10^{-3}$ and $T = 5$. We see that the solution is not periodic. This is to be expected since our three angle model is even more complicated than the two angle model. Note that the high peaks are not singularities, see figure 20.

4.5 Firing

For $t \leq t_R$ the position of the projectile r_{2c} has changed, see equation (39).

$$r_{2c}(t) = \begin{bmatrix} -l_2 \sin(\theta) - l_5 \sin(\psi - \theta) \\ l_2 \cos(\theta) + l_3 - l_5 \cos(\psi - \theta) \end{bmatrix} \tag{52}$$

We can differentiate r_{2c} with respect to t.

$$\dot{r}_{2c}(t) = \begin{bmatrix} -l_2 \cos(\theta_R)\dot{\theta}_R - l_5 \cos(\psi_R - \theta_R)(\dot{\psi}_R - \dot{\theta}_R) \\ -l_2 \sin(\theta_R)\dot{\theta}_R + l_5 \sin(\psi_R - \theta_R)(\dot{\psi}_R - \dot{\theta}_R) \end{bmatrix} \tag{53}$$

For $t \geq t_R$ the equations stay the same, see (18) and (19).

$$\dot{r}_2(t) = \begin{bmatrix} \beta \\ -g \cdot (t - t_R) + \gamma \end{bmatrix} \qquad r_2(t) = \begin{bmatrix} \beta \cdot (t - t_R) + \delta \\ -\frac{1}{2} g \cdot (t - t_R)^2 + \gamma \cdot (t - t_R) + \epsilon \end{bmatrix} \tag{54}$$

At $t = t_R$ we know that the speed and position of the projectile should be the same. From $\dot{r}_{2c}(t_R) = \dot{r}_2(t_R)$ follows

$$\beta(\theta_R) = -l_2 \cos(\theta_R)\dot{\theta}_R - l_5 \cos(\psi_R - \theta_R)(\dot{\psi}_R - \dot{\theta}_R)$$
$$\gamma(\theta_R) = -l_2 \sin(\theta_R)\dot{\theta}_R + l_5 \sin(\psi_R - \theta_R)(\dot{\psi}_R - \dot{\theta}_R)$$

From $r_{2c}(t_R) = r_2(t_R)$ follows

$$\delta(\theta_R) = -l_2 \sin(\theta_R) - l_5 \sin(\psi_R - \theta_R)$$
$$\epsilon(\theta_R) = l_2 \cos(\theta_R) + l_3 - l_5 \cos(\psi_R - \theta_R)$$

The equations for $t_E(\theta_R), s(\theta_R)$ and $E(\theta_R)$ stay the same. Note that $\beta(\theta_R), \gamma(\theta_R)$ and $\epsilon(\theta_R)$ did change.

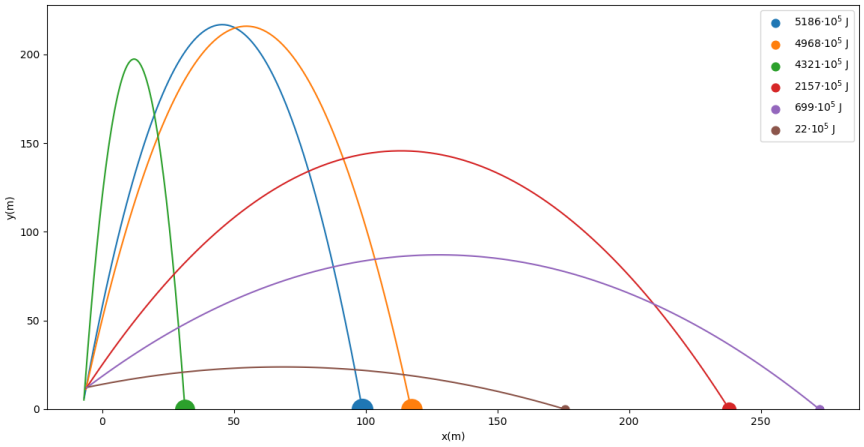


Figure 22: In this plot we see the path of the projectile after firing with the three angle trebuchet and different release angles θ_R. Besides the range, we can see the impact on the ground in the legend. A bigger circle at $y = 0$ means a bigger impact.

Now we compute the optimal θ_R for range and impact.

