Basement Bomb Shelter Feasibility in the Civil Engineering Building

An Analysis of Structural Integrity and Safety Measures

Preface

This project marks the culmination of my Bachelor of Civil Engineering at Delft University of Technology.

This report is intended for readers with a foundational understanding of physics and mathematics in the field of Civil Engineering, particularly in structure analysis and safety measures. It assumes familiarity with concepts such as load bearing, dynamic response simulation and material properties.

The feasibility of using the Civil Engineering building as a bomb shelter was thoroughly examined in this report, it starts with an introduction of the problem at hand and methodology, followed by ammunition impact, then an analytical and a numerical simulation of spring-mass system for testing accuracy of models, then a dynamic response simulation is done. This report ends with result discussion, recommendations for futures research and appendices for simulation code.

This project would not have been possible without the guidance of the supervisors/ assessors Dr. ir. P.C.J. (Pierre) Hoogenboom and Prof. dr. H.M. (Henk) Jonkers.

Delf, June 2024

Feras Saab

Summery

Using the basement of the Civil Engineering building at TU Delft as a bomb shelter is investigated in this Bachelor end project. After the increased geopolitical risks and war knocking at the doorstep of Europe following the latest war between Russia and Ukraine, research to the safety of building under attack scenario is paramount.

This study focuses on assessing whether the Civil Engineering building basement at TU Delft is feasible to serve as a bomb shelter, the assessment considers a scenario where the above-ground structure of the building is collapsing due to ammunition impact on the first-floor columns. In this case a substantial force will be exerted on the basement roof causing it to vibrate in response to the sudden load. The main research question addressed in this report is: Is the basement of the Civil Engineering Building a safe place to take shelter if a ground floor column is blown away by a bomb?

First begin the search for the destructive capability of modern warfare ammunition, especially the highly used high-explosive (HE) mortar rounds with a 120mm caliber and an explosive weight of approximately 2 kg of TNT. and its effect on impact on reinforced concrete columns was estimated. The chapter estimates that a force of 523 MN will be released from such rounds. Comparing it to compressive strength of a concrete columns, which is calculated to withstand around 14.85 MN, gave a clear conclusion these type of ammunition are more than capable to blast the Civil engineering building columns and to start the collapse events.

To create a spring-mass model that accurately simulates the dynamic response of the basement roof using numerical methods, it is essential to first develop a standard spring-mass system with fixed conditions that can be solved analytically. By comparing the numerical results with the analytical solutions, the exact error of the numerical spring-mass system can be calculated. This error assessment is crucial for ensuring the accuracy and reliability of the model when the basement roof is subsequently analyzed. It begins with harmonic motion, where a mass on a spring oscillates due to the restoring force described by Hooke's Law. The motion is modeled by the equation $m \cdot \frac{d^2x}{dt^2} + kx = 0$. The chapter examines the impact of damping, leading to underdamped, critically damped, and overdamped behaviors. It also investigates forced oscillations, highlighting resonance when the driving frequency matches the system's natural frequency, causing large oscillations.

In the next chapter, all three spring-mass systems were modeled using Euler's forward numerical method to calculate position and velocity over time. Euler's method approximates equations by discretizing time into small steps, updating position and velocity iteratively. Comparisons between numerical and analytical solutions confirm that the numerical method's mean error is under 1%, even for complex systems such as those with forced oscillations. Modeling is done using Python programing.

Chapter 4 details a numerical simulation of the dynamic response of a concrete basement roof subjected to collapse forces. The simulation models the slab as a spring-mass-damper system. The shelter's dimensions and material properties, such as column width, slab thickness, and

concrete density, are specified. The study concludes that the basement of the Civil Engineering Building at TU Delft is currently not safe to use as a bomb shelter. To enhance its structural integrity, recommendations include adding additional columns, using stronger materials, conducting detailed dynamic load analyses, incorporating design redundancy, and adhering to updated engineering standards.

Further research should focus on improving the building's design to withstand extreme impacts and exploring advanced materials that offer better resilience. Additionally, enhancing simulation models to include various collapse scenarios can provide a more comprehensive understanding of potential failures and mitigation strategies.

Contents

Introduction

Due to increasing geopolitical risks and the possibility of war breaking out in the Netherlands, ensuring safety in emergency scenarios is paramount. The former commander of the Dutch Land Force, Mark de Kruif, has warned that the Russian military might be capable of attacking NATO within two years. (De Kruif, 2024)

If war occurs around TU Delft, underground shelters are essential for protection. Several large buildings on the TU Delft campus have basements that could serve as shelters. However, their structural integrity under modern warfare conditions is crucial. This study investigates the potential collapse of a building due to bomb impacts or explosions and the impact on those seeking shelter in the basement.

In addition to this research, Luc de Wil is examining the impact of a grenade on the ground floor of the same building Together, these complementary studies aim to provide a comprehensive overview of the risks associated with taking shelter in the basement of the Civil Engineering Building at TU Delft (L.A. De Wild, A study on the use of the basement of the CEG building as a bomb shelter, Bachelor end project, Delft University of Technology, 2024).

Main Research Question

Is the basement of the Civil Engineering Building a safe place to take shelter if a ground floor column is blown away by a bomb?

Sub-questions

1. What ammunition can blow away the column of a building?
2. What are the structural specifications and dimensions of the Civil Engineering and Geosciences (CEG) building, particularly the ground floor?
3. What load will then be placed on the basement roof?
4. Will the basement roof collapse under this load?

This report is structured as follows: It begins with an analysis of modern warfare ammunition, specifically high-explosive (HE) mortar rounds, and their capability to destroy reinforced concrete columns. Following this, the study develops a theoretical framework for a spring-mass system to model the dynamic response of the basement roof under collapse forces. The numerical simulations are performed using Euler's forward method, validated against analytical solutions to ensure accuracy. The results are then applied to the specific case of the Civil Engineering Building's basement roof, assessing its structural integrity under the simulated collapse scenario. The study concludes with recommendations for enhancing the structural resilience of the shelter.

Methodology

- **Literature Review:** Examine scholarly articles, military publications, and technical reports to understand the types of munitions and their power output.

- **Structural Measurements:** Conduct on-site measurements and structural assessments to determine the building's load-bearing capacity and potential weak points.

- **Simulation and Calculations:** Use Python programming to model the effects of bomb impacts on the ground floor and assess the potential for collapse.

- **Extrapolation and Risk Scenarios:** Project different collapse scenarios and evaluate their impact on the safety of the basement shelters. Examine how debris and structural failure could affect those in the basement.

- **Capacity and Amenities Assessment:** Determine the capacity of the basement shelter and assess available amenities (such as toilets, ventilation, and emergency exit) for safety and comfort during extended sheltering.

1 Ammunition Capable of blasting a Column of a Building

In modern warfare, certain types of munitions are specifically designed to penetrate and destroy reinforced structures such as concrete columns. Understanding the types of ammunition that can cause significant structural damage is crucial for assessing the safety and integrity of buildings under potential attack. This chapter will explore high-explosive mortar rounds, calculate the force they produce, and evaluate their capability to blow away concrete columns.

1.1 High-Explosive Mortar Rounds

High-explosive (HE) mortar rounds are commonly used in military operations due to their destructive capability and effective fragmentation pattern. These rounds are designed to cause maximum damage upon impact, making them a likely candidate for blowing away structural columns.

Specifications of HE Mortar Rounds:

- Caliber: 120mm

- Explosive Weight: Approximately 2 kg of TNT equivalent

- Effective Firing Range: Up to 7,240 meters

- Lethality Radius: Approximately 69 meters (Nammo, 2023)

1.2 Calculation of Explosive Force

The explosive force generated by a mortar round can be estimated using the energy released by the TNT equivalent. One kilogram of TNT releases approximately 4.184 megajoules (MJ) of energy. (*Tons (Explosives) to Gigajoules Conversion Calculator*, n.d.)

For a 120mm mortar round with 2 kg of TNT: Total Energy Released $= 2$ kg $\times 4.184$ MJ/kg $= 8.368$ MJ

To convert this energy into force, we consider the impulse-momentum principle. The impulse is the product of the force and the time over which the force acts. The explosion causes the concrete column to shatter in a very short duration, 8 milliseconds to be exact. (Pääkkönen, 1991)

The force can be calculated as follows:

$$\text{Impulse} = \text{Force} \times \text{Time duration} \quad (1.1)$$

Given:

$$\text{Force} = \frac{\text{Total Energy Released}}{\text{Explosion Duration}} = \frac{8.368\ \text{MJ}}{8 * 10^{-3}\ \text{s}} = 1046\ \text{MN} \quad (1.2)$$

However, it is important to note that not all of this force will act in the axial direction of the column. A more realistic assumption is that only a fraction of the force contributes to axial loading, depending on the angle of impact and the distribution of explosive energy. For instance, if we assume that 50% of the force acts axially, the effective force becomes: $\text{Effective Force} = 0.5 \times 1046\ \text{MN} = 523\text{MN}$

1.3 Structural Damage Assessment

Concrete columns are designed to withstand compressive forces but are vulnerable to high-impact explosive forces. The strength of a concrete column depends on its dimensions, reinforcement, and the quality of the concrete. To evaluate whether a 120mm HE mortar round can blow away a concrete column, we need to compare the explosive force with the column's load-bearing capacity.

Concrete Column Specifications:

- Column Width: 0.55 meters - Concrete Strength (f_ck): 30 MPa - Reinforcement: Standard steel bars

Compressive Strength Calculation:

The compressive force (F_c) that a concrete column can withstand is given by:

$$F_c = \text{Area} \times \text{Concrete Strength}$$

With the columns Area $= 0.55 \times 0.90 = 0.495$ m^2 and this mean $F_c = 0.495\ \times 30 \times 10^6$ N/m^2 $= 14.85 \times 10^6$ N (this assume a constant strength thus a varying spring constant k is not taken into consideration to accurately reflect changes in stiffness. When the reinforcement yields, the tangent stiffness is effectively reduced to zero. This leads to overestimation of the strength of the columns, but in this case, it does not affect the conclusion)

The compressive strength of the column is approximately 14.85 MN. When compared to the explosive force of 523 MN, it is evident that a single HE mortars round will completely blow away a well-reinforced concrete column.

2 Mass-Spring System (Analytical solution)

Mechanical vibrations play a significant role in many branches of physics and engineering, particularly in systems that involve oscillatory motion. A classic example of such a system is a mass moving back and forth on a spring, often referred to as harmonic motion. This chapter explores the fundamental principles of harmonic motion, focusing on a simple mass-spring system. And how it relates to subject matter studied in this research project.

2.1 Mass-Spring System

Consider a system consisting of a mass m resting on a frictionless surface. The mass is attached to a spring, which exerts a force proportional to its displacement from the equilibrium position. The equilibrium position is where the spring is neither stretched nor compressed, representing the state of rest for the system. The primary question to address is how the system behaves when the mass is displaced from this equilibrium position.

2.1.1 Force Analysis and Hooke's Law

The force exerted by the spring on the mass is governed by Hooke's Law, which states that the force is proportional to the displacement from the equilibrium position. Mathematically, this force can be described as: (The Editors of Encyclopaedia Britannica, 1998)

$$F = -kx \tag{2.1}$$

where k is the spring constant, a measure of the spring's stiffness, and x is the displacement from the equilibrium position. The negative sign indicates that the force exerted by the spring is always directed towards the equilibrium position, thus providing a restoring force.

