Modeling and Simulation of Fragmenting Blasts

Ishaan Srivastava

December 2019

1 Introduction

In this assignment, we construct a model to simulate the behavior of a two-stage firework comprising a propulsive and explosive charge. First we model the firework being fired out of a mortar. We initialise the blast from the moving firework, and use this explosion to model the dynamics and thermodynamics of particles shot outward by the explosive charge. Once these parts have been modelled, a genetic algorithm is used to determine three system parameters: m_e, the mass of the explosive, m_f the mass of the firework, and t_e, the time at which the explosion occurs. the goal is to find parameters that create a large spread of ejected particles as safely as possible

2 Modelling and Theory

2.1 Firework Modelling

We make a series of simplifying assumptions, namely:

- The firework does not reach an altitude high enough that we need to factor in changing air density.
- The firework is spherical in shape.
- The firework is initially launched in the vertical direction via an explosion underneath it. Thus initial velocity is only in the $\mathbf{e_3}$ direction (corresponding to the vertical z-axis), and 0 in the other 2.
- Rotational velocities are low enough for point mass idealisation of the firework.

Newton's second law gives us $m_r\mathbf{a_r} = \mathbf{\Psi_r^{tot}}$, where $\mathbf{\Psi_r^{tot}}$ is the total or net force vector, given by:

$$\mathbf{\Psi_r^{tot}} = \mathbf{F_{d,r}} + \mathbf{F_{g,r}}$$

, where the second and third terms in the equation are the drag force and gravitational force vectors, respectively. Drag force is given by:

$$\mathbf{F_{d,r}} = \frac{1}{2}\rho_a C_{d,r} A_{r,x} \|\mathbf{v_a} - \mathbf{v_r}\|(\mathbf{v_a} - \mathbf{v_r})$$

Where ρ_a is air density, $C_{d,r}$ is drag coefficient, $A_{r,x}$ is firework cross-sectional surface area, $\mathbf{v_a}$ is the air velocity (equivalent to the zero vector for this project) and $\mathbf{v_r}$ is the firework velocity. The drag coefficient is approximated by Reynolds number of the drag flows acting on the sphere, which in turn is calculated using the equation

$$Re = \frac{2R_r \|\mathbf{v_r} - \boldsymbol{v_a}\|}{\mu_a}$$

, where R_r is the firework radius, and μ_a is air viscosity. Refer to function computeCD in the code to see how drag coefficient is approximated using the Reynolds number. Going back to our drag force equation, the next term is $A_{r,x} = \pi R_r^2$, which comes from the standard equation for cross-sectional area of a sphere. Note that the norm in the drag force is equation is the standard 2-norm. The gravitational force is given by

$$\mathbf{F_{g,r}} = -m_r g \boldsymbol{e_3}$$

Where m_r is the total mass, given by $m_r = m_s + m_e + m_p$, where m_s is the inert structural mask, m_e is the previously defined mass of the explosive, and m_p is the total mass of the particles. How to calculate m_p is discussed later in the report, in the particle dynamics section. Looking at our equation for $\mathbf{F_{g,r}}$, we note that it acts only in the vertical direction and is 0 elsewhere. Having found suitable expressions for $\mathbf{\Psi_r^{tot}}$, we use Forward Euler integration to perform time stepping for the firework's position and velocity, given by

$$\boldsymbol{v_r}(t + \Delta t) = \boldsymbol{v_r}(t) + \boldsymbol{a_r}(t) = \boldsymbol{v_r}(t) + \frac{\Delta t}{m_r} \mathbf{\Psi^{tot}}(t)$$

$$\boldsymbol{r_r}(t + \Delta t) = \boldsymbol{r_r}(t) + \Delta t \mathbf{v_r}(t)$$

Where $\boldsymbol{r_r}$ is the position vector corresponding to the firework. We integrate in time till t_e, at which point the firework explodes. In our model, the firework starts at the origin, such that $\boldsymbol{r_r}(t = 0) = \mathbf{0}$, and has initial velocity $\boldsymbol{v_r}(t = 0) = (\sqrt{\frac{2\eta H_e m_f}{m_r}})\boldsymbol{e_3}$. The derivation of this equation is presented in the particle dynamics section of the report. As mentioned before, integrating our total force equation in time gives us the position and velocity vector of the firework at the time of the explosion, and we denote them as

$$\boldsymbol{r_r}(t = t_e) = \boldsymbol{r_0}$$

$$\boldsymbol{v_r}(t = t_e) = \boldsymbol{v_0}$$

These terms are used in our section on particle dynamics. As a final note, if the firework reaches the ground before the t_e, then the firework explodes at ground level instead.

2.2 Ejecta Modelling and Particle Dynamics

As with the firework, we model the explosive ejecta or particles after making simplifying assumptions:

- They are instantly accelerated to high speeds when the explosion occurs
- They are initially arranged in a uniformly random spherical shell around the explosive charge.
- Point mass idealisation is valid.
- They cool uniformly, with the rate of cooling not strongly depending on velocity.