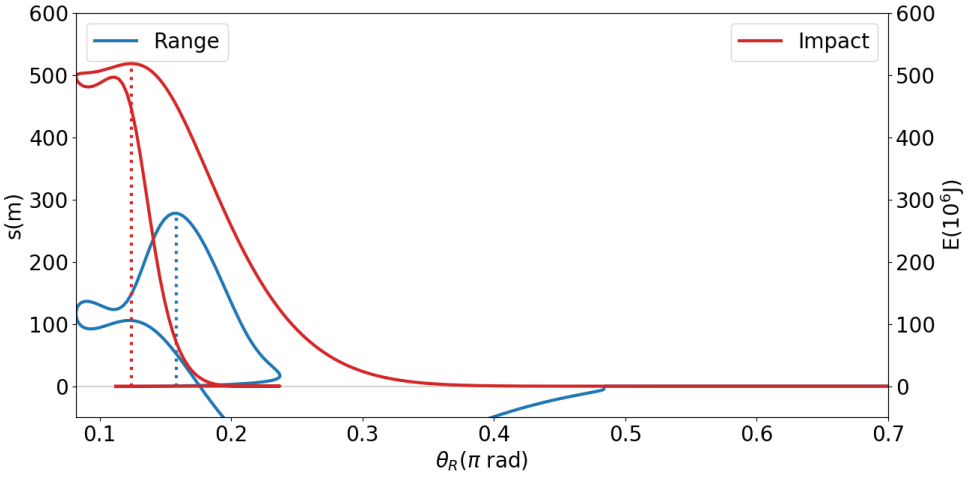


Figure 23: In this plot we see the range and impact of the two angle trebuchet with different release angles θ_R. The optimal range of 278.1 m is reached with release angle $\theta_R = 0.16\pi$ rad and the optimal impact of $51.9 \cdot 10^6$ J is reached with release angle $\theta_R = 0.12\pi$ rad. Note that we can have a same possible release angle on different times. This is why we get multiple values at certain release angles. The negative blue line is not interesting for us, since we want to shoot forward.

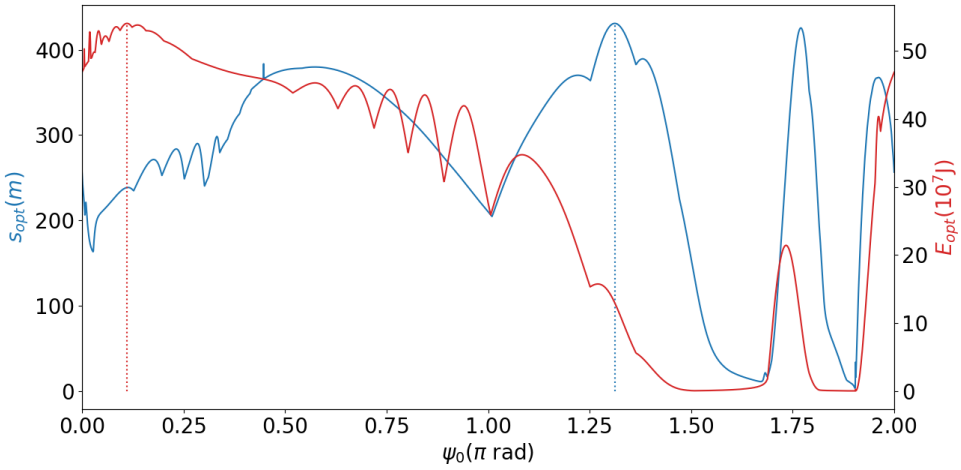


Figure 24: In this plot we see that the ψ_0 we choose can make a big difference in range and impact. We computed the optimum range and impact for 2500 different ψ_0. The optimum range of 431.0 m is reached with $\psi_0 = 1.31\pi$ and $\theta_R = 0.12\pi$. The optimum impact of $54.1 \cdot 10^7$ J is reached with $\psi_0 = 0.11\pi$ and $\theta_R = 0.12\pi$.

4.6 Observations

The optimum impact with the three angle trebuchet is $51.9 \cdot 10^6$ J, which is 2.7 times better compared to the two angle model. With the three angle trebuchet we can shoot up to 278.1 meters, which is a little less than the two angle trebuchet. However we have seen in figure 24 that how we choose our ψ_0 results in a very different range and impact. When we choose a optimal ψ_0 we can shoot further than the two angle trebuchet.

5 Conclusion and discussion

With this project we wanted to model the movements of a counterweight trebuchet while firing. We start with an oversimplified model and made the model more realistic in consecutive steps. We started with the one angle model, which has only angle θ, the angle between the main beam and the stand. With the Euler-Lagrange method, we derived the equation of motion for θ. We got a periodic solution for θ that keeps oscillating with the same amplitude. This is what we expect, since we did not include friction in the model.

To get the two angle model, we added the angle ϕ to, the angle between the main beam and the counterweight. The equations of motion got more complicated and we could not calculate an analytic solution anymore, we needed to do it numerically. Since our $\ddot{\phi}$ depends on $\ddot{\theta}$, we wrote our equations of motion as a matrix equation. Our solution was not periodic anymore and had extreme peaks. Because the equations of motion of our two angle model are similar to that of a double pendulum, which is known to exhibit chaotic behavior, we expected these results. The extreme peaks make it harder to reach close to the optimal range or impact in practice. Our optimal range and impact increased enormously from the one angle to the two angle trebuchet. We went from 68.7 m and $45.5 \cdot 10^5$ J to 321.6 m and $19.4 \cdot 10^6$ J. We did not expect that it would have a large effect, since the beam we added, l_4, is quite small in comparison. Even though l_4 is small, addition of this beam may still have a noticeable effect because l_4 is attached to the large counterweight.

Lastly we added the angle ψ to get our three angle model. The equations of motion got even more complicated. Our model reduces to a matrix equation $C\vec{x} = \vec{w}$, which means that we have to carefully check $\det(C) \neq 0$. With this model we could check our Lagrangian, but not the equations of motion anymore since they got to complicated. We saw that the starting angle ϕ_0 we chose, had a big effect on our optimal range and impact. When we choose the right ϕ_0, we can shoot harder and a little further than the two angle trebuchet, exact numbers are given in the caption of figure 24. We expected to be able to shoot a lot further with the three angle model, then with the two angle model. The sling makes the trebuchet a different kind of catapult, which is known to shoot further and harder than other kinds of catapults.