2.1.2 Newton's Second Law and Differential Equations

To understand the dynamics of the mass-spring system, we turn to Newton's second law, which states that the sum of the forces acting on an object is equal to the mass of the object multiplied by its acceleration. In this context, acceleration can be written as the second derivative of displacement x with respect to time t. (NASA Glenn Research Center, 2023)

Given that the only force in this system is the spring force, Newton's second law can be written as:

$$m \cdot \frac{d^2x}{dt^2} = -kx \tag{2.2}$$

Rearranging, we get a second-order differential equation:

$$m \cdot \frac{d^2x}{dt^2} + kx = 0 \tag{2.3}$$

This differential equation describes the motion of the mass-spring system and can be solved to understand the behavior of the oscillations.

2.1.3 Solving the Differential Equation

To solve this differential equation, we assume a solution of the form: (Blaauwendraad, 2016)

$$x(t) = e^{rt} \tag{2.4}$$

where r is an unknown constant. By substituting this assumed solution into the differential equation, we obtain the characteristic equation: (Blaauwendraad, 2016)

$$m \cdot r^2 + k = 0 \tag{2.5}$$

Solving for r, we get:

$$r = \pm i\sqrt{\frac{k}{m}} \tag{2.6}$$

Introducing the term $\omega_0 = \sqrt{\frac{k}{m}}$, which represents the natural frequency of the system, the solution to the differential equation can be written as: (Blaauwendraad, 2016)

$$x(t) = A\,cos(w_0 t)\ + B\,cos(w_0 t) \tag{2.7}$$

where A and B are constants determined by initial conditions.

2.1.4 Combining Trigonometric Functions

To simplify the solution, we can express it in a different form using a single trigonometric function. By applying trigonometric identities, the above expression can be rewritten as:

$$x(t) = C\,cos(w_0 t\ +\ \gamma) \tag{2.8}$$

where C represents the amplitude of the oscillations, and γ is a phase shift that depends on the initial conditions. The amplitude C indicates the maximum displacement from the equilibrium position, while the natural frequency γ determines the rate of oscillation.

Graphical Interpretation and Analysis:

In figure 1, this solution represents oscillatory motion with three key parameters: amplitude, frequency, and phase shift. The amplitude C reflects the maximum range of oscillation, while the natural frequency γ determines the speed of oscillation. The phase shift γ describes the initial position of the oscillation. The behavior of the oscillations can be visualized through graphs that illustrate these parameters' effects. The Python code to plot this graph is placed in Appendix B.

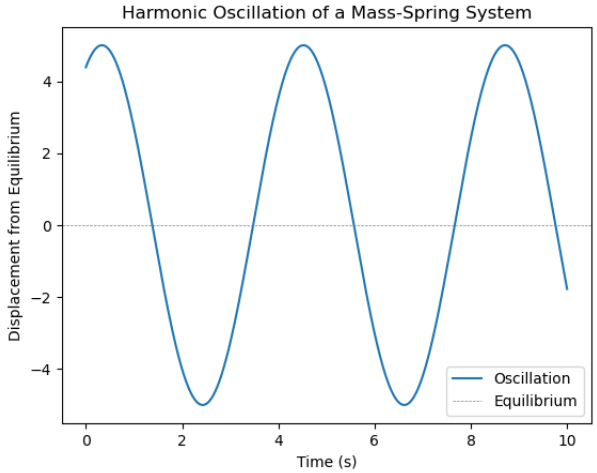

Figure 1

2.2 Friction and Damping in Mass-Spring Oscillations

In the context of a mass oscillating on a spring, the effects of friction and damping are crucial in understanding the system's behavior over time. In an ideal scenario with no friction, a mass attached to a spring will oscillate indefinitely. However, in real-world applications, friction inevitably comes into play, gradually reducing the amplitude of the oscillations until the system returns to its equilibrium position. This chapter explores how friction affects a mass-spring system and delves into the various scenarios that can arise from different levels of damping.

- **Damping Force (Friction):** Friction or damping in a system acts against the motion, proportional to the velocity of the mass. This can be modeled as: (Blaauwendraad, 2016)

$$F_f = -c \cdot \frac{dx}{dt} \tag{2.9}$$

Where (c) is the damping coefficient, indicating the level of friction, and ($\frac{dx}{dt}$) is the velocity of the mass.

- **Net Force and Newton's Law**: The total force acting on the mass is the sum of these forces, leading to Newton's second law, which states that force is equal to mass times acceleration:

$$m \cdot \frac{d^2x}{d^2t} = F_s + F_f \tag{2.10}$$

2.2.1 Differential Equation with Damping

Combining these forces, we get a second-order differential equation that describes the motion of the mass-spring system with friction:

$$m \cdot \frac{d^2x}{d^2t} + c \cdot \frac{dx}{dt} + k \cdot x = 0 \tag{2.11}$$

To solve this differential equation, we typically use an exponential form for the solution: $x(t) = e^{rt}$ Substituting this into the differential equation leads to a characteristic equation: $m \cdot r^2 + c \cdot r + k = 0$. This characteristic equation can be solved using the quadratic formula to obtain the roots:

$$r = \frac{-c \pm \sqrt{c^2 - 4mk}}{2m} = \frac{-c}{2m} \pm \frac{\sqrt{c^2 - 4mk}}{2m} \tag{2.12}$$

Dividing the r into real part and another imaginary part, for simplicity $\mathrm{i}\omega_1 = \frac{\sqrt{c^2-4mk}}{2m}$ depending on the discriminant $(c^2 - 4mk)$, the solution can yield different types of damping behaviors, each with distinct physical interpretations.

2.2.2 Types of Damping

Underdamped Case

When $(c^2 - 4mk)$ is negative, the roots are complex, indicating oscillatory behavior with damping. This is known as the underdamped case, where the system oscillates but with decreasing amplitude over time due to the frictional force. The solution can be expressed as: (Blaauwendraad, 2016)

$$x(t) = e^{-c/2m \cdot t} \cdot (A \cdot \cos(\omega_1 \cdot t) + B \cdot \sin(\omega_1 \cdot t)) \quad (2.13)$$

$$x(t) = e^{-c/2m \cdot t} \cdot (C \cdot cos(\omega_1 \cdot t - \gamma)) \quad (2.14)$$

where $(\omega_1 = \sqrt{4mk - c^2}/(2m))$represents the damped frequency. In this scenario, the oscillations gradually decrease in amplitude, with the exponential decay term causing the oscillations to die out as time progresses. This behavior is typical in mechanical systems with moderate levels of friction.

Overdamped Case

In the overdamped case, $(c^2 - 4mk)$ is positive, resulting in two distinct real roots. The system does not oscillate; instead, it exhibits exponential decay towards the equilibrium position. The general solution is: $x(t) = A \cdot e^{r_1 \cdot t} + B \cdot e^{r_2 \cdot t}$

where$(r_1)$and (r_2) are both negative, indicating the rate of exponential decay. In this case, the high level of friction prevents oscillation, and the system quickly settles to its equilibrium position without significant oscillation. (Blaauwendraad, 2016)

Critically Damped Case

The critically damped case occurs when $(c^2 - 4mk)$ is precisely zero. This condition is rare but represents the optimal level of damping where the system returns to equilibrium in the shortest time without oscillation. The solution for this case is: $x(t) = (A + B \cdot t) \cdot e^{-c/(2m) \cdot t}$

In this scenario, the system may initially move away from the equilibrium due to the linear term, but it quickly settles back to zero due to exponential decay, representing the balance between damping and oscillation. (Blaauwendraad, 2016)

Graphical Interpretation and Conclusion:

In figure 2 Graphical representations of these damping cases help visualize how each type of damping affects the system's behavior. In the underdamped case, the oscillations gradually decrease in amplitude, with an "envelope" created by the exponential decay. In the overdamped case, the system quickly returns to equilibrium without oscillation. In the critically damped case, the system has an initial oscillation but settles to equilibrium efficiently. The Python code to plot this graph is placed in Appendix B.

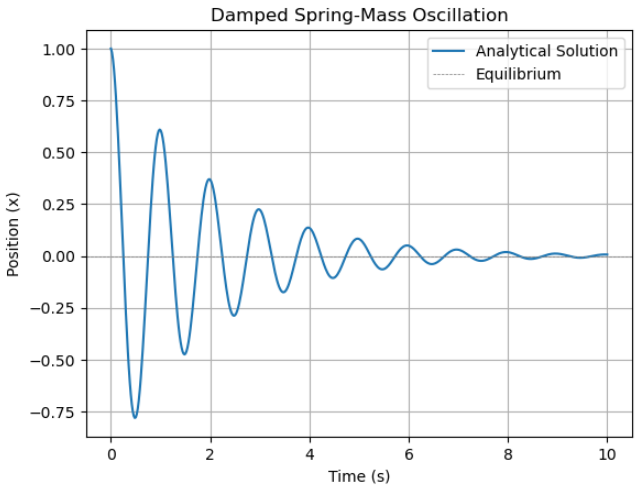

Figure 2

Understanding these different damping scenarios is crucial for designing systems that rely on controlled oscillations, such as shock absorbers in vehicles or building designs that consider seismic activity. By analyzing the behavior of damped mass-spring systems, engineers can ensure the stability and safety of various applications.

2.3 Forced Oscillations and Resonance in Damped Mass-Spring Systems

2.3.1 Forced Oscillations in Damped Systems

Consider a mass attached to a spring with a damping mechanism. When an external periodic force is applied to this system, it begins to oscillate with a frequency that matches the frequency of the driving force. The response of the system depends on the amplitude and frequency of the external force, as well as the system's natural frequency and damping characteristics.

The equation of motion for a forced mass-spring system with damping can be derived using Newton's second law. Given a mass (m), spring constant (k), damping coefficient (b), and an external force $(F\_0\ cos\ (\omega t))$, the system's motion can be described as:

$$m \cdot \ddot{x} + b \cdot \dot{x} + k \cdot x = F_0 \cdot \cos(\omega t) \qquad (2.15)$$

This second-order differential equation represents the balance between inertial, damping, and restoring forces, with the additional driving force providing external energy to the system.

2.3.2 Resonance and Natural Frequency

One of the most intriguing aspects of forced oscillations is resonance, which occurs when the driving frequency (ω) matches the system's natural frequency $(\omega\_0 = \sqrt{\frac{k}{m}})$. At resonance, even a small driving force can lead to large oscillations due to constructive interference, where energy from the driving force accumulates over time.

The amplitude of oscillations at resonance depends on the damping coefficient. Lower damping leads to higher amplitudes, while increased damping reduces the impact of resonance. This balance is critical in applications like suspension systems, bridges, and buildings, where excessive oscillations can be dangerous.

2.3.3 Types of Forced Oscillations

The behavior of a forced mass-spring system with damping can vary significantly based on the level of damping and the driving frequency. We explore the following cases:

Under-Damped Oscillations

When damping is relatively low, the system exhibits oscillatory behavior even with the presence of damping. At resonance, the oscillations reach their maximum amplitude, but they gradually decay due to the damping force. The general solution for this case includes both the natural oscillatory response and the forced response, given by:

$$x(t) = e^{-b/2m \cdot t} \cdot (A \cdot \cos(\omega_1 \cdot t) + B \cdot \sin(\omega_1 \cdot t)) + \frac{F_0/m}{\sqrt{(\omega_0^2 - \omega^2)^2 + (b \cdot \omega/m)^2}} \cdot \cos(\omega \cdot t - \phi) \quad (2.16)$$

where (ϕ) is the phase shift resulting from damping and the driving frequency, and $(\omega_1 = \sqrt{\omega_0^2 - (b/2m)^2})$. The under-damped scenario is characterized by sustained oscillations with gradually decaying amplitude.