We initialise N_P particles on a uniformly random spherical shell around the position of the firework, with the general equation:

$$\boldsymbol{r_i}(t = t_e) = \boldsymbol{r_0} + R_r(cos(\theta_s)sin(\phi_s)\boldsymbol{e_1} + sin(\theta_s)sin(\phi_s)\boldsymbol{e_2} + cos(\phi_s)\boldsymbol{e_3})$$

Where $\boldsymbol{r_0}$ is the position vector corresponding to center of the firework (which we found in the previous section), R_r is the distance of the particles from the center of the firework, equivalent to the firework's radius, θ_s is the azimuthal angle relative to the $\boldsymbol{e_1}$ direction and ϕ_s is the "polar angle" relative to the $\boldsymbol{e_3}$ direction. Note that the particles are initialised uniformly on the surface of the sphere, not uniformly on θ_s and ϕ_s. The particles themselves are modelled as spheres of fixed density ρ_p and radius R_p. Relating the volume of a sphere to its density, we get that the mass of each individual particle m_i is given by:

$$m_i = \frac{4}{3}\pi R_r^3 \rho_p$$

This gives us an expression for the total mass of the particles m_p, which we had used to calculate m_r earlier in this report

$$m_p = \sum_{i=1}^{N_P} m_i = N_P m_i$$

We now find the initial velocity of the particles. To do this, we assume that a certain fixed percentage of the explosive's chemical energy becomes kinetic energy in the particles. We also know that the total energy produced by the explosion is given by the product of the mass of the explosive m_e and the heat of explosion of the explosive, denoted by H_e, meaning total energy released during the explosion is $H_e m_e$. In explosions, most of the energy released is lost in the form of heat, while some is also converted into other forms such as sound energy in the form of a shock wave. We assume that some fixed fraction of the explosion energy, denoted η, is transformed into additional kinetic energy in the projectiles. By conservation of energy, we can express the relationship between the explosion energy and the velocity change of the particles by:

$$\sum_{i=1}^{N_P} = \frac{1}{2} m_i \Delta v_i^2 = \eta H_e m_e$$

If all particles have the same velocity change, $\Delta v_i = \Delta v$, giving:

$$\Delta v_i = \Delta v = (\sqrt{\frac{2\eta H_e m_e}{m_r}})$$

This is similar to our earlier equation for $\boldsymbol{v_r}(t=0)$, except that was in terms of m_f instead of m_e. We assume that all the particles have the same velocity change in terms of magnitude, however since the particles themselves are pointing in different directions, this velocity change will affect different particles differently in terms of direction. We assume that the particles are sent directly outward from the center of the uniformly random spherical shell on which the particles are initialised. To find the direction vectors corresponding to directly outward, we find the normal vector for each particle and then normalise it since we're interested only in its direction. Hence, outward unit normal vector $\boldsymbol{n_i} = \frac{\boldsymbol{r_i} - \boldsymbol{r_0}}{\|\boldsymbol{r_i} - \boldsymbol{r_0}\|}$ and

$$\boldsymbol{v_i}(t = t_e) = \boldsymbol{v_0} + \Delta v \boldsymbol{n_i}$$

Where $\boldsymbol{v_r}(t = t_e) = \boldsymbol{v_0}$ is the velocity of the firework at the time of the explosion, which we have defined earlier in the report. Once the explosion has taken place, the only two forces acting on a given particle are drag and gravity. In other words,

$$\mathbf{\Psi_i^{tot}} = \mathbf{F_{d,i}} + \mathbf{F_{g,i}}$$

We recall that $m_i \mathbf{a_i} = \mathbf{\Psi_i^{tot}}$. The following equations are analogous to the firework case, and terms are defined only if they haven't already been.

$$\mathbf{F_{d,i}} = \frac{1}{2}\rho_a C_{d,i} A_{p,x} \|\mathbf{v_a} - \mathbf{v_i}\|(\mathbf{v_a} - \mathbf{v_i})$$

We find the Reynolds Number $Re = \frac{2R_p\|\mathbf{v_i} - \boldsymbol{v_a}\|}{\mu_a}$ and use it to approximate $C_{d,i}$. Cross-sectional area of each particle $A_{p,x} = \pi R_p^2$. $\boldsymbol{v_i}$ refers to the velocity of a given particle. Once we solve for $\mathbf{\Psi_i^{tot}}$ at a given time t, we use the Forward Euler method to integrate the motion equations for the particles. Each particle is tracked until it reaches the ground, corresponding to $z \leq 0$. Once a particle reached the ground, its other two coordinates and time of impact were recorded.