With our three angle trebuchet, we can shoot a lot further (431 m) than in practice where values around 165 m are reached. This is to be expected since we did not take friction or the beam weight into account. The optimal release angles for impact and range with our three angle model ($\theta_R = 0.12\pi$) are very close to what is used in practice ($\theta_R = 0.11\pi$). This could mean that even though we can not predict the trajectory of the projectile correctly, we can determine a close to optimal release angle. However we have also seen that small difference in choosing our ψ_0 has a big effect on our results. So our model is not yet sufficient enough to predict the movements of a counterweight trebuchet firing realistically.

In a follow up project, we would first take the ground into account. With the ground there is less mobility for ϕ, so hopefully this will solve the problem that choosing a certain ϕ_0 has an big effect on the results. The second step would be taking the beam weight into account, since it is a twelve ton structure. A lot of energy that now goes into the projectile might be lost there.

References

[Altic, 2008] Altic, J. (2008). *Double Pendulum*.

[Barger and Olsson, 1995] Barger, V. D. and Olsson, M. G. (1995). *CLASSICAL MECHANICS: A Modern Perspective*.

[Hansen, 1992] Hansen, D. P. V. (1992). *Experimental Reconstruction of a Medieval Trébuchet*.

[Mahieu and Brandhuber, 2012] Mahieu, E. and Brandhuber, F. (2012). Optimizing the counterweight trebuchet.

[Purton, 2009] Purton, P. (2009). *A History of the Early Medieval Siege, c.450-1200*.

[Rutan and Wieczorek, 2005] Rutan, S. and Wieczorek, B. (2005). *Modern Siege Weapons: Mechanics of the Trebuchet*.

[Siano, 2013] Siano, D. B. (2013). *Trebuchet Mechanics*.

A Appendix

A.1 Reconstruction

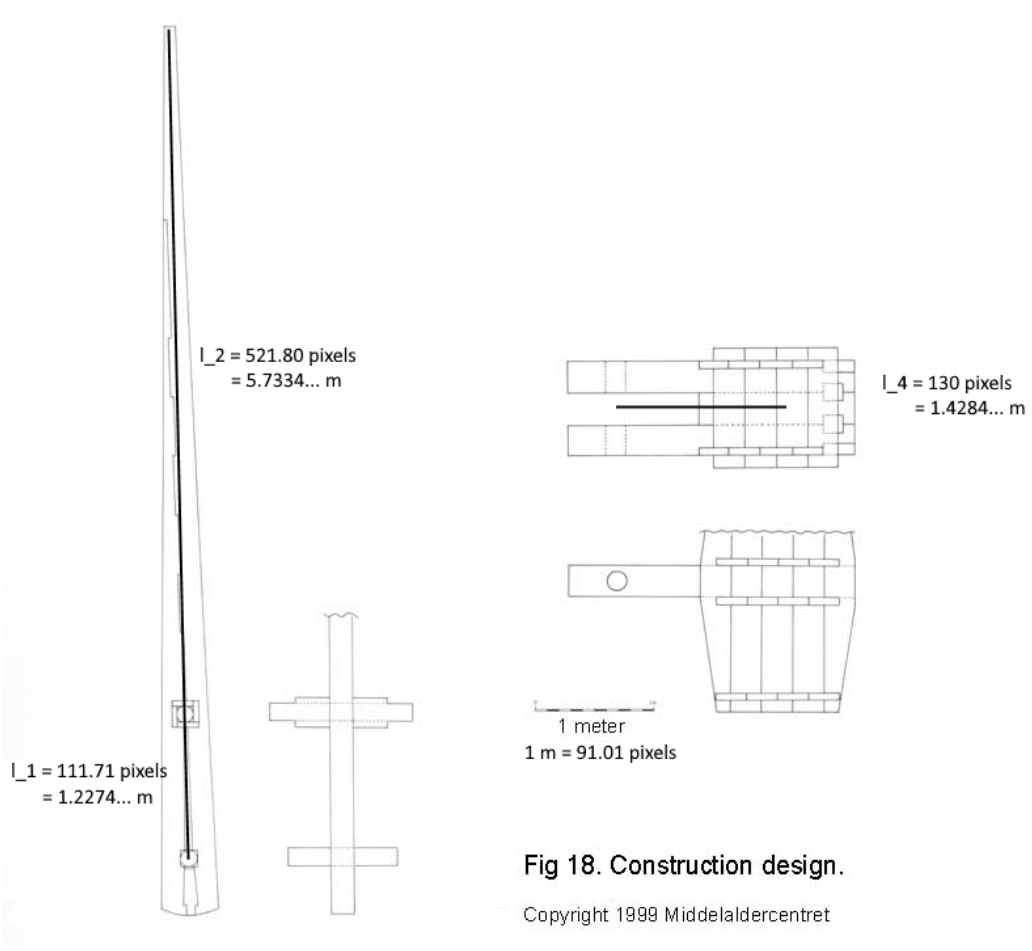


Figure 25: The first blue print of the reconstruction in Denmark [Hansen, 1992]. From this picture we can extract the values l_1, l_2 and l_4 that were used in their setup.