Critically Damped Oscillations

Critical damping represents the optimal level of damping where the system returns to equilibrium without oscillation but in the shortest time possible. This condition occurs when $(b^2 = 4 \cdot m \cdot k)$. The solution in this case is:

$$x(t) = (A + B \cdot t) \cdot e^{-b/2m \cdot t} \quad (2.17)$$

In this scenario, the system experiences a smooth return to equilibrium, making it ideal for applications where oscillations must be minimized, such as in precision engineering.

Over-Damped Oscillations

When the damping is higher than the critical level, the system does not oscillate but slowly returns to equilibrium. This behavior occurs when $(b^2 > 4 \cdot m \cdot k)$. The general solution is:

$$x(t) = A \cdot e^{r_1 \cdot t} + B \cdot e^{r_2 \cdot t} \quad (2.18)$$

where (r_1) and (r_2) are the roots of the characteristic equation. This situation is typical in systems requiring stability without oscillation, such as heavy machinery or high-speed transportation.

Graphical Interpretation and Resonance Phenomena:

In figure 3 Graphs illustrating forced oscillations reveal the complex dynamics at play. For example, the phase space diagram, which plots position against velocity, demonstrates the oscillatory patterns in under-damped systems and the smooth transitions in critically damped and over-damped cases. Resonance phenomena can also be visualized through amplitude vs. driving frequency plots, indicating the sharp increase in amplitude as the driving frequency approaches the system's natural frequency. This behavior underscores the importance of carefully controlling the driving frequency and damping in engineering applications to avoid undesirable resonance effects. The Python code to plot this graph is placed in Appendix B.

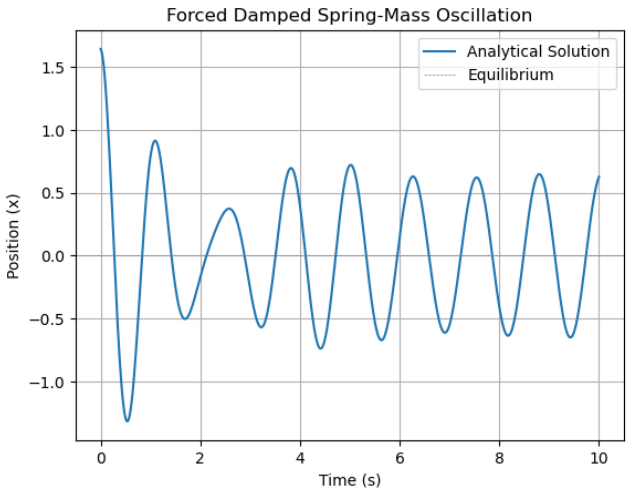

Figure 3

3 Mass-Spring System (Numerical solution)

In this project, we implemented a numerical simulation of a spring-mass system to compute the position and speed of the mass over time. The simulation is based on Euler's method, a straightforward approach to solving ordinary differential equations. This report explains how Euler's method works, its limitations, and how we mitigated those limitations through an appropriate choice of time step.

3.1 Spring-Mass System without damper

A spring-mass system consists of a mass attached to a spring that obeys Hooke's Law, which states that the force exerted by the spring is proportional to the displacement from its equilibrium position. The system experiences oscillatory motion due to this restoring force. Newton's second law governs the dynamics of the system, leading to the following differential equations:

1. Velocity equation:

$$\frac{dx}{dt} = v \tag{3.1}$$

2. Acceleration equation:

$$\frac{dv}{dt} = g - \frac{k \cdot x}{m} \tag{3.2}$$

where:

- (x) is the position, - (v) is the velocity,

- (g) is the gravitational constant (9.81 m/s²), - (k) is the spring constant (40 N/m),

- (m) is the mass (1 kg).

3.1.1 Euler Forward Method

Euler's method approximates the solution of these differential equations by discretizing time into small steps, (Δt). Given the current values of position and velocity, it calculates the new values after a small-time increment. The following formulas outline how Euler's method computes these updates: (Libretexts, 2022)

1. Update position:

$$x_{t+\Delta t} = x_t + \Delta t \times v_t \tag{3.3}$$

2. Update velocity:

$$v_{t+\Delta t} = v_t + \Delta t \times \left(g - \frac{k \cdot x_t}{m}\right) \tag{3.4}$$

This approach is applied iteratively to generate a sequence of position and velocity values over time. The simulation computes these values over a predefined time span, using a set of initial conditions and a specified time step, (Δt).

3.1.2 Limitations of Euler Forward and How to Overcome Them

While Euler's method is straightforward and easy to implement, it has some notable limitations:

- **Stability:** The method can become unstable for large time steps, leading to erroneous results. In the case of oscillatory systems like the spring-mass system, large time steps can result in growing oscillations that don't reflect the physical reality.

- **Accuracy:** Euler's method is a first-order approximation, which can lead to errors accumulating over time, especially with larger time steps. (*Forward and Backward Euler Methods*, n.d.

To overcome these limitations, we used a small-time step, (Δt), to ensure stability and accuracy. Smaller time steps yield more accurate results, although they increase computational cost due to the need for more iterations. In this simulation, a time step of 10^5 time points over a 5-second span.

3.1.3 Results and Analysis

In appendix A python code is places that have been used to calculate displacement of this spring mass system using Euler forward and the result are shown in figures 4 and 5. The position and velocity of the spring-mass system were calculated using the Euler forward method, and the results were visualized through plots. The oscillations in position and velocity,

as well as the phase diagram (plotting velocity against position), provided insights into the behavior of the system.

By employing a small-time step, we achieved a stable simulation with a good representation of the oscillatory motion characteristic of the spring-mass system.

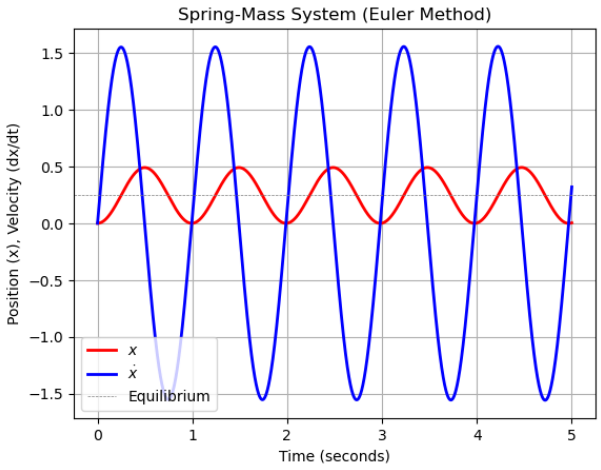

Figure 5

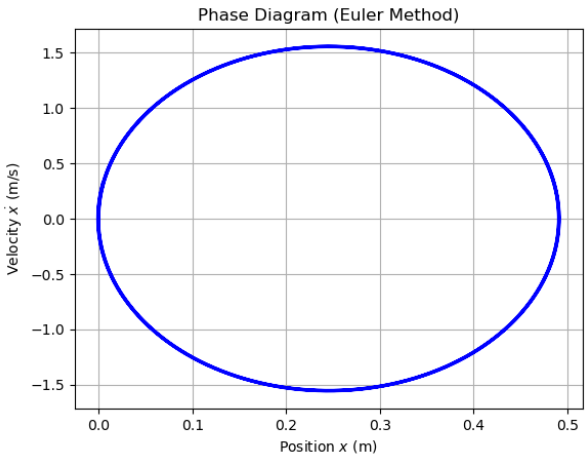

Figure 4

3.2 Spring-Mass system with damper

Let's simulate a spring-mass-damper system, a mechanical system characterized by a mass attached to a spring with a damping element. The damping term introduces resistance to the system's motion, affecting the oscillation's amplitude and rate of decay. This report outlines the simulation process using Euler's forward method and discusses the importance of a fine time step for stability and accuracy.

Where a new term is added

- (b): Damping coefficient, 1 Ns/m.

3.2.1 Euler Forward Method and Damping

Euler's forward method computes the system's behavior by incrementally updating the position and velocity over time. The method is applied iteratively, with updates determined by the current velocity and the calculated acceleration. The inclusion of a damping term alters the acceleration equation, introducing a force that opposes motion, thus reducing oscillation amplitude over time.

3.2.1.1 Euler Forward Algorithm

1. Update Position:

The new position is derived from the current velocity:

$$x[i] = x[i-1] + h \times x_{dot}[i-1] \tag{3.5}$$

2. Calculate Acceleratio*:

The acceleration now accounts for the gravitational force, the spring force (restoring force), and the damping force:

$$\text{acceleration} = g - \frac{k \cdot x[i-1]}{m} - \frac{b \cdot x_{dot}[i-1]}{m} \tag{3.6}$$

3. Update Velocity:

The new velocity is updated based on the calculated acceleration: $x_{dot}[i] = x_{dot}[i-1] + h \times \text{acceleration}$

3.2.2 Limitations and Importance of a Fine Time Step

Euler's forward method can be prone to stability issues and inaccuracies, especially with large time steps. The inclusion of damping increases stability, but the accuracy can still be compromised if the time step is too large. By using a fine time step (10^5), the simulation achieves a stable and accurate representation of the spring-mass-damper system's behavior.

Overall, this simulation demonstrates how Euler's forward method can be used to simulate a spring-mass-damper system, with appropriate time step choices to maintain stability and capture the system's oscillatory behavior. In appendix A python code is places that have been used to calculate displacement of this spring mass system using Euler forward and the result are shown in figures 6 and 7.

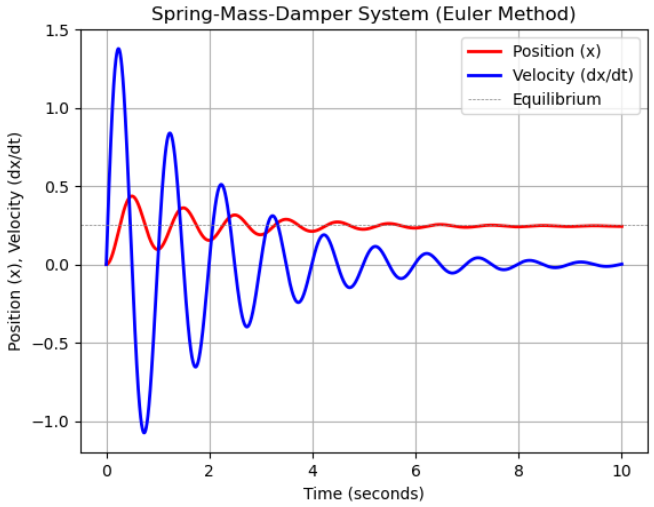

Figure 7

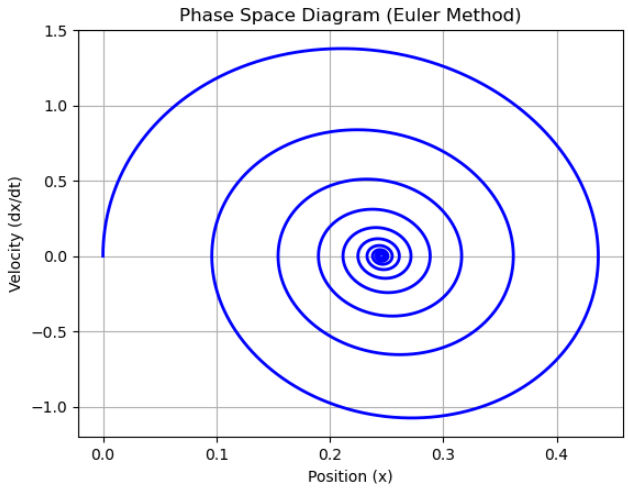

Figure 6

3.3 Spring-Mass system with driving force

3.3.1 Added constants and Initial Conditions

- F_0 : Amplitude of the driving force (10 N).
- w : Angular frequency of the driving force (1.5 rad/s).