3 Deliverables

3.1 Particle Dynamics

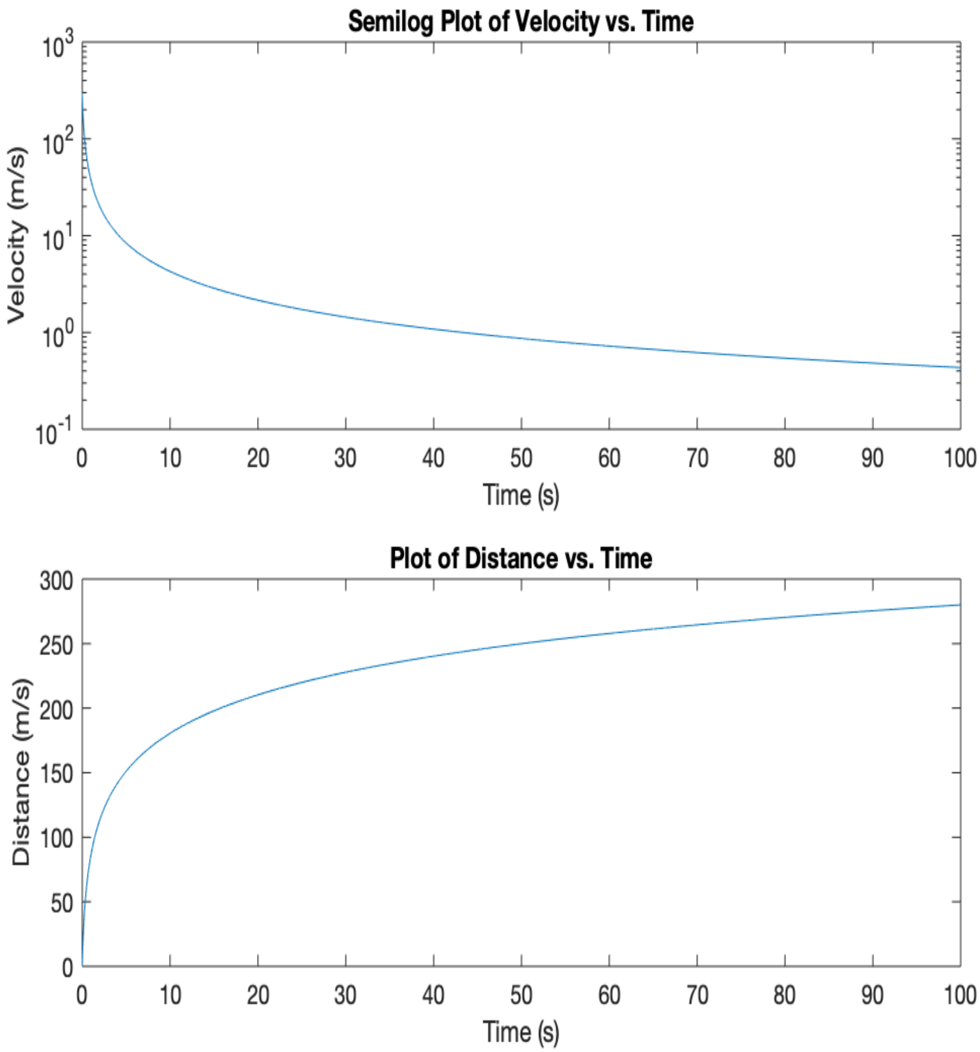


Figure 1: Position and Semilog Velocity Plot for a Particle

The key feature of the velocity semilog plot is the exponential velocity decrease as time progresses. In the first 10 seconds, the velocity drops from 300 m/s to about 7m/s, but even after 100 seconds, the velocity is not 0. We can attribute this to solely to the effect of the drag force, since we're neglecting gravity here. Looking at the semilog plot, we can see the relationship between drag force and velocity is non-linear, but we cannot figure out the exponent in their relationship from the plot alone.

If we try to make similar plots with either the parameters given in the project spec or our optimal parameters, runtime increase, but at the cost of accuracy, and with the risk of introducing numerical instability. A clear example of this is when we perform our Forward Euler integration, the firework or particle may already be a few meters in the ground, which is inaccurate and affects the accuracy and quality of our results. I noticed these issues for Δt of the order $10^{-}1$