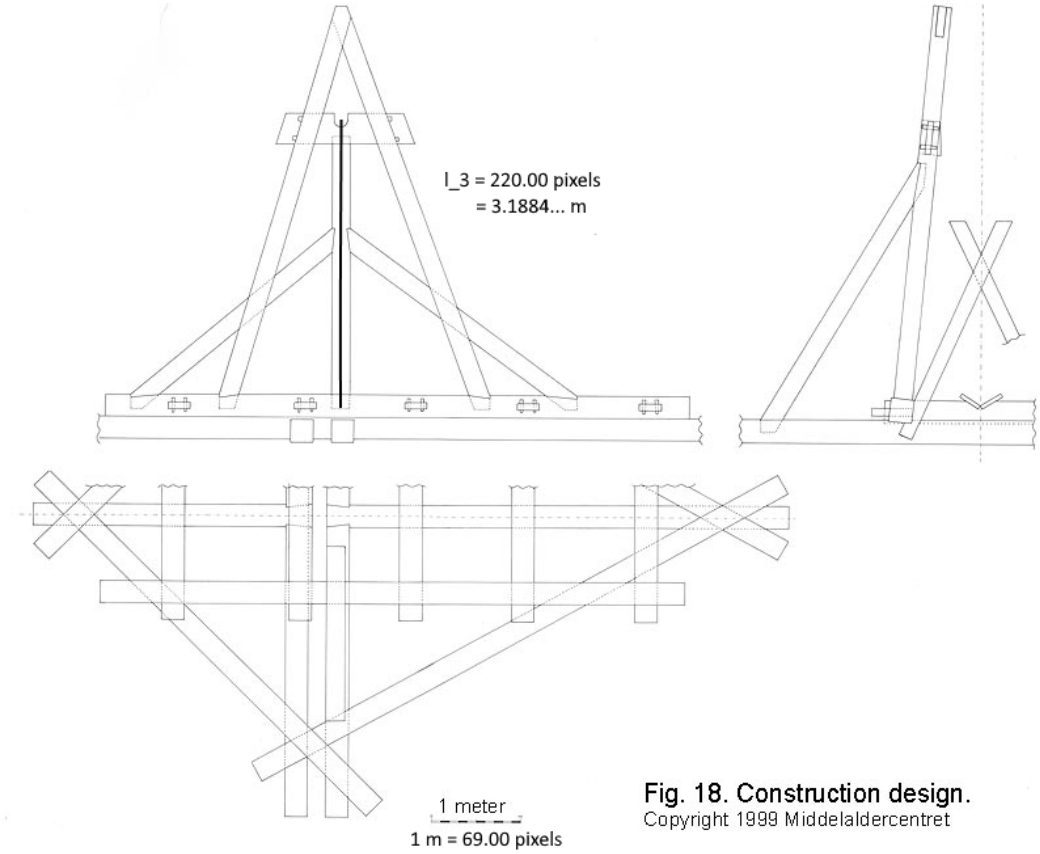


Figure 26: The second blue print of the reconstruction in Denmark [Hansen, 1992]. From this picture we can extract the value l_3 that was used in their setup.

Shots 1-4:	Ballast 1000 kg	sling 5 m	projectile 15 kg.
Shots 5-8:	Ballast 1500 kg	sling 5 m	projectile 15 kg
Shots 9-13:	Ballast 2000 kg	sling 5 m	projectile 15 kg
Shot 14:	Ballast 1000 kg	sling 5 m	projectile 15 kg
Shot 15:	Ballast 2000 kg	sling 5 m	projectile 15 kg
Shot 16:	Ballast 2000 kg	sling 5 m	projectile 15 kg
Shot 17:	Ballast 2000 kg	sling 5 m	projectile 15 kg
Shot 18:	Ballast 2000 kg	sling 4 m	projectile 15 kg
Shot 19:	Ballast 2000 kg	sling 5 m	projectile 20 kg
Shot 20:	Ballast 2000 kg	sling 5 m	projectile 25 kg
Shot 21:	Ballast 2000 kg	sling 5 m	projectile 47 kg

Figure 27: From this table we can extract the values m_1, m_2 and l_5 for a certain shot that was done with the reconstruction in Denmark [Hansen, 1992].