3.3.2 Euler Forward Method with Damping and Driving Force

Euler's forward method is used to compute the position and velocity iteratively over time. This method is well-suited for simple simulations, though it can be less accurate with larger time steps. The inclusion of damping and driving force modifies the acceleration equation, introducing terms that represent these additional forces.

3.3.3 Euler Forward Algorithm

1. Update Position:

The position is updated using the current velocity:

$$x[i] = x[i-1] + h \times x_{dot}[i-1] \tag{3.8}$$

2. Compute Acceleration:

The acceleration includes the gravitational force, damping force, spring restoring force, and driving force:

$$\text{acceleration} = g - \frac{b \cdot x_{dot}[i-1]}{m} + \frac{F_0 \cdot \cos(w \times t[i-1])}{m} - \frac{k \cdot x[i-1]}{m} \tag{3.9}$$

3. Update Velocity:

The new velocity is calculated based on the acceleration: $x_{dot}[i] = x_{dot}[i-1] + h \times \text{acceleration}$

In figure 9 the displacement and velocity diagram of this system is shown, and in figure 8 the phase diagram is also shown. An observation can be made that after the initial displacement the spring mass system will find a new oscillatory equilibrium that is caused by the constant driving force. In appendix A the python code used to calculate and plot the graphs is shown.

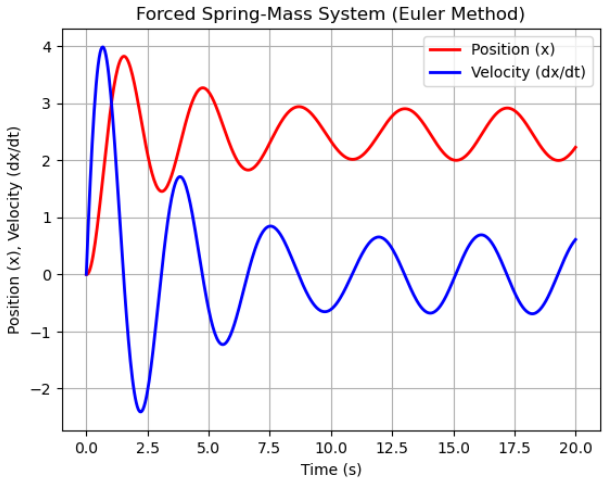

Figure 9

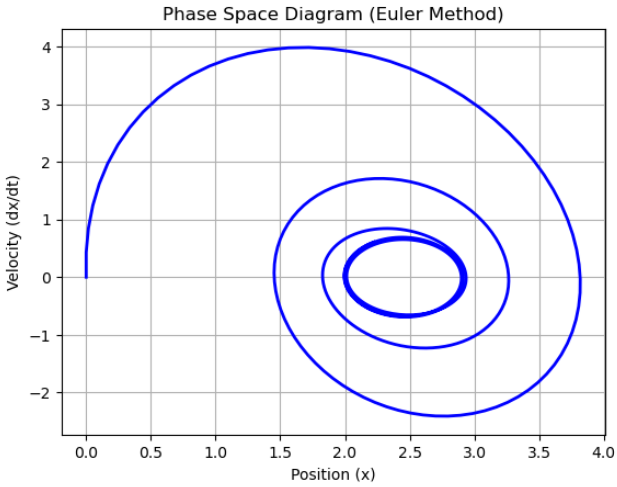

Figure 8

3.4 Accuracy Testing of the Numerical Method

To verify the accuracy of the numerical method employed in this study, a comparison between the numerical and analytical solutions for a forced damped mass-spring system was performed. The analytical solution incorporates both the transient and steady-state responses, providing a comprehensive benchmark against which the numerical results can be evaluated.

The results of the numerical and analytical solutions were compared by plotting the position and velocity over time. The absolute errors and percentage errors were calculated to quantify the accuracy of the numerical method.

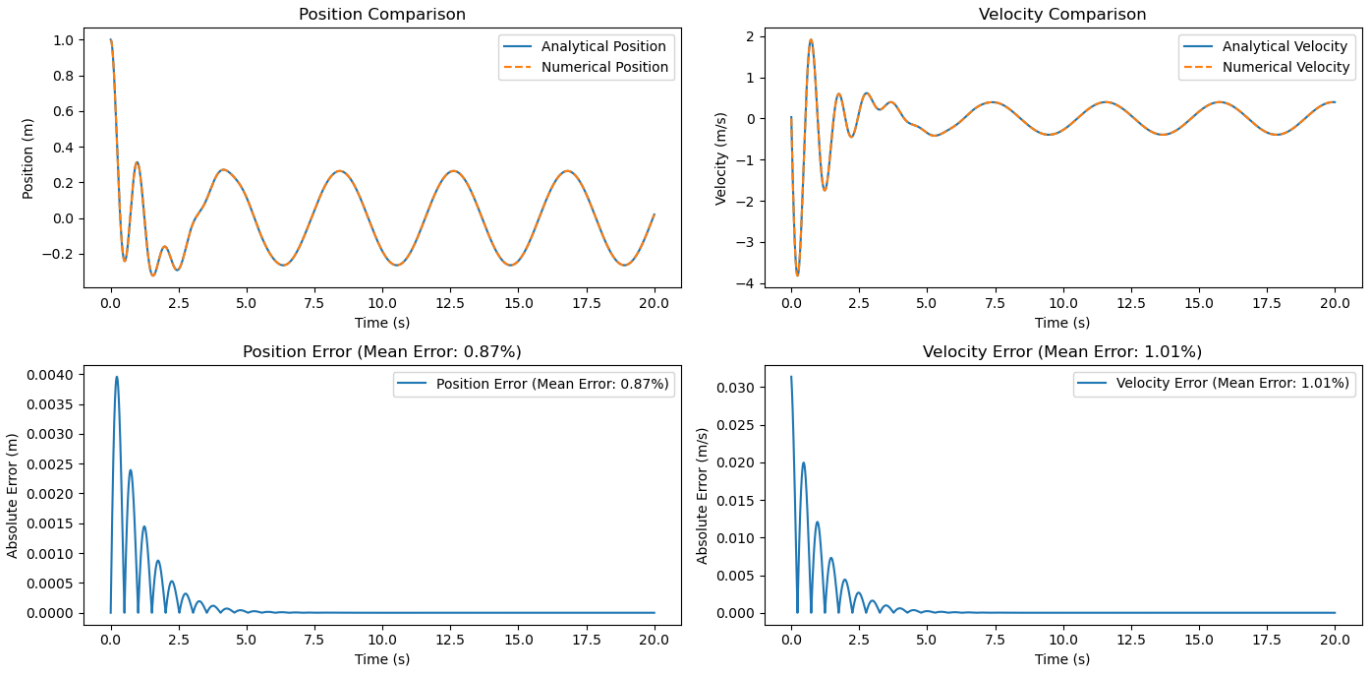

Figure 10 shows the comparison of the position and velocity between the analytical and numerical solutions. The close agreement between the two indicates the accuracy of the numerical method. presents the absolute errors in position and velocity, highlighting the transient behaviour and the steady-state accuracy.

- Mean Position Error Percentage: 0.87%

- Mean Velocity Error Percentage: 1.01%

In Appendix C the python code used to calculate the mean error is shown. These results demonstrate that the numerical method provides a highly accurate approximation of the analytical solution, validating its use for the forced oscillations in a damped mass-spring system.

4 Dynamic Response of a Concrete Roof Slab Under Collapse Forces

This chapter presents a numerical simulation of the dynamic response of a concrete roof slab in an underground shelter subjected to collapse forces. The simulation models the slab as a spring-mass-damper system and applies forces representing the sequential collapse of building floors above the shelter. This approach helps evaluate the structural integrity and safety of the shelter under catastrophic events.

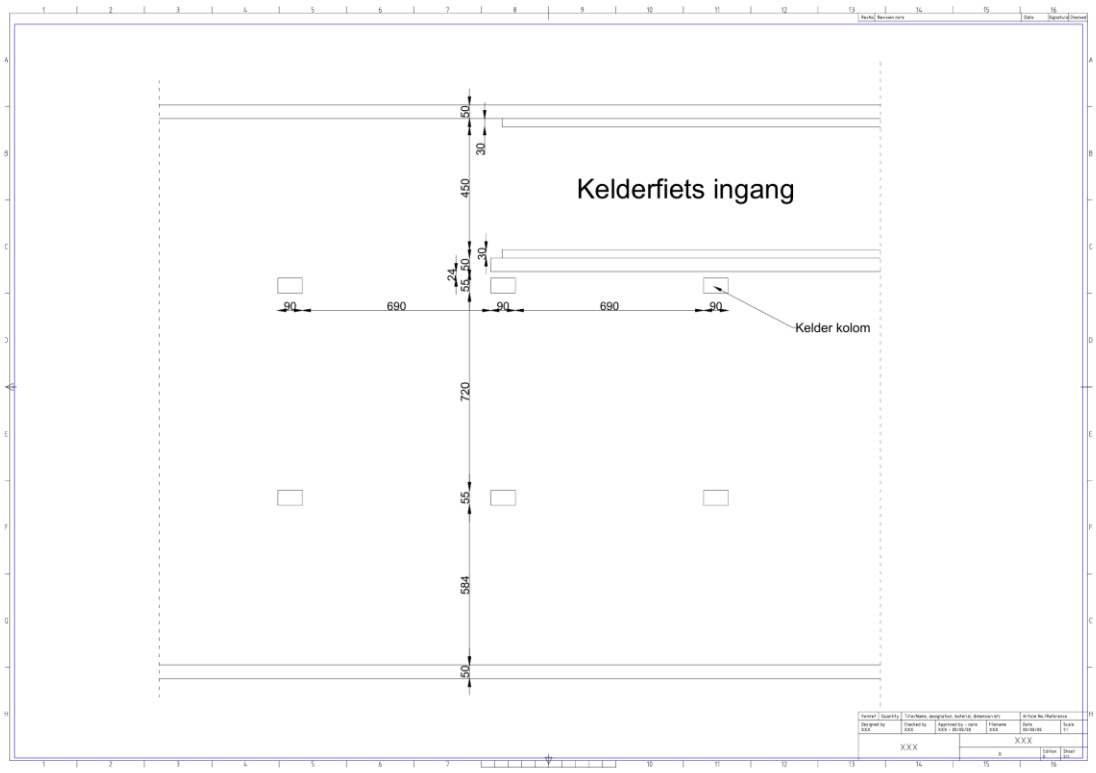

Figure 11 Overzichtstekening kelder drawn on AutoCAD

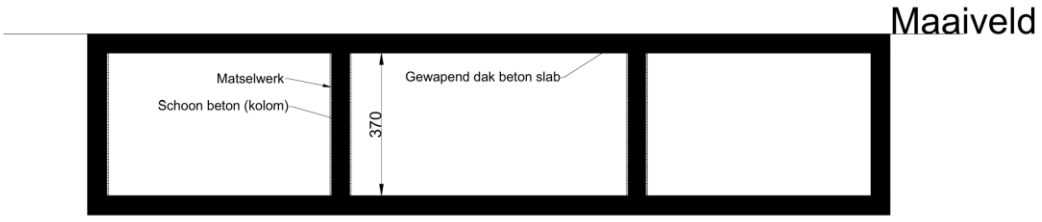

Figure 12 Doorsnede kelder drawn on AutoCAD

4.1 Shelter Dimensions and Material Properties

The underground shelter is characterized by the following dimensions and material properties:

- **Column Width:** 0.55 meters - **Distance Between Columns (h.o.h):** 7.75 meters

Distance Between Columns: 7.20 meters

- **Distance to Wall:** 6.36 meters - **Slab Thickness:** 0.5 meters

- **Concrete Density:** 2400 kg/m³ - **Modulus of Elasticity (E) for C30 Concrete:** 3 GPa

4.2 Effective Length and Moment of Inertia

The effective length of the beam, which represents the slab supported between columns, is calculated as:

$$l = 0.75 \times \text{Distance Between Columns} = 0.75 \times 7.75\,\text{m} = 5.81\,\text{m} \qquad (4.1a)$$

The moment of inertia (I) for the rectangular cross-section of the slab is given by: (Hartsuijker & Welleman, 2007)

$$I = \frac{\text{Effective Width} \times \text{Slab Thickness}^3}{12} = \frac{3\,\text{m} \times (0.5\,\text{m})^3}{12} = 0.03125\,\text{m}^4 \qquad (4.1b)$$

4.3 Spring Constant Calculation

The spring constant (k) is initially calculated for a beam supported at two points: (Hartsuijker & Welleman, 2007)

$$k = \frac{384}{5} \times \frac{E \times I}{l^4} = \frac{384}{5} \times \frac{30 \times 10^9\,\text{Pa} \times 0.03125\,\text{m}^4}{(5.81\,\text{m})^4} = 3.38 \times 10^7\,\text{N/m} \qquad (4.2a)$$

This is equation is being derived from the general spring mass equation (1.1) and from the general displacement equation on beam on two support $k = \frac{5}{384} \times \frac{l^4 * q}{E \times I}$, For a beam supported at four points, the spring constant is approximately doubled: $k_{\text{adjusted}} = 2 \times k = 6.76 \times 10^7$ N/m

The stiffness of cracked concrete is approximately 1/4 of uncracked concrete. In fact, it is a non-linear spring; when the reinforcement flows, the stiffness is zero (tangent stiffness). When unloaded, the stiffness again equals the torn stiffness. An illustrative graph was added that shows the changes in the stiffness. In practice there must be a set displacement and force on where the Elasticity modulus will change accordingly. In designing this shelter roof phase 1 will be passed, and if a design load of 3.5 kN/m^2 is taken this will lead to a displacement of

$$u = \frac{5}{385} \times \frac{q*l^4}{EI} = \frac{5}{385} \times \frac{7.35*3.5*10^3*5.81^4}{3*10^9*0.03125} = 0.004\ meter \qquad (4.2b)$$

And this will lead that the displacement at which the concrete is fully cracked is equal to 4 time the calculated displacement, equal to 0.016 meter. In Appendix D of the Python model, the varying spring constant k is taken into consideration to accurately reflect changes in stiffness. When the reinforcement yields, the tangent stiffness is effectively reduced to zero. This variation is crucial because, without it, the model would inaccurately apply a constant stiffness of 3 GPa at all times. By incorporating this variation, the model more accurately represents the dynamic response of the system under different loading conditions, ensuring a realistic simulation of the structural behavior.

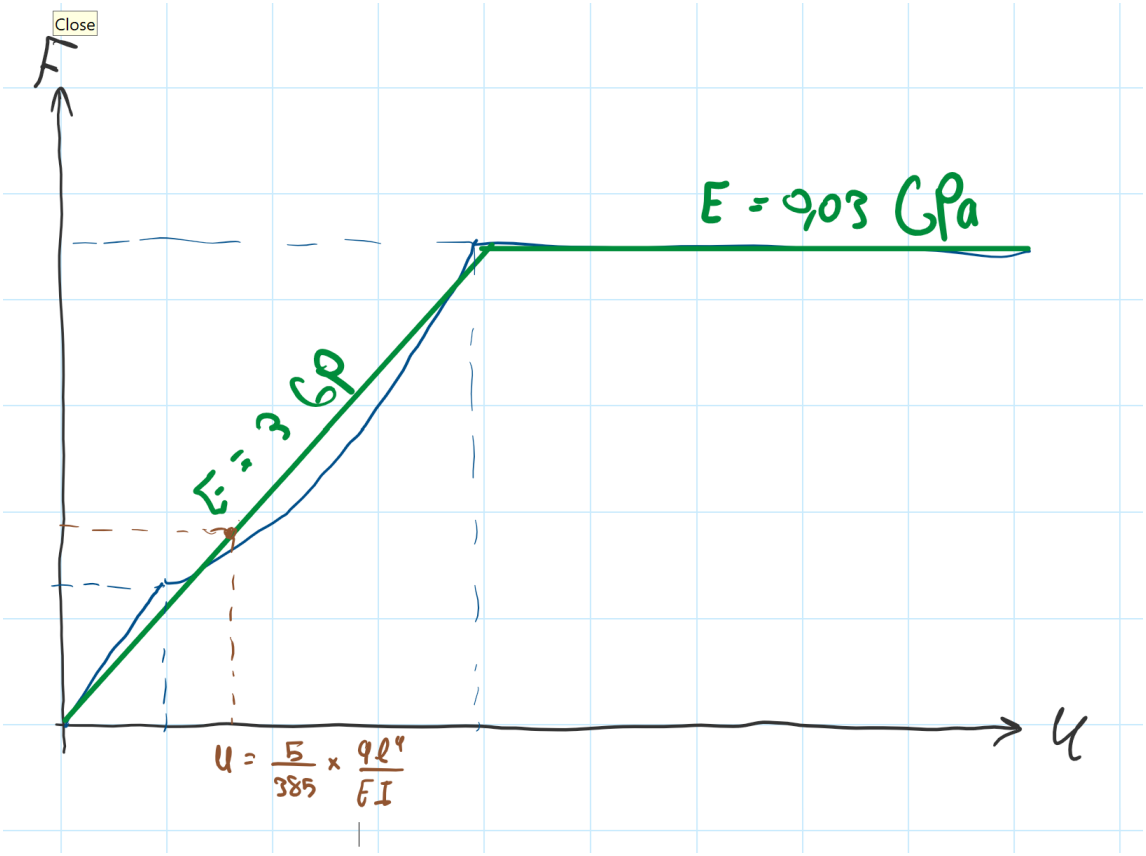

Figure 13 graph showing Elasticity modulus changing from not cracked to partly cracked to fully cracked concrete

4.4 Damping Coefficient Calculation

The damping coefficient (c) is calculated using: (*Critical Damping Coefficient*, 2022)

$$c = 2 \times \zeta \times \sqrt{k \times m} \qquad (4.3)$$

where (ζ) is the damping ratio (4%) and (m) is the mass of the slab.

Given: m = Concrete Density × Slab Thickness × Effective Length = $2400\,\text{kg/m}^3 \times 0.5\ \text{m} \times 5.81\,\text{m} = 6{,}972\,\text{kg}$

Thus: $c = 2 \times 0.04 \times \sqrt{6.76 \times 10^7\,\text{N/m} \times 6{,}972\,\text{kg}} = 238{,}144\,\text{Ns/m}$

4.5 Simulation Setup

To simulate the dynamic response of the shelter in case of a building collapse, the shelter's roof is modeled as a modified version of a spring-mass-damper system. This model is adapted to represent the concrete slab roof in the underground floor. The simulation incorporates the forces due to the building's collapse.

A 2D cross-section of the shelter is taken and represented as a beam on four support points because the shelter has two columns and two walls on the side. The effective length of the beam is modeled accordingly. The shelter's structural layout includes columns measuring 90x55 cm, with two columns 775 cm apart from heart to heart. The distance from the heart of a column to the heart of a side wall is 636 cm, with the side wall itself being 50 cm thick. The height of each floor is 420 cm. All structural components are made from C30 concrete, with the roof slab thickness being 50 cm. The same layout applies to the floors above, except there are no side walls, and the floor thickness is 50 cm.

4.5.1 Collapse Scenario and Forces

The building is divided into six levels: -1 (shelter), 0 (ground level), 1, 2, 3, 4, and 5. The collapse is modeled in sequential moments, denoted as t0, t1, t2, etc.

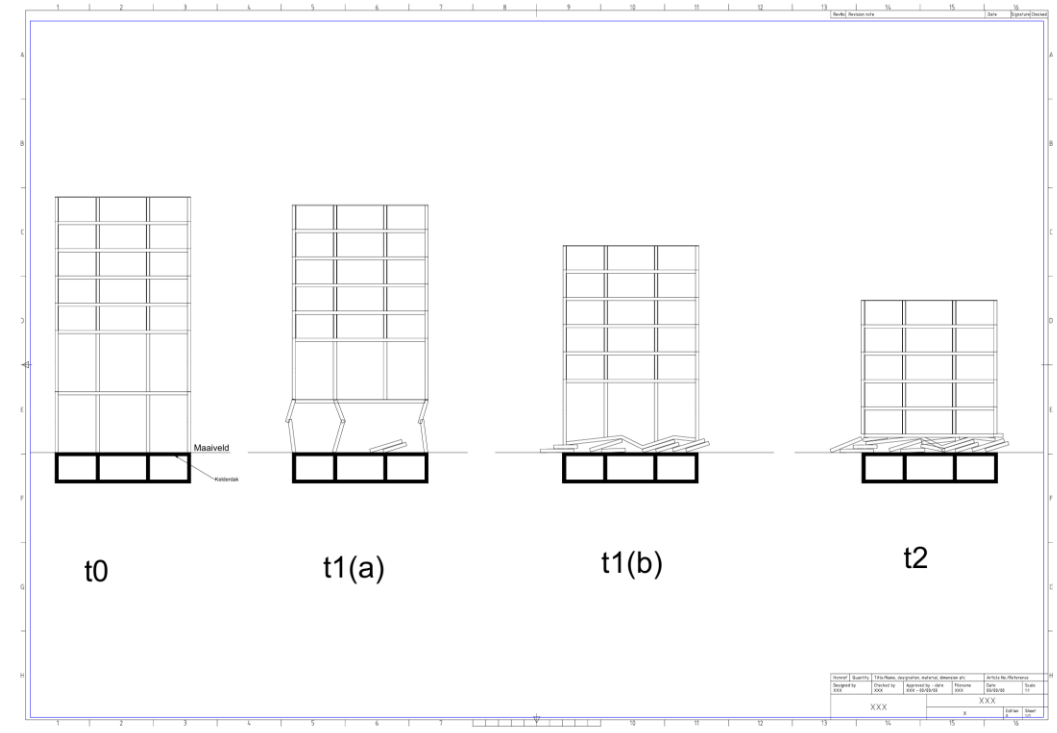

Figure 14 Sketch strip for the moment of collapse of building t0, t1 and t2. Drawn on AutoCAD

Collapse Events

- **t0:** Before the collapse, the mass of the spring-mass system is the mass of the concrete roof of the shelter.

- **t1:** A column on the ground floor is blasted, causing the building (levels 1-5) to come down. The falling mass exerts a force ($F1$) equal to the weight of the entire building falling at gravitational acceleration (g). This force acts as a point force between the columns, leading to the roof slab's displacement and dynamic response.

- **t2:** The first level floor reaches the ground level floor. The mass of the spring-mass system now includes the ground floor's mass. A column of the first-floor collapses, causing levels 2-5 to fall. The new force ($F2$) is exerted, which is less than ($F1$) due to the reduced mass.

- **t3, t4, t5, t6:** Similar events occur, with the mass and forces changing as each subsequent floor collapses until the entire building is at ground level.