3.2 Particle Thermodynamics

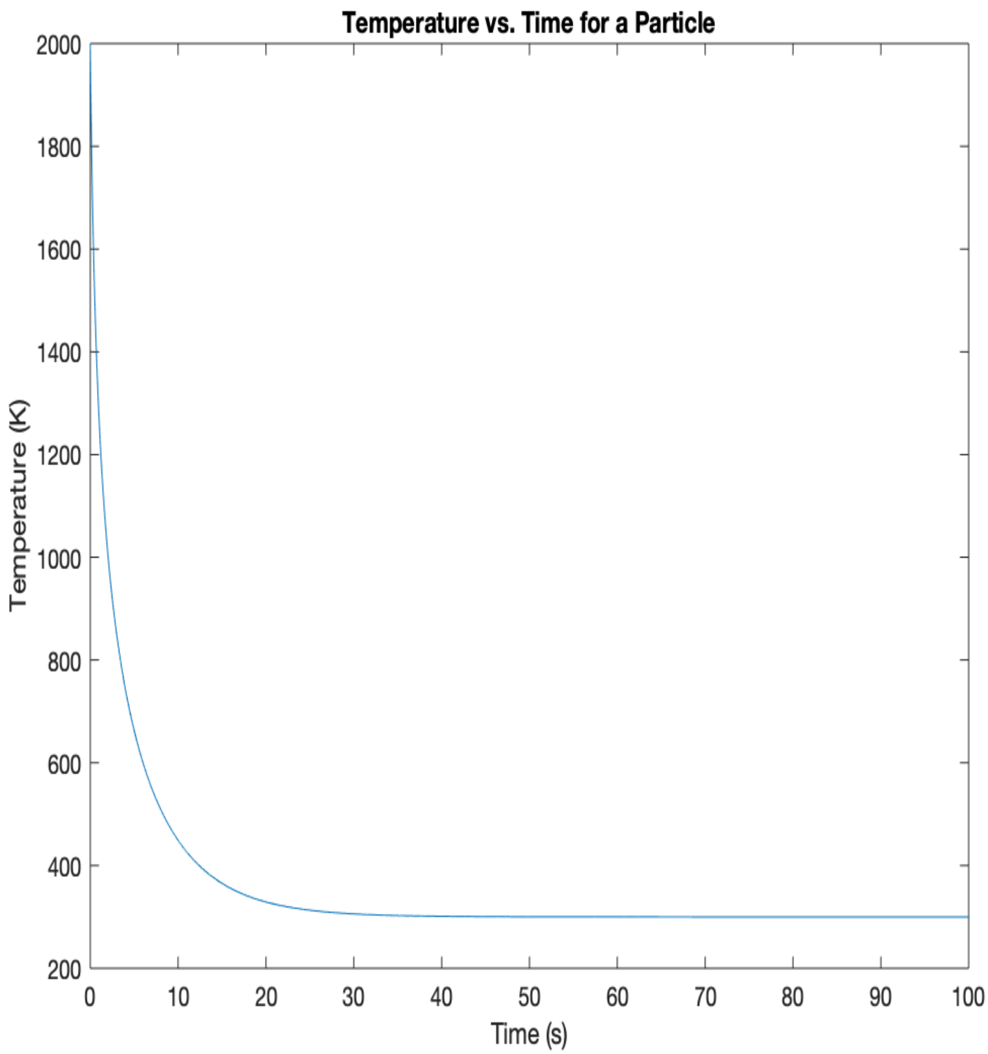


Figure 2: Temperature vs. Time for a Particle

Looking at Figure 2, the notable feature is that temperature decreases exponentially with the passage of time. Given the quartic terms used in $\dot{Q}_r$, this non-linear relationship between temperature and time is unsurprising.