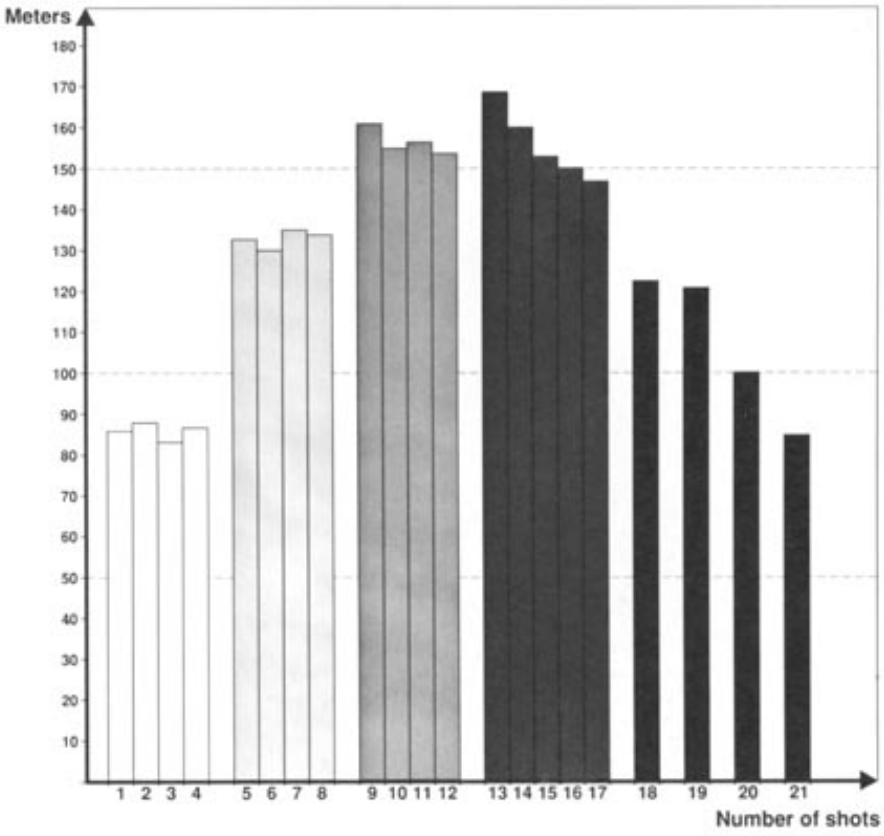


Fig. 17. Experiment results. Drawn by C. Oksen.

Figure 28: In this graph we can see that the optimal range of 168 meters was reached with $m_1 = 2000$ kg, $m_2 = 15$ kg and $l_5 = 5$ m [Hansen, 1992].

A.2 Python code

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as integrate
import sympy as sp
import time
import matplotlib.animation as animation

g = 9.81
t = sp.Symbol('t')
theta, phi, psi = sp.symbols('theta phi psi', cls=sp.Function)
th, ph, ps = theta(t), phi(t), psi(t)

class trebuchet:
    def __init__(self, l1=1.2, l2=5.7, l3=3.2, l4=1.4, l5=5., m1=2000.,
                 m2=15., theta0=0.7*np.pi, dtheta0=0., phi0 = None,
                 dphi0 = 0, psi0 = None, dpsi0 = 0, dt = 0.001, T=1.):
        self._l1 = l1
        self._l2 = l2
        self._l3 = l3
        self._l4 = l4
        self._l5 = l5
        self._m1 = m1
        self._m2 = m2
        self._theta0 = theta0
        self._dtheta0 = dtheta0
        self._phi0 = phi0 if phi0 != None else np.pi - theta0
        self._dphi0 = dphi0
        self._psi0 = psi0 if psi0 != None else theta0 - 0.5*np.pi
        self._dpsi0 = dpsi0
        self._T = T
        self._dt = dt
        self._t = list(np.arange(0,T,self._dt))
        self._r1, self._r2 = self.__r12()
        self._A, self._v = self.__Av(self.__L())
        self._angles = 1 + (l4 != 0) + (l5 != 0)
        self._y = self.__set_y()
        self._delta, self._epsilon = self._r2[0], self._r2[1]
        self._beta, self._gamma = self._delta.diff(t), self._epsilon.diff(t)
        self._s_list, self._E_list = self.__shooting_list()
        self._s_opt, self._E_opt = self.__optimal_angle()
        self._L = self.__L()

##########################################################################
#support methods

    def __r12(self):
        a, b = th+ph, ps-th
        rn1 = lambda x,y,z: y*sp.sin(th)+z*sp.sin(x)
        rn2 = lambda x,y,z: y*sp.cos(th)+z*sp.cos(x)+self._l3
        r1 = [rn1(a,self._l1,-self._l4), rn2(a,-self._l1,self._l4)]
        r2 = [rn1(b,-self._l2,-self._l5), rn2(b,self._l2,-self._l5)]
        return r1, r2
```

```
def __L(self):
    [x1, y1], [x2, y2] = self._r1[:2], self._r2[:2]
    vn_sq = lambda x,y: (x.diff(t))**2+(y.diff(t))**2
    v1_sq, v2_sq = vn_sq(x1,y1), vn_sq(x2,y2)
    m1, m2 = self._m1, self._m2
    return 0.5*(m1*v1_sq+m2*v2_sq)- g*(m1*y1+m2*y2)

def __Av(self, L):
    A, V = [], []
    angls = [th]
    if self._l4 != 0: angls.append(ph)
    if self._l5 != 0: angls.append(ps)
    for angle in angls:
        f = L.diff(angle)-L.diff(angle.diff(t)).diff(t)
        v = f
        for angle in angls:
            spd = sp.Derivative(angle, (t, 2))
            a = (f - f.subs(spd, 0))/spd
            A.append(sp.simplify(a))
            v = v.subs(spd, 0)
        V.append(sp.simplify(-v))
    return A,V

def __fill_out_matrix(self, C, status):
    global t
    global theta, phi, psi
    C = [c.subs(sp.Derivative(theta(t), t),status[1]) for c in C]
    C = [c.subs(theta(t),status[0]) for c in C]
    C = [c.subs(sp.Derivative(phi(t), t),status[3]) for c in C]
    C = [c.subs(phi(t),status[2]) for c in C]
    C = [c.subs(sp.Derivative(psi(t), t),status[5]) for c in C]
    C = [c.subs(psi(t),status[4]) for c in C]
    return C

def __derivs(self, status, t):
    A = self.__fill_out_matrix(self._A, status)
    v = self.__fill_out_matrix(self._v, status)
    B = np.matrix(np.zeros(shape=(self._angles, self._angles)))
    W = np.matrix(np.zeros(shape=(self._angles, 1)) )
    for i in range(self._angles**2):
        B.itemset(i,float(A[i]))
    for i in range(self._angles):
        W.itemset(i,float(v[i]))
    det = np.linalg.det(B)
    if self.angles == 3:
        test = det/(-self.l4**2*self.l5**2*self.m1*self.m2)
        if abs(test) < 0.1:
            print('det(A) = '+str(det)+', inverse becomes: '+str(B.getI()))
    M = B.getI()*W
    ddth, ddph, ddps = float(M[0]), 0., 0.
    if self._l4 != 0:
        ddph = float(M[1])
```