4.5.2 Updated Mass Calculation

Volume Calculation: Volume = thickness × width × length = $0.5\,\text{m} \times \frac{8.3}{2}\,\text{m} \times \frac{8.4}{2}\,\text{m} = 8.715\,\text{m}^3$

Mass Calculation: Mass = Volume × density = $8.715\,\text{m}^3 \times 2400\,\text{kg/m}^3 = 20{,}916\,\text{kg}$

Updated Force Calculation:

4.5.2 Velocity at Impact:

- Free fall distance: 4.2 meters (height of one floor).

- Using the formula ($v = \sqrt{2gh}$) This equation is derived by assuming that air resistance is negligible and equating the potential energy to the kinetic energy. The potential energy at height h h is converted entirely into kinetic energy as the object falls. Solving for v: (Carbary, 2021)

$$v = \sqrt{2gh} = \sqrt{2 \times 9.81\,\text{m/s}^2 \times 4.2\,\text{m}} \approx 9.1\,\text{m/s} \qquad (4.4)$$

But for the first two floors the height is dubbed thus $v = 12.84\ \text{m/s}$

Force Calculation:

- The impact force can be estimated using impulse-momentum principle. If the debris comes to a stop within a short duration (Δt), the force (F) can be approximated by:

$$F = \frac{\Delta p}{\Delta t} = \frac{m \cdot v}{\Delta t} \qquad (4.5a)$$

$-Assuming(\Delta t = 0.1\,\text{s})$:

$$F = \frac{20{,}916\,\text{kg} \times 9.1\,\text{m/s}}{0.1\,\text{s}} \approx 1{,}903{,}356\,\text{N} \approx 1.9 \times 10^6\,\text{N} \qquad (4.5b)$$

$$F\_first\ and\ second\ floor = \frac{20{,}916\,\text{kg} \times 12.84\,\text{m/s}}{0.1\,\text{s}} \approx 2685614.4\,\text{N} \approx 2.7 \times 10^6\,\text{N} \qquad (4.5c)$$

4.5.3 Numerical Simulation Using Euler Forward Method

In appendix D the python code for the model of the building is given where Euler forward method is employed to simulate the dynamic response over time. The position (x), velocity (v), and acceleration (a) of the slab are updated at each time step based on the collapse forces, damping, and spring forces. The collapse forces are modeled using a simplified approach where each collapse event applies a force (F) for a specified duration:

$$F(t) = \sum \text{collapse\_force}\left(t, t_{\text{collapse}}, F_{\text{collapse}}, \text{duration}\right) \tag{4.6}$$

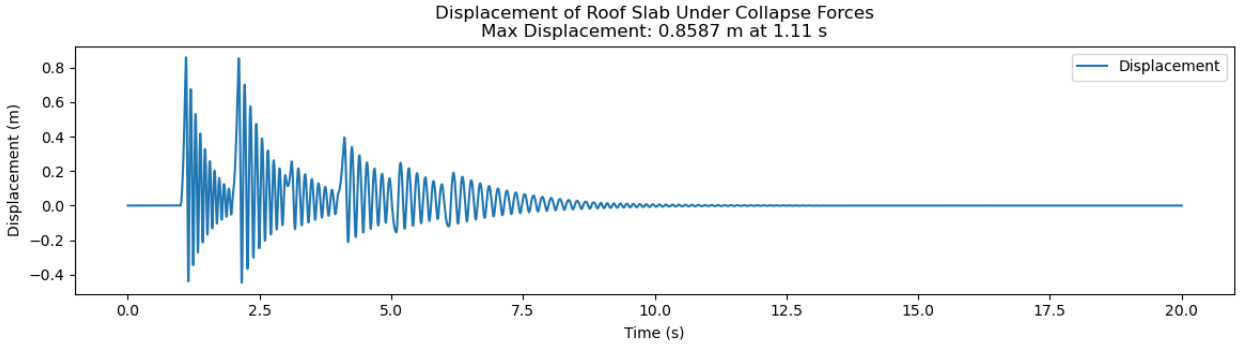

Figure 15 The displacement graph shows the roof slab's displacement over time. Peaks in the graph indicate significant deflections due to applied collapse forces.

From Figure 15, it is evident that the first two peaks of the graph correspond to the collapse of the first two floors. These peaks mean a higher displacement magnitudes due to the greater forces involved. Additionally, as the spring-mass system changes in mass after each collapse event, a slightly different response is observed even if the applied force remains constant.

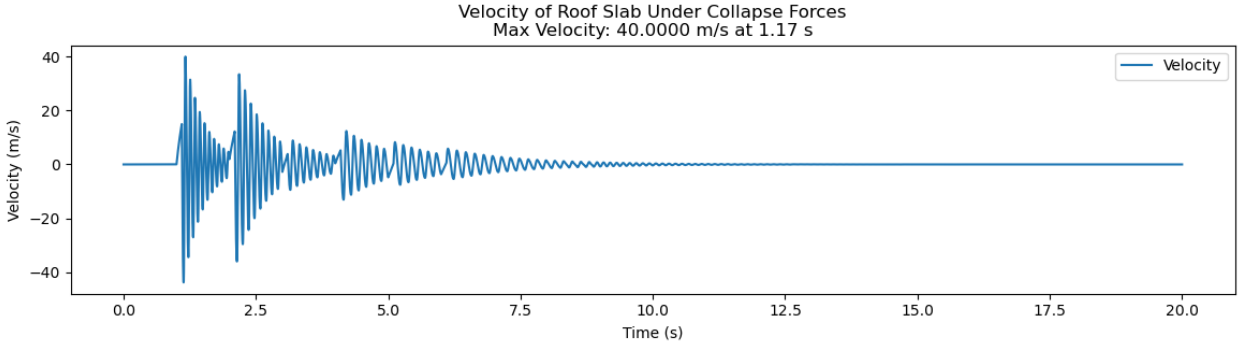

Figure 16 Velocity of shelter roof slab.

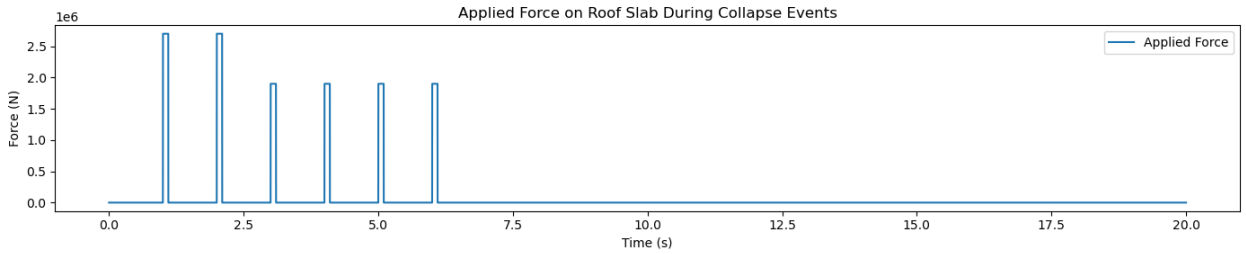

Figure 17 The applied force graph illustrates the forces exerted on the roof slab during the collapse events. Notice the short duration of these applied forces; this duration estimates the time required for the system to halt the falling mass during the building's collapse.

5 Calculation of Reinforcement and Maximum Displacement Before Failure

To determine whether the dynamic response leads to the failure of the shelter structure, we need to perform several calculations based on the reinforcement properties and the stress-strain relationships. Here's a step-by-step process:

5.1 Parameters

5.1.1 Calculate the Compressive Force in the Concrete (N_c)

- Use the rule of thumb formula:

$$N_c = \alpha \cdot b \cdot X_u \cdot f_{cd} \tag{5.1}$$

(Abspoel et al., 2013)

- Given:

  - $(\alpha = 0.75)$ - $(b = 3\,\text{m})$

  - $(H = 0.5\,\text{m})$(thickness of the roof slab)

  - $(X_u = \frac{1}{5}H = \frac{1}{5} \times 0.5 = 0.1\,\text{m})$

  - $(f_{cd} = \frac{f_{ck}}{1.5}), with (f_{ck} = 30\,\text{MPa})$: $f_{cd} = \frac{30}{1.5} = 20\,\text{MPa} = 20 \times 10^6\,\text{N/m}^2$

Thus $N_c = 0.75 \cdot 3 \cdot 0.1 \cdot 20 \times 10^6 = 4.5 \times 10^6\,\text{N}$

5.1.2 Determine the Force in the Reinforcement (N_s)

$-(N_s = N_c)$

- Given:

$-(f_{yd} = \frac{f_{yk}}{1.15}), with (f_{yk} = 500\,\text{N/mm}^2)$: $f_{yd} = \frac{500}{1.15} \approx 434.78\,\text{N/mm}^2 = 434.78 \times 10^6\,\text{N/m}^2$

5.1.3 Calculate the Area of Reinforcement (A_s)

$$N_s = A_s \cdot f_{yd} \tag{5.2}$$

(Abspoel et al., 2013)

$4.5 \times 10^6 =\cdot 434.78 \times 10^6$ ➔ $A_s = \frac{4.5 \times 10^6}{434.78 \times 10^6} \approx 0.01035\,\text{m}^2$

5.1.4 Determine the Number of Reinforcement Bars

- Given the diameter of each bar is 16 mm ($d = 16\,\text{mm} = 0.016\,\text{m}$):

$A_{\text{single bar}} = \pi \left(\frac{d}{2}\right)^2 = \pi \left(\frac{0.016}{2}\right)^2 \approx 2.01 \times 10^{-4}\,\text{m}^2$

- Number of bars:

Number of bars $= \frac{A_s}{A_{\text{single bar}}} = \frac{0.01035}{2.01 \times 10^{-4}} \approx 51.49$, Thus, approximately 52 reinforcement bars are used.

5.1.5 Calculate Maximum Extension Before Breaking:

- Given the ultimate strain for reinforcement steel ($\epsilon_{ud} = 4.5\backslash\% = 0.045$):

- The initial length of the slab is the effective length: $\Delta L_{\text{max}} = L \cdot \epsilon_{ud} = 7.2\,\text{m} \cdot 0.045 \approx 0.324\,m$

Assuming a simply supported beam with uniform extension along its length. The parabolic shape is chosen as it more accurately represents the deflection of beams under uniform conditions compared to a circular shape. Using a measure tape and extending the length with 0.3 m while fixing the endpoints in place, will lead to a deflection shape with a maximum of 0.75 m at the midpoint.

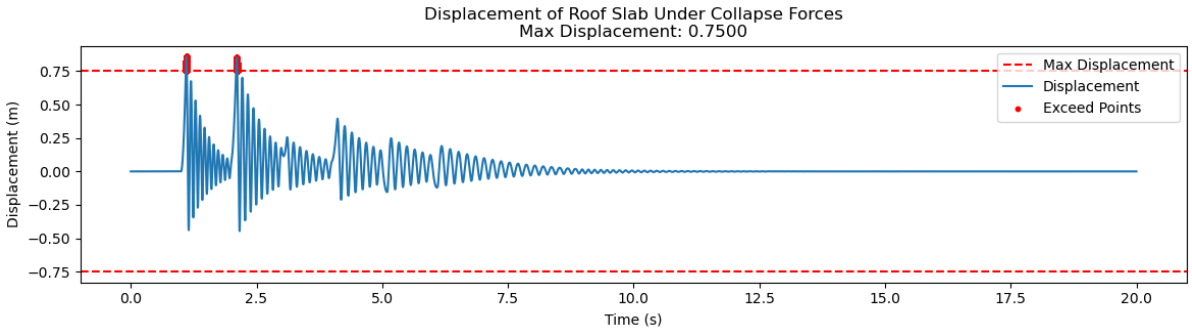

Figure 18 The maximum allowable displacement is in red, and the model clearly indicates that this threshold is exceeded during the first two collapse events.