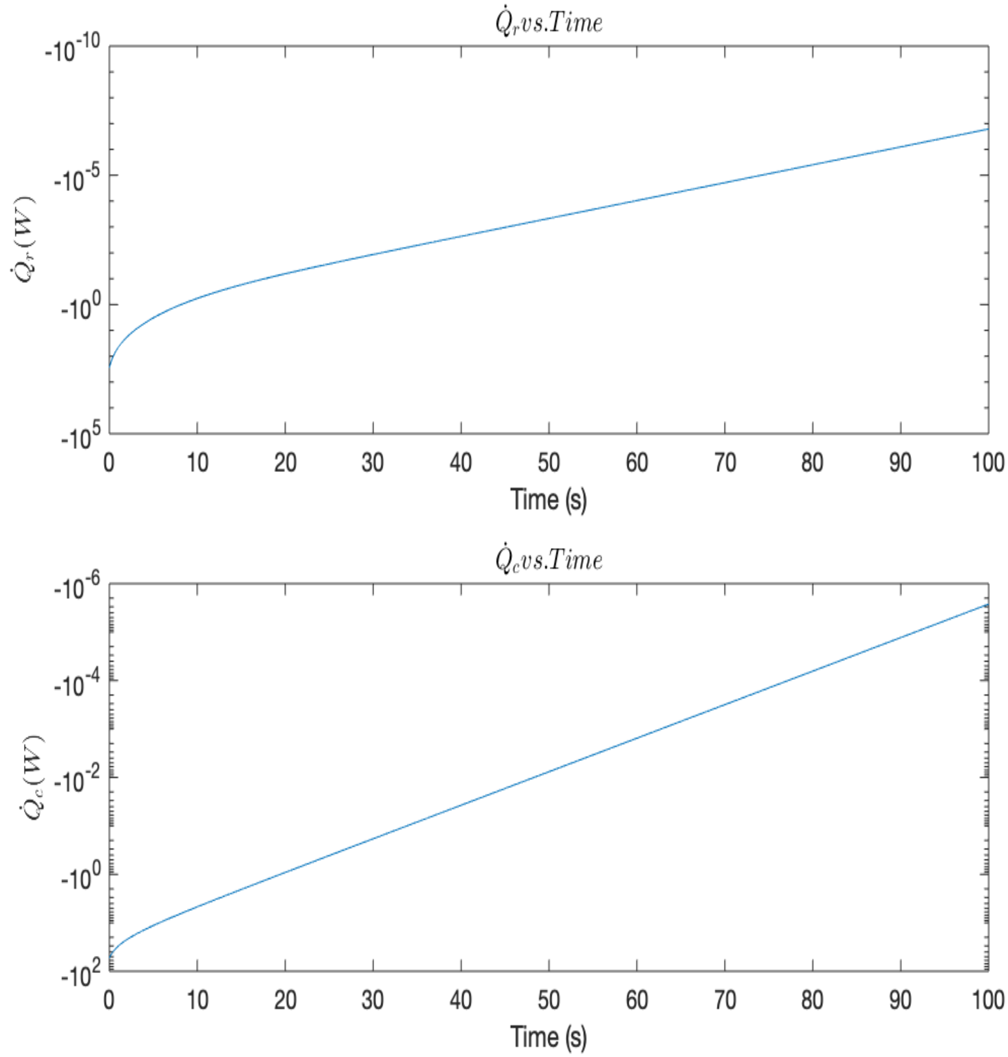


Figure 3: Semilog plots of $\dot{Q}_r$ and $\dot{Q}_c$ vs. Time for a Particle

Note that both graphs have only negative values on the y-axis. This gives us the direction of heat transfer. Since all the values are negative, heat is flowing out of the particle. We can now discuss the shapes. Once again, we see our variables are non-linearly related. Both graphs asymptotically converge to 0, consistent with thermodynamic equilibrium, but the graph of $\dot{Q}_r$ vs. time seems to approach 0 quicker, again due to the nature of the quartic terms for ambient and particle temperature.