```
        if self._l5 != 0:
            ddps = float(M[2])
        return [status[1], ddth, status[3], ddph, status[5], ddps]

    def __set_y(self):
        state0 = [self.theta0, self.dtheta0, self.phi0,
                  self.dphi0, self.psi0, self.dpsi0]
        y = integrate.odeint(func = self.__derivs, y0 = state0, t = self.t)
        return y

    def __te(self, gamma, epsilon):
        global g
        w = gamma**2 + 2*g*epsilon
        if w<0: return 0
        else: return (gamma+w**(0.5))/g

    def __s(self, beta, gamma, epsilon):
        return beta*self.__te(gamma, epsilon)

    def __E(self, beta, gamma, epsilon):
        global g
        return 0.5*self.m2*(beta**2+(-g*self.__te(gamma, epsilon)*gamma)**2)

    def __shooting_list(self):
        s_list, E_list = [], []
        for status in self._y:
            be = self.fill_out_function(self._beta, status)
            ga = self.fill_out_function(self._gamma, status)
            ep = self.fill_out_function(self._epsilon, status)
            s_list.append(self.__s(be, ga, ep))
            E_list.append(self.__E(be, ga, ep))
        return s_list, E_list

    def __optimal_angle(self):
        theta = [item[0] for item in self._y]
        s_opt = theta[self._s_list.index(max(self._s_list))]
        E_opt = theta[self._E_list.index(max(self._E_list))]
        return s_opt, E_opt

    def fill_out_function(self, f, status):
        global t
        global theta, phi, psi
        f = f.subs(sp.Derivative(theta(t), t),status[1])
        f = f.subs(theta(t),status[0])
        f = f.subs(sp.Derivative(phi(t), t),status[3])
        f = f.subs(phi(t),status[2])
        f = f.subs(sp.Derivative(psi(t), t),status[5])
        f = f.subs(psi(t),status[4])
        return f

################################################################################
#property's
```

```
@property
def A(self):
    return self._A

@property
def v(self):
    return self._v

@property
def l1(self):
    return self._l1

@property
def l2(self):
    return self._l2

@property
def l3(self):
    return self._l3

@property
def l4(self):
    return self._l4

@property
def l5(self):
    return self._l5

@property
def m1(self):
    return self._m1

@property
def m2(self):
    return self._m2

@property
def theta0(self):
    return self._theta0

@property
def dtheta0(self):
    return self._dtheta0

@property
def phi0(self):
    return self._phi0

@property
def dphi0(self):
    return self._dphi0

@property
```

```
def psi0(self):
    return self._psi0

@property
def dpsi0(self):
    return self._dpsi0

@property
def T(self):
    return self._T

@property
def dt(self):
    return self._dt

@property
def t(self):
    return self._t

@property
def angles(self):
    return self._angles

@property
def r1(self):
    return self._r1

@property
def r2(self):
    return self._r2

@property
def y(self):
    return self._y

@property
def beta(self):
    return self._beta

@property
def gamma(self):
    return self._gamma

@property
def delta(self):
    return self._delta

@property
def epsilon(self):
    return self._epsilon

@property
def s_list(self):
```

```
        return self._s_list

    @property
    def E_list(self):
        return self._E_list

    @property
    def s_opt(self):
        return self._s_opt

    @property
    def E_opt(self):
        return self._E_opt

    @property
    def L(self):
        return self._L

##############################################################################

def num_pic(treb):
    num, ax1 = plt.subplots(figsize = (14,7))
    ax2 = ax1.twinx()
    plt.rc('font', size = 20)
    state = treb.y
    ax2.plot([0,treb.T],[0,0], color='lightgrey', linewidth = 1)
    ax2.plot(treb.t, [item[1] for item in state],
             color='pink', label=r'$\dot{\theta}$', linewidth=3)
    ax1.plot(treb.t, [item[0] for item in state],
             color='red', label=r'$\theta$', linewidth=3)
    if treb.l4 != 0:
        ax2.plot(treb.t, [item[3] for item in state],
                 color='lightgreen', label=r'$\dot{\phi}$', linewidth=3)
        ax1.plot(treb.t, [item[2] for item in state],
                 color='darkgreen', label=r'$\phi$', linewidth=3)
    if treb.l5 != 0:
        ax2.plot(treb.t, [item[5] for item in state],
                 color='lightskyblue' ,label=r'$\dot{\psi}$', linewidth=3)
        ax1.plot(treb.t, [item[4] for item in state],
                 color='blue', label=r'$\psi$', linewidth=3)
    ax1.legend(loc='upper left')
    ax2.legend(loc='upper right')
    ax1.set_xlabel('t(s)')
    ax1.set_ylabel('angle(rad)')
    ax2.set_ylabel('velocity(rad/s)')
    if treb.angles == 1 and treb.T<=2.:
        ax1.set_ylim(-1,3)
        ax2.set_ylim(-10,30)
    elif treb.angles == 2 and treb.T<=2.:
        ax1.set_ylim(-2,6)
        ax2.set_ylim(-20,60)
    elif treb.angles == 3 and treb.T<=2.:
        ax1.set_ylim(-4,6)
```