This means that the basement roof is unable to withstand the collapse event. Furthermore, when the first mass impacts the ground floor, a significant force of 2.7 * 10^9 N is exerted. This results in a displacement greater than the maximum allowable displacement for the underground level roof, leading to its collapse.

5.2 Consideration of Measurement and Simulation Errors

The aim is to stay within limit of 10% and that must include:

- The expected error from the Euler forward simulation model is approximately 1%.
- Measurement errors can be up to 8 to 10%, particularly since the building drawings were not available, and hand measurements had to be performed.

6 Conclusion and recommendations

This study aimed to determine the safety of the basement in the Civil Engineering Building at TU Delft as a shelter if a ground floor column were to be blown away by a bomb. The research commenced with the development of a spring-mass system model, which was analysed under various conditions, both analytically and numerically. This initial phase was crucial to establish a reliable numerical system applicable to the building. By understanding the behaviour of the spring-mass system, we were able to predict the expected error levels in our model, ensuring the accuracy and dependability of our subsequent analysis.

Following this theoretical groundwork, detailed measurements of the Civil Engineering and Geosciences (CEG) building's basement and various levels were conducted to obtain precise structural specifications. With this data, we created a comprehensive simulation model to predict the building’s behaviour in a collapse scenario, focusing on the dynamic response to forces exerted by the collapse of a ground floor column.

The model simulated the impact of a bomb blast capable of blowing away a ground floor column, using high-explosive mortar rounds known for their significant destructive power. The analysis included calculating the force these munitions could generate, which far exceeds the load-bearing capacity of typical reinforced concrete columns. This scenario assessed the load that would be placed on the basement roof due to the collapse of the floors above it. The dynamic response simulations indicated that the displacements caused by the collapse events would exceed the maximum allowable displacements, leading to structural failure. When the first mass impacts the ground floor, the force exerted results in a displacement greater than what the underground roof can withstand, ultimately causing it to collapse.

In conclusion, based on our simulations and calculations, the basement of the Civil Engineering Building is not currently safe to use as a shelter under the specified conditions. The collapse of a ground floor column due to a bomb impact leads to forces and displacements that exceed the structural capacity of the basement roof. This study underscores the urgent need for structural reinforcements and further research to enhance the safety of underground shelters in scenarios involving extreme impacts. By addressing these vulnerabilities, it is possible to significantly improve the structural integrity and safety of such shelters, providing better protection for individuals in emergency situations.

Future Research and Recommendations

To prevent the collapse of the shelter, future researchers should focus on the following areas:

1. **Structural Reinforcement**: One potential solution is to add additional columns in the basement. By halving the distance between existing columns, the load distribution would be improved, reducing the stress on any single point. This could effectively increase the overall stability of the underground roof and help it withstand collapse events.

2. **Material Improvements**: Investigate the use of stronger or more flexible materials that can better absorb and distribute the forces experienced during collapse events. Advanced composite materials or high-strength alloys could be considered.
3. **Simulation and Modeling**: Enhance simulation models to include more variables and potential scenarios. This could include varying the impact angles, the masses involved, and different collapse sequences to better predict and mitigate potential failures.

By focusing on these research areas, future improvements can be made to enhance the structural integrity and prevent the collapse of underground shelters in the event of significant impacts.

Literature list

Abspoel, R., Vries, P., & Bijlaard, F. (2013). Staalconstructies. In *Webedu* (No. 06917280056). Delft university of technology. Retrieved June 18, 2024, from https://onlinereaders.tudelft.nl/index.php?orderableObject=4200118

Blaauwendraad, J. (2016). Collegedictaat CTB2300 Dynamica van Systemen. In *webedu.nl* (No. 06917280059). Delft university of technology. Retrieved June 18, 2024, from https://onlinereaders.tudelft.nl/index.php?orderableObject=22500343

Carbary, J. (2021, April 16). *Kinetic and potential energy - wyzant lessons*. Wyzant Lessons. https://www.wyzant.com/resources/lessons/science/physics/kinetic-and-potential-energy/

Critical damping coefficient. (2022, February 20). Physics Forums: Science Discussion, Homework Help, Articles. https://www.physicsforums.com/threads/where-does-the-equation-c-2-sqrt-km-for-critical-damping-come-from.1012525/#:~:text=What%20does%20the%20equation%20C,oscillating%20or%20vibrating%20too%20much.

De Kruif, M. (2024, March 22). *Dit zijn tekenen van groot Russisch offensief* [Video]. Telegraaf. Retrieved June 24, 2024, from https://www.telegraaf.nl/video/566840997/dit-zijn-tekenen-van-groot-russisch-offensief

Differential Equations Solution Guide. (n.d.). https://www.mathsisfun.com/calculus/differential-equations-solution-guide.html

Forward and backward Euler methods. (n.d.). https://web.mit.edu/10.001/Web/Course_Notes/Differential_Equations_Notes/node3.html

Hartsuijker, C., & Welleman, H. (2007). Constructiemechanica 3 module: Stabiliteit van het evenwicht. In *webedu.nl* (No. 06917280022). Delft university of technology. Retrieved June 18, 2024, from https://onlinereaders.tudelft.nl/index.php?orderableObject=11111

Libretexts. (2022, July 26). *1.2: Forward Euler method*. Mathematics LibreTexts. https://math.libretexts.org/Bookshelves/Differential_Equations/Numerically_Solving_Ordi

nary_Differential_Equations_(Brorson)/01%3A_Chapters/1.02%3A_Forward_Euler_method

Nammo. (2023, July 26). *120 mm Mortar High Explosive Round - Nammo*. https://www.nammo.com/product/our-products/ammunition/large-caliber-ammunition/mortar-rounds/120-mm-mortar-high-explosive-round/

NASA Glenn Research Center. (2023, August 7). *Newton's Laws of Motion | Glenn Research Center | NASA*. Glenn Research Center | NASA. https://www1.grc.nasa.gov/beginners-guide-to-aeronautics/newtons-laws-of-motion/

Numeracy, Maths and Statistics - Academic Skills kit. (n.d.). https://www.ncl.ac.uk/webtemplate/ask-assets/external/maths-resources/mechanics/dynamics/impulse-and-momentum.html#:~:text=The%20impulse%20of%20a%20force,is%20the%20change%20in%20momentum.

Pääkkönen, R. (1991). Low-Frequency Noise Impulses from Explosions. *Journal of Low Frequency Noise, Vibration and Active Control*, *10*(3), 78–82. https://doi.org/10.1177/026309239101000302

The Editors of Encyclopaedia Britannica. (1998, July 20). *Hooke's law | Description & Equation*. Encyclopedia Britannica. https://www.britannica.com/science/Hookes-law

Tons (Explosives) to gigajoules conversion calculator. (n.d.). http://www.unitconversion.org/energy/tons-explosives-to-gigajoules-conversion.html

Appendix A Python Code Numerical solution

This code was used to calculate the displacement of spring mass system with no damping.

```
import numpy as np
import matplotlib.pyplot as plt

# Constants
g = 9.81  # gravitational constant
k = 40  # spring constant
m = 1  # mass

# Time settings
t_start = 0  # initial time
t_end = 5  # final time
num_steps = 10**5  # number of steps in the time span
h = (t_end - t_start) / num_steps  # time step size
t = np.linspace(t_start, t_end, num_steps)  # time array

# Initial conditions: position and velocity
x = np.zeros(num_steps)  # position array
x_dot = np.zeros(num_steps)  # velocity array

x[0] = 0  # initial position
x_dot[0] = 0  # initial velocity

# Euler's forward method to solve the system
for i in range(1, num_steps):
    # Update position
    x[i] = x[i - 1] + h * x_dot[i - 1]

    # Compute acceleration
    acceleration = g - (k * x[i - 1]) / m

    # Update velocity
    x_dot[i] = x_dot[i - 1] + h * acceleration

# Plotting position and velocity over time
plt.plot(t, x, 'r', lw=2, label=r'$x$')  # position over time
plt.plot(t, x_dot, 'b', lw=2, label=r'$\dot x$')  # velocity over time
plt.axhline(0.25, color='gray', linestyle='--', linewidth=0.5,
label='Equilibrium')
plt.title('Spring-Mass System (Euler Method)')
plt.xlabel('Time (seconds)')
plt.ylabel('Position (x), Velocity (dx/dt)')
plt.legend()
plt.grid(True)
plt.show()

# Phase diagram (Position vs. Velocity)
plt.plot(x, x_dot, 'b', lw=2)
plt.title('Phase Diagram (Euler Method)')
```

This is used to calculate the displacement of spring mass system with damping.

```
import numpy as np
import matplotlib.pyplot as plt

# Constants
g = 9.81  # gravitational constant, used to calculate acceleration due to
gravity
k = 40  # spring constant
m = 1  # mass
b = 1  # damping coefficient

# Time settings
t_start = 0
t_end = 10
num_steps = 10**5
h = (t_end - t_start) / num_steps  # time step
t = np.linspace(t_start, t_end, num_steps)  # time points

# Initial conditions: position and velocity
x = np.zeros(num_steps)
x_dot = np.zeros(num_steps)

x[0] = 0  # initial position
x_dot[0] = 0  # initial velocity

# Define the spring-mass-damper system with Euler's forward method
for i in range(1, num_steps):
    # Update position
    x[i] = x[i - 1] + h * x_dot[i - 1]

    # Calculate acceleration (second-order ODE)
    acceleration = g - (k * x[i - 1] / m) - (b * x_dot[i - 1] / m)

    # Update velocity
    x_dot[i] = x_dot[i - 1] + h * acceleration

# Plot position and velocity over time
plt.plot(t, x, 'r', lw=2, label='Position (x)')
plt.plot(t, x_dot, 'b', lw=2, label='Velocity (dx/dt)')
plt.axhline(0.25, color='gray', linestyle='--', linewidth=0.5,
label='Equilibrium')
plt.title('Spring-Mass-Damper System (Euler Method)')
plt.xlabel('Time (seconds)')
plt.ylabel('Position (x), Velocity (dx/dt)')
plt.legend()
plt.grid(True)
plt.show()

# Plot phase space (position vs. velocity)
```

This is used to calculate the displacement of spring mass system with damping and driving force.

```
import numpy as np
import matplotlib.pyplot as plt

# Constants
m = 10  # mass
k = 40  # spring constant
b = 10  # damping coefficient
F_0 = 10  # amplitude of the driving force
w = 1.5  # angular frequency of the driving force
g = 9.81

# Time settings
t_start = 0
t_end = 20
num_steps = 500
h = (t_end - t_start) / num_steps  # time step
t = np.linspace(t_start, t_end, num_steps)  # time array

# Initial conditions: position and velocity
x = np.zeros(num_steps)
x_dot = np.zeros(num_steps)

x[0] = 0  # initial position
x_dot[0] = 0  # initial velocity

# Implement Euler's forward method to solve the system
for i in range(1, num_steps):
    # Update position
    x[i] = x[i - 1] + h * x_dot[i - 1]

    # Compute acceleration
    acceleration = g - (b * x_dot[i - 1] / m) + (F_0 * np.cos(w * t[i - 1]) /
m) - (k * x[i - 1] / m)

    # Update velocity
    x_dot[i] = x_dot[i - 1] + h * acceleration

# Plot position and velocity over time
plt.plot(t, x, 'r', lw=2, label='Position (x)')
plt.plot(t, x_dot, 'b', lw=2, label='Velocity (dx/dt)')
plt.title('Forced Spring-Mass System (Euler Method)')
plt.xlabel('Time (s)')
plt.ylabel('Position (x), Velocity (dx/dt)')
plt.legend()
plt.grid(True)
plt.show()
```