3.3 Firework Dynamics

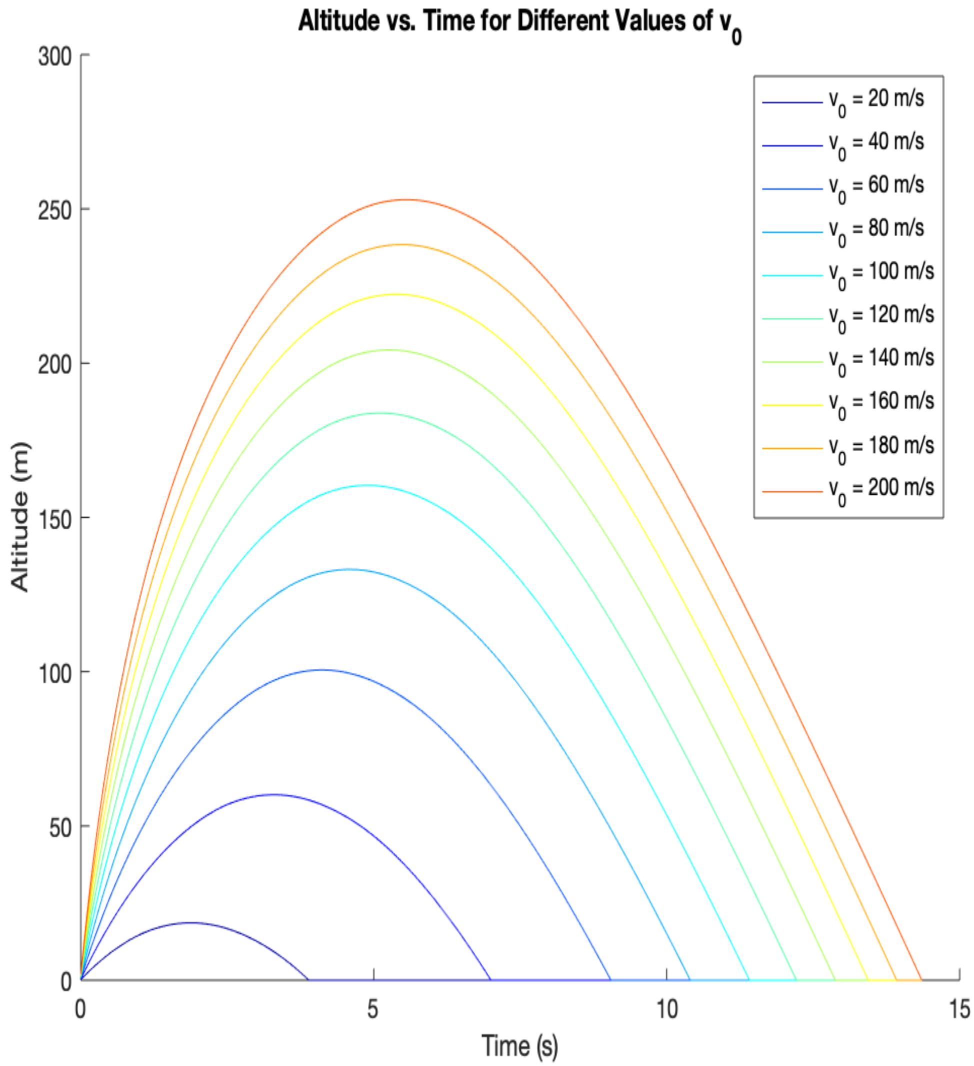


Figure 4: Firework Altitude vs. Time for different values of v_0

In the above figure, we plot the firework's altitude as a function of time with different values of v_0 at a fixed mass without an explosion occurring. The two forces acting on the firework are drag and gravity. Since the plot is of altitude, we are concerned only with movement in the $\boldsymbol{e_3}$ direction. The graph illustrates the quadratic relationship between altitude and time. Unsurprisingly, higher values of v_0 are consistent with higher (or equivalent, in the trivial cases) altitude at all points in time. Thus, a greater v_0 will cause the firework to spend more time in the air before falling back to the ground, and max altitude attained will also be greater

3.4 Genetic Algorithm

For my genetic algorithm, the parameters I chose were 60 strings and 100 Generations. I retained 12 parents and produced 12 offspring, generating 36 random strings per generation after the first generation. I used mutations to minimise inbreeding and to help escape local minima my strings may get stuck in. To do so, I initialised my ϕ vector as random numbers between -0.5 and 1.5, so that the mean 0.5. If at any point my parameters for a given string became negative, which isn't physically possible, I set the corresponding values to 0 instead.