```
        ax2.set_ylim(-40,60)
    if treb.angles == 3 and treb.l4 <= 0.01 and treb.l5 <= 0.01:
        name = 'num_small'
    if treb.angles == 2 and treb.l4 <= 0.01:
        name = 'num_small'
    elif treb.T<4:
        name = 'num_short'
    else:
        name = 'num_long'
    name = name + str(treb.angles) + '.png'
    num.savefig(name)

##########################################################################

def path_plot(treb, n, focus, clr='lightgrey'):
    global g
    be = float(treb.fill_out_function(treb.beta, treb.y[n]))
    ga = float(treb.fill_out_function(treb.gamma, treb.y[n]))
    ep = float(treb.fill_out_function(treb.epsilon, treb.y[n]))
    de = float(treb.fill_out_function(treb.delta, treb.y[n]))
    w = ga**2 + 2*g*ep
    if w<0: te = 0
    else: te = (ga+w**(0.5))/g
    z = np.linspace(0, te, 100)
    x = be*z + de
    y = -0.5*g*z**2 + ga*z + ep
    plt.plot(x, y , color=clr)
    s1 = treb.E_list[n]/max(treb.E_list)
    plt.scatter(x[-1], 0, color=clr, s=float(abs(s1*400.)))
    lbl, s2= '', treb.E_list[n]
    if treb.angles == 1:
        s2 = s2/(10**4)
        lbl = str(round(s2))+r'$\cdot 10^4$ J '
    elif treb.angles == 2:
        s2 = s2/(10**5)
        lbl = str(round(s2))+r'$\cdot 10^5$ J '
    elif treb.angles == 3:
        s2 = s2/(10**5)
        lbl = str(round(s2))+r'$\cdot 10^5$ J '
    plt.scatter(x[-1], 0, color=clr, s=60., label=lbl)

def picture(treb, k = 6, focus='J'):
    colors = ['tab:blue', 'tab:orange', 'tab:green', 'tab:red', 'tab:purple',
              'tab:brown', 'tab:pink', 'tab:gray', 'tab:olive', 'tab:cyan',
              'b', 'r']
    theta = [round(item[0],1) for item in treb.y]
    if treb.angles == 3:
        R = [674, 750, 655, 760, 765, 780]
    else:
        a = theta.index(min(theta, key=lambda x:abs(x-0.2)))
        b = theta.index(min(theta, key=lambda x:abs(x-1.)))
        R = np.linspace(a, b , k)
    shooting = plt.figure(figsize = (14,7))
```

```
    for i in range(min(k,len(colors))):
        path_plot(treb, int(R[i]), focus, clr = colors[i])
    plt.legend(loc='upper right')
    plt.xlabel('x(m)')
    plt.ylabel('y(m)')
    plt.ylim(0)
    naam = 'shooting' + str(treb.angles) + '.png'
    shooting.savefig(naam)

##########################################################################

def opti(treb):
    theta = [item[0]/np.pi for item in treb.y]
    theta.reverse()
    s_list = treb.s_list
    s_list.reverse()
    opti, ax3 = plt.subplots(figsize = (14,7))
    ax4 = ax3.twinx()
    ax3.set_xlabel(r'$\theta_R(\pi$ rad)')
    ax3.set_ylabel(r's(m)')
    if treb.angles ==1:
        ax4.set_ylabel(r'E($10^5$J)')
        E_list = [E*10**(-5) for E in treb.E_list]
    elif treb.angles ==2:
        ax4.set_ylabel(r'E($10^5$J)')
        E_list = [E*10**(-5) for E in treb.E_list]
    elif treb.angles ==3:
        ax4.set_ylabel(r'E($10^6$J)')
        E_list = [E*10**(-6) for E in treb.E_list]
    E_list.reverse()
    ax3.plot([min(theta),max(theta)], [0,0], color= 'lightgrey')
    ax3.plot([treb.s_opt/np.pi, treb.s_opt/np.pi], [0,max(s_list)],
             color= 'tab:blue', linestyle=':', linewidth=3)
    print(treb.s_opt/np.pi)
    print(max(s_list))
    ax4.plot([treb.E_opt/np.pi, treb.E_opt/np.pi], [0,max(E_list)],
             color= 'tab:red', linestyle=':', linewidth=3)
    print(treb.E_opt/np.pi)
    print(max(E_list))
    ax3.plot(theta, s_list, color= 'tab:blue', label='Range', linewidth=3)
    ax4.plot(theta, E_list, color= 'tab:red', label='Impact', linewidth=3)
    ax3.set_xlim(min(theta),max(theta))
    if treb.angles == 1:
        ax3.set_ylim(-20,100)
        ax4.set_ylim(-20,100)
    elif treb.angles == 2:
        ax3.set_ylim(-50,400)
        ax4.set_ylim(-50,400)
    elif treb.angles == 3:
        ax3.set_ylim(-50,600)
        ax4.set_ylim(-50,600)
    ax3.legend(loc='upper left')
    ax4.legend(loc='upper right')
```

```
    naam = 'opti' + str(treb.angles) + '.png'
    opti.savefig(naam)

##########################################################################

def check(treb):
    global t, th, ph, ps
    f = treb.L.diff(th.diff(t))*th.diff(t) - treb.L
    if treb._l4 != 0: f = f + treb.L.diff(ph.diff(t))*ph.diff(t)
    if treb._l5 != 0: f = f + treb.L.diff(ps.diff(t))*ps.diff(t)
    E_lst = []
    E_lst.append(treb.fill_out_function(f, treb.y))
    if max(E_lst)-min(E_lst) != 0:
        print(max(E_lst)-min(E_lst))