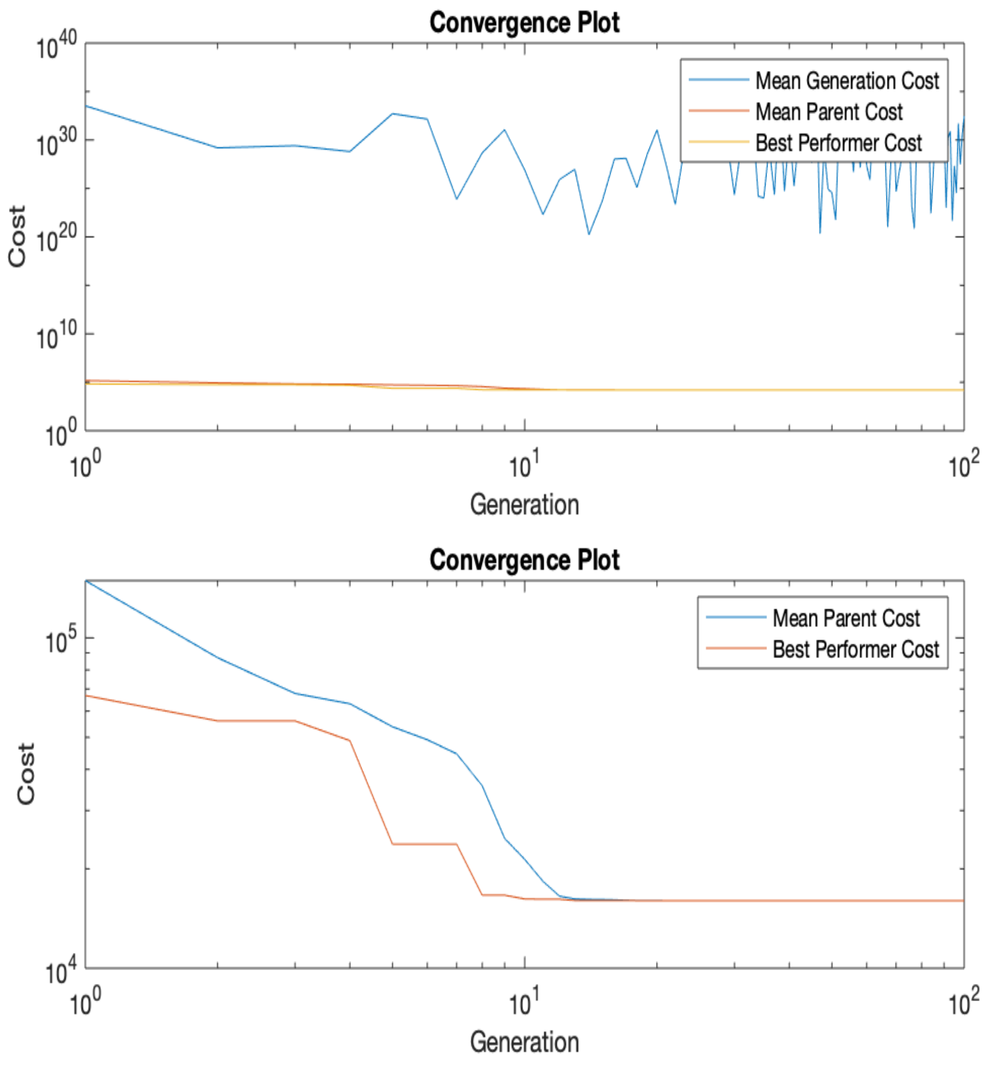
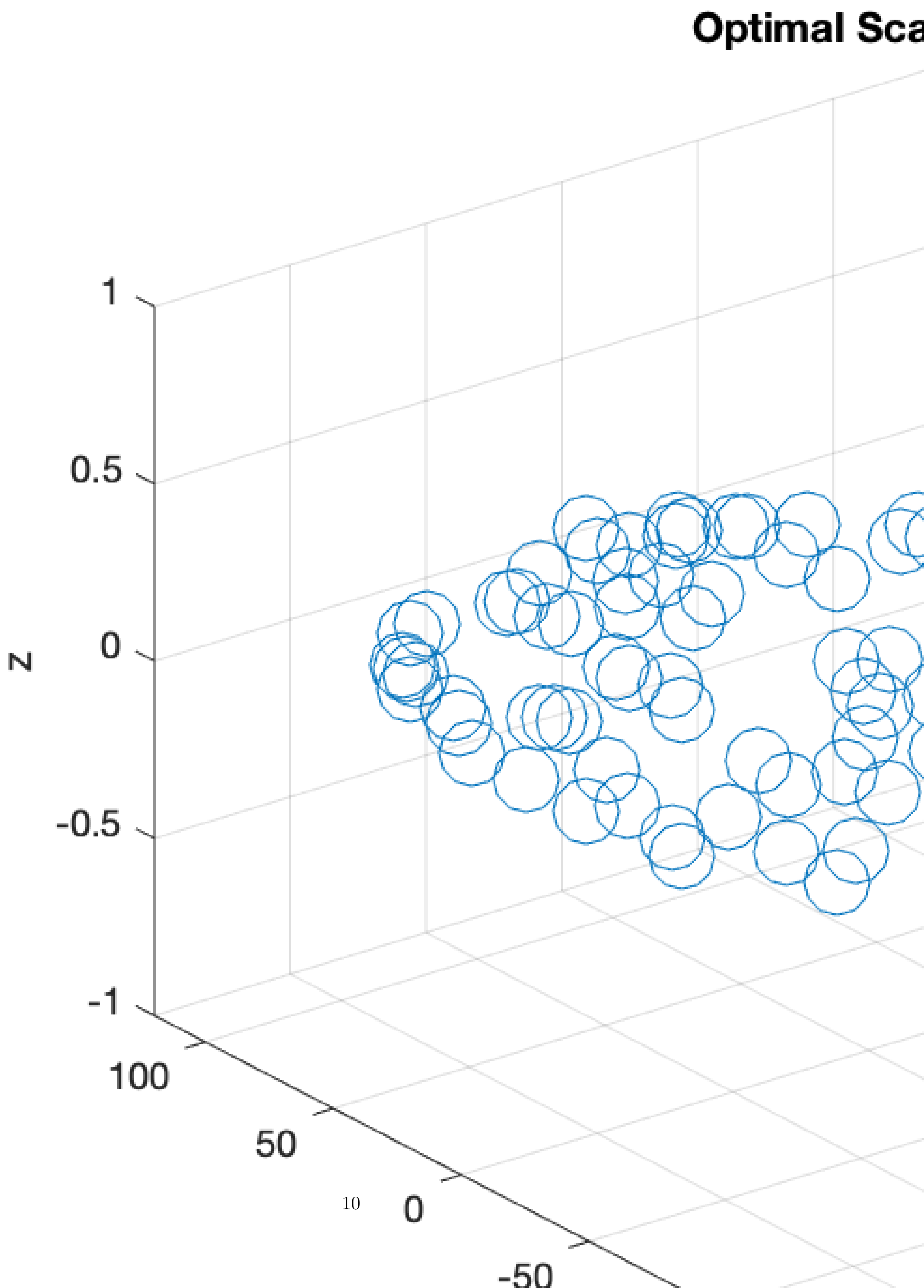


Figure 5: Convergence Plots on log-log scale, with and without Mean Generation Cost

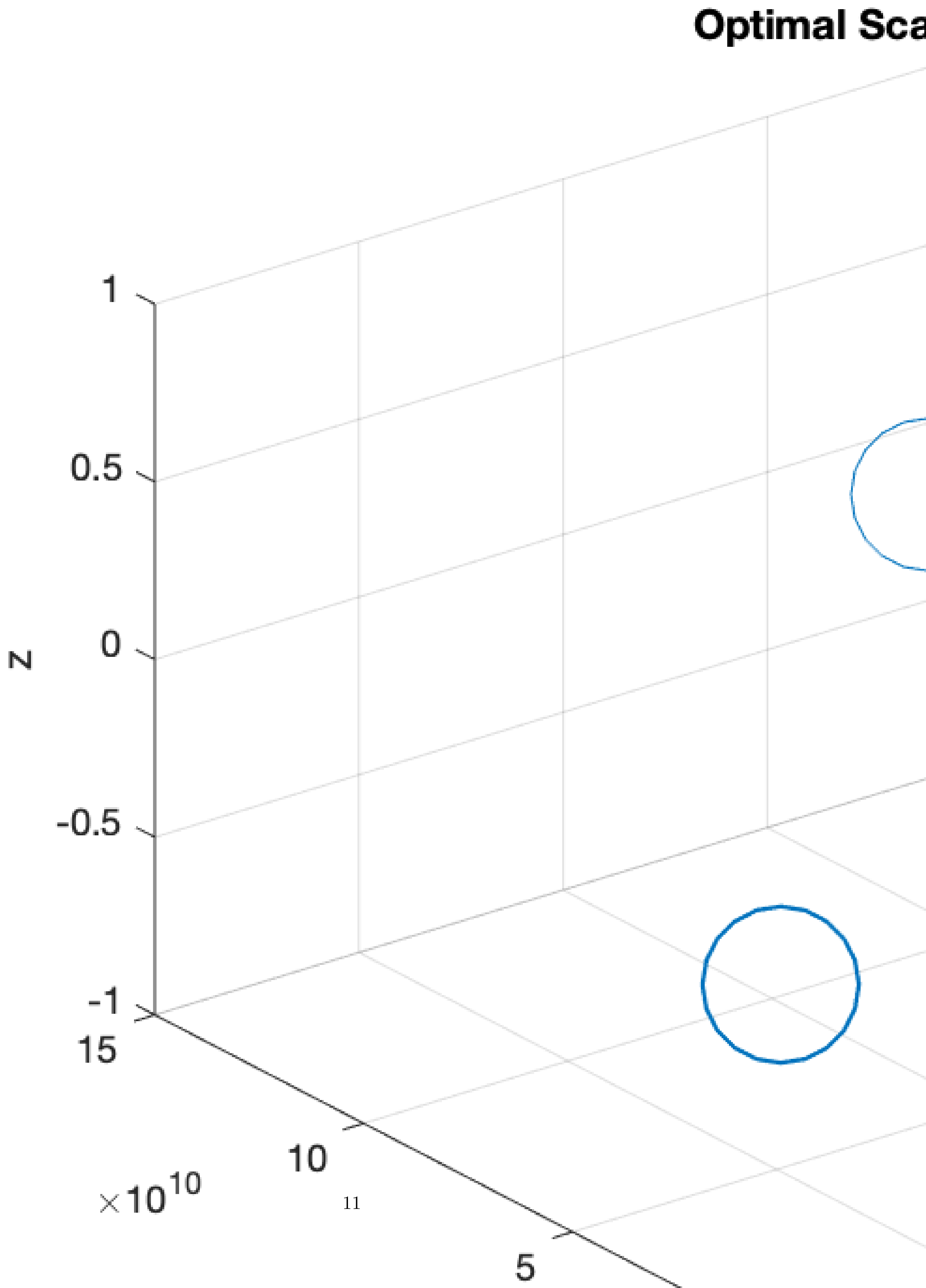
I believe my mean generation cost is so high because of numerical instability issues, which I tried controlling for, but may have been unable to do. I did use nanmean to exclude any possible NaNs that may have arisen in my calculations, but that may not have been sufficient. That being said, the mean parents and best performing genetic strings converged to a reasonable cost, given that the relatively high weights and number of terms in our cost function would yield a cost expected to be in the thousands, especially because in a lot of cases we were squaring the difference between our desired goal attributes and actual attributes

String Number	m_e	m_f	t_e	Cost
1	0.4102	4.918	5.4221	16013
2	0.4102	4.918	5.4221	16013
3	0.4102	4.918	5.4221	16013

m_e affects the cost function because greater values of m_e result in greater charge cost and incur a greater penalty on the first term. Additionally, if m_e exceeds the max value of 1 kg, then the fifth penalty term is activated, adding to the cost. m_f also adds to the charge cost in the first penalty time, and increases the cost through that term. t_e indirectly affects the cost function. Looking at the third term we see that it pertains to the height of the firework when the explosion occurs. The time of explosion is related to this, and thus t_e determined when the firework explodes, which affects the height at which it explodes, and the greater the difference in actual and desired height, the greater the cost. Firework height at detonation was 510.4181 m, which closely matches desired height of 500 m. Max temp of particle when it reached the ground was 311.9838 K, also close to desired value



Optimal Sca
1
0.5
0
-0.5
-1
z
100
50
0
-50


Optimal Sca
1
0.5
0
-0.5
-1
z
15
10
5
×10^10