##########################################################################

def p1(n):
    x, x1, x2, y1, y2 = [], [], [], [], []
    for i in range(n):
        psi0 = (float(i)/n)*2
        x.append(psi0)
        treb = trebuchet(psi0 = psi0*np.pi, dt = 0.01)
        check(treb)
        x1.append(treb.s_opt/np.pi)
        y1.append(max(treb.s_list))
        x2.append(treb.E_opt/np.pi)
        y2.append(max(treb.E_list)/(10**7))
        print(x)
        print(y1)
        print(y2)
        print(time.strftime("%M%S", time.localtime()))
    x.append(2.)
    x1.append(x1[0])
    y1.append(y1[0])
    x2.append(x2[0])
    y2.append(y2[0])
    return x, x1, x2, y1, y2

def p2(n, x, y1, y2):
    p0, ax5 = plt.subplots(figsize = (14,7))
    ax6 = ax5.twinx()
    ax5.set_xlabel(r'$\psi_0(\pi$_rad)')
    ax6.set_ylabel(r'$E_{opt}(10^7$J)', color= 'tab:red')
    ax5.set_ylabel(r'$s_{opt}(m)$', color= 'tab:blue')
    ax5.plot([x[y1.index(max(y1))],x[y1.index(max(y1))]], [0,max(y1)],
                color= 'tab:blue', linestyle=':')
    ax6.plot([x[y2.index(max(y2))],x[y2.index(max(y2))]], [0,max(y2)],
                color= 'tab:red', linestyle=':')
    ax5.plot(x, y1, color= 'tab:blue', label='optimal_distance')
    ax6.plot(x, y2, color= 'tab:red', label='optimal_impact')
    ax5.set_xlim(x[0], x[-1])
    naam = 'psi' + str(n) + '.png'
```

```
    p0.savefig(naam)

##########################################################################

print(time.strftime("%M%S", time.localtime()))
treb1 = trebuchet(l4 = 0, l5=0)
num_pic(treb1)
opti(treb1)
picture(treb1)
check(treb1)
print(time.strftime("%M%S", time.localtime()))
treb2 = trebuchet(l5=0)
num_pic(treb2)
opti(treb2)
picture(treb2)
check(treb2)
print(time.strftime("%M%S", time.localtime()))
treb3 = trebuchet()
num_pic(treb3)
opti(treb3)
picture(treb3)
check(treb3)
print(time.strftime("%M%S", time.localtime()))
treb4 = trebuchet(l4 = 0, l5=0, T=5.)
num_pic(treb4)
check(treb4)
print(time.strftime("%M%S", time.localtime()))
treb5 = trebuchet(l5=0, T=5.)
num_pic(treb5)
check(treb5)
print(time.strftime("%M%S", time.localtime()))
treb6 = trebuchet(T=5.)
num_pic(treb6)
check(treb6)
print(time.strftime("%M%S", time.localtime()))

##########################################################################

print(time.strftime("%H%M%S", time.localtime()))
n = 2500
x, x1, x2, y1, y2 = p1(n)
p2(n, x, y1, y2)
print(time.strftime("%H%M%S", time.localtime()))

##########################################################################

print(time.strftime("%M%S", time.localtime()))
treb = trebuchet()
y = treb.y

x1a = treb.l1*np.sin(y[:, 0])
y1a = -treb.l1*np.cos(y[:, 0])+treb.l3
x2a = -treb.l2*np.sin(y[:, 0])
```

```
y2a = treb.l2*np.cos(y[:, 0]) + treb.l3
x1b = x1a - treb.l4*np.sin(y[:, 0]+y[:, 2])
y1b = y1a + treb.l4*np.cos(y[:, 0]+y[:, 2])
x2b = x2a - treb.l5*np.sin(y[:, 4]-y[:, 0])
y2b = y2a - treb.l5*np.cos(y[:, 4]-y[:, 0])
fig = plt.figure()
s = treb.l2+treb.l5
ax = fig.add_subplot(111, autoscale_on=False, xlim=(-s,s), ylim=(-s,s))
ax.grid()
line, = ax.plot([], [], 'o-', lw=2)
time_template = 'time = %.1fs'
time_text = ax.text(0.05, 0.9, '', transform=ax.transAxes)
def init():
    line.set_data([], [])
    time_text.set_text('')
    return line, time_text
def animate(i):
    dt = 0.01
    thisx = [0,0,x1a[i],x1b[i],x1a[i],x2a[i],x2b[i]]
    thisy = [0,treb.l3,y1a[i],y1b[i],y1a[i],y2a[i],y2b[i]]
    line.set_data(thisx, thisy)
    time_text.set_text(time_template % (i*dt))
    return line, time_text
ani = animation.FuncAnimation(fig, animate, np.arange(1, len(y)),
                              interval=1, blit=True, init_func=init)
angles = 3
if treb.l4 == 0:
    angles -= 1
if treb.l5 == 0:
    angles -= 1
naam = 'ani' + str(angles) + '_' + str(treb.T) + '_' + str(treb.dt) + '_' + '.mp4'
ani.save(naam)
plt.show()

print(time.strftime("%M%S", time.localtime()))
```