Appendix B Python code analytical solutions

First spring mass system:

```
import numpy as np
import matplotlib.pyplot as plt

# Function to simulate the oscillations of a mass-spring system
def simulate_oscillation(amplitude, omega, phase, time):
    # Return the displacement at given time(s)
    return amplitude * np.cos(omega * time - phase)

# Parameters
# Amplitude (maximum displacement from equilibrium)
amplitude = 5.0  # You can change this to adjust amplitude

# Angular frequency (related to spring constant and mass)
omega = 1.5  # You can change this to adjust frequency

# Phase shift (how much the oscillation is shifted)
phase = 0.5  # You can change this to adjust phase shift

# Time array for simulation
time = np.linspace(0, 10, 1000)  # 0 to 10 seconds with 1000 points

# Simulate oscillation
displacement = simulate_oscillation(amplitude, omega, phase, time)

# Plotting
plt.plot(time, displacement, label="Oscillation")
plt.title("Harmonic Oscillation of a Mass-Spring System")
plt.xlabel("Time (s)")
plt.ylabel("Displacement from Equilibrium")
plt.axhline(0, color='gray', linestyle='--', linewidth=0.5,
label='Equilibrium')
plt.legend()
plt.show()
```

Second spring mass system:

```
import numpy as np
import matplotlib.pyplot as plt

# Constants
m = 1  # mass
k = 40  # spring constant
c = 1  # damping coefficient
t = np.linspace(0, 10, 500)  # time array

# Angular frequency for the underdamped case
omega_1 = np.sqrt(4 * k * m - c ** 2) / (2 * m)

# Decay constant
alpha = c / (2 * m)

# Arbitrary constants for oscillatory part (choose initial amplitude and
phase)
A = 1.0  # amplitude constant
B = 0.0  # initial phase shift constant

# Analytical solution for underdamped oscillations
x_analytical = np.exp(-alpha * t) * (A * np.cos(omega_1 * t) + B *
np.sin(omega_1 * t))

# Plotting the analytical solution
plt.plot(t, x_analytical, label='Analytical Solution')
plt.axhline(0, color='gray', linestyle='--', linewidth=0.5,
label='Equilibrium')
plt.xlabel('Time (s)')
plt.ylabel('Position (x)')
plt.title('Damped Spring-Mass Oscillation')
plt.legend()
plt.grid()
plt.show()
```

Third spring mass system:

```
import numpy as np
import matplotlib.pyplot as plt

# Constants
m = 1  # mass
k = 40  # spring constant
b = 1  # damping coefficient
F0 = 10  # amplitude of the driving force
omega = 5  # driving frequency
t = np.linspace(0, 10, 500)  # time array

# Angular frequency for the underdamped case
omega_1 = np.sqrt(4 * k * m - b ** 2) / (2 * m)  # damped natural frequency

# Decay constant for damping
alpha = b / (2 * m)

# Steady-state solution (particular solution)
x_steady_state = (F0 / (m * ((omega_1 ** 2 - omega ** 2) ** 2 + (b * omega /
m) ** 2) ** 0.5)) * np.cos(omega * t)

# Homogeneous solution (natural oscillations)
x_homogeneous = np.exp(-alpha * t) * np.cos(omega_1 * t)

# Total analytical solution: homogeneous + steady-state
x_analytical = x_homogeneous + x_steady_state

# Plotting the analytical solution
plt.plot(t, x_analytical, label='Analytical Solution')
plt.axhline(0, color='gray', linestyle='--', linewidth=0.5,
label='Equilibrium')
plt.xlabel('Time (s)')
plt.ylabel('Position (x)')
plt.title('Forced Damped Spring-Mass Oscillation')
plt.legend()
plt.grid()
plt.show()
```

Appendix C Python code to compare analytical and numerical solutions

```
import numpy as np
import matplotlib.pyplot as plt

# Parameters
m = 1.0  # mass (kg)
k = 40.0  # spring constant (N/m)
c = 2.0  # damping coefficient (Ns/m)
F0 = 10.0  # Amplitude of driving force (N)
omega_drive = 1.5  # Driving angular frequency (rad/s)

# Derived parameters
omega_0 = np.sqrt(k / m)
gamma = c / (2 * m)
omega_d = np.sqrt(omega_0**2 - gamma**2)

# Time parameters
t_start = 0
t_end = 20  # End time, change this as needed
dt = 0.00001  # Fixed time step for consistent accuracy
num_points = int((t_end - t_start) / dt) + 1  # Calculate number of points
t_points = np.linspace(t_start, t_end, num_points)

# Initial conditions
x_init = 1.0  # Initial position (m)
v_init = 0.0  # Initial velocity (m/s)

# Analytical solution: Steady-state response
A = F0 / (m * np.sqrt((omega_0**2 - omega_drive**2)**2 + (2 * gamma *
omega_drive)**2))
delta = np.arctan2(2 * gamma * omega_drive, omega_0**2 - omega_drive**2)
x_steady_state = A * np.cos(omega_drive * t_points - delta)
v_steady_state = -A * omega_drive * np.sin(omega_drive * t_points - delta)

# Analytical solution: Transient response
C1 = x_init - x_steady_state[0]
C2 = (v_init + gamma * C1) / omega_d
x_transient = np.exp(-gamma * t_points) * (C1 * np.cos(omega_d * t_points) +
C2 * np.sin(omega_d * t_points))
v_transient = np.exp(-gamma * t_points) * (-C1 * omega_d * np.sin(omega_d *
t_points) + C2 * omega_d * np.cos(omega_d * t_points)) - gamma * x_transient

# Full analytical solution
x_analytical = x_steady_state + x_transient
v_analytical = v_steady_state + v_transient

# Numerical solution
x_numerical = np.zeros(len(t_points))
v_numerical = np.zeros(len(t_points))
```

```
x_numerical[0] = x_init
v_numerical[0] = v_init

# Euler forward method
for i in range(1, len(t_points)):
    a = (F0 * np.cos(omega_drive * t_points[i-1]) - c * v_numerical[i-1] - k *
x_numerical[i-1]) / m
    v_numerical[i] = v_numerical[i-1] + dt * a
    x_numerical[i] = x_numerical[i-1] + dt * v_numerical[i-1]

# Compute error
position_error = np.abs(x_numerical - x_analytical)
velocity_error = np.abs(v_numerical - v_analytical)

# Handle division by zero by masking zero values
position_error_percentage = np.where(np.abs(x_analytical) > 0, (position_error
/ np.abs(x_analytical)) * 100, 0)
velocity_error_percentage = np.where(np.abs(v_analytical) > 0, (velocity_error
/ np.abs(v_analytical)) * 100, 0)

mean_position_error_percentage = np.mean(position_error_percentage)
mean_velocity_error_percentage = np.mean(velocity_error_percentage)

# Plotting the results
plt.figure(figsize=(14, 7))

plt.subplot(2, 2, 1)
plt.plot(t_points, x_analytical, label='Analytical Position')
plt.plot(t_points, x_numerical, label='Numerical Position',
linestyle='dashed')
plt.xlabel('Time (s)')
plt.ylabel('Position (m)')
plt.legend()
plt.title('Position Comparison')

plt.subplot(2, 2, 2)
plt.plot(t_points, v_analytical, label='Analytical Velocity')
plt.plot(t_points, v_numerical, label='Numerical Velocity',
linestyle='dashed')
plt.xlabel('Time (s)')
plt.ylabel('Velocity (m/s)')
plt.legend()
plt.title('Velocity Comparison')

plt.subplot(2, 2, 3)
plt.plot(t_points, position_error, label=f'Position Error (Mean Error:
{mean_position_error_percentage:.2f}%)')
plt.xlabel('Time (s)')
```

```
plt.ylabel('Absolute Error (m)')
plt.legend()
plt.title(f'Position Error (Mean Error:
{mean_position_error_percentage:.2f}%)')

plt.subplot(2, 2, 4)
plt.plot(t_points, velocity_error, label=f'Velocity Error (Mean Error:
{mean_velocity_error_percentage:.2f}%)')
plt.xlabel('Time (s)')
plt.ylabel('Absolute Error (m/s)')
plt.legend()
plt.title(f'Velocity Error (Mean Error:
{mean_velocity_error_percentage:.2f}%)')

plt.tight_layout()
plt.show()

# Display error percentages
print(f'Mean Position Error Percentage:
{mean_position_error_percentage:.2f}%')
print(f'Mean Velocity Error Percentage:
{mean_velocity_error_percentage:.2f}%')
```

Appendix D Python Code of the building's basement roof simulation

```
import numpy as np
import matplotlib.pyplot as plt

# Shelter dimensions and material properties
column_width = 0.55  # meters
slap_effective_width = 3  # meters
distance_between_columns = 7.75  # meters
distance_to_wall = 6.36  # meters
slab_thickness = 0.5  # meters
concrete_density = 2400  # kg/m^3

# Define initial modulus of elasticity
E_initial = 30e9  # Pa (modulus of elasticity for C30 concrete)
E_cracked = E_initial / 3  # Pa, modulus of elasticity for cracked concrete

# Calculate effective length of the beam
effective_length = 0.75 * distance_between_columns

# Moment of inertia for the rectangular cross-section of the slab
I = (slap_effective_width * slab_thickness**3) / 12

# Calculate the initial spring constant k (beam on four support points)
k_initial = (384 / 5) * (E_initial * I) / (effective_length**4)
k_cracked = (384 / 5) * (E_cracked * I) / (effective_length**4)

# For beam on four support points, the spring constant is approximately
doubled
k_initial *= 2
k_cracked *= 2

# Damping coefficient
damping_ratio = 0.04  # 4%
c = 2 * damping_ratio * np.sqrt(k_initial * (concrete_density * slab_thickness
* effective_length))

# Initial mass of the roof slab
initial_mass = concrete_density * slab_thickness * effective_length

# Time parameters
t_start = 0
t_end = 20  # seconds (for a longer simulation)
dt = 0.00001  # Fixed time step for consistent accuracy
num_points = int((t_end - t_start) / dt) + 1  # Calculate number of points
t_points = np.linspace(t_start, t_end, num_points)

# Initial conditions
x_init = 0.0  # Initial displacement (m)
v_init = 0.0  # Initial velocity (m/s)

# Numerical solution arrays
x_numerical = np.zeros(len(t_points))
v_numerical = np.zeros(len(t_points))
a_numerical = np.zeros(len(t_points))
F_applied = np.zeros(len(t_points))

x_numerical[0] = x_init
```

```
v_numerical[0] = v_init

# Force due to building collapse at each time step (simplified model)
def collapse_force(t, t_collapse, F_collapse, duration):
    return F_collapse if t_collapse <= t < t_collapse + duration else 0

# Define collapse events
collapse_events = [
    (1, 2.7e6, 0.1),  # At t1, force due to initial collapse
    (2, 2.7e6, 0.1),  # At t2, subsequent collapse force
    (3, 1.9e6, 0.1),  # At t3, subsequent collapse force
    (4, 1.9e6, 0.1),  # At t4, subsequent collapse force
    (5, 1.9e6, 0.1),  # At t5, subsequent collapse force
    (6, 1.9e6, 0.1)   # At t6, subsequent collapse force
]

# Euler forward method for numerical simulation
for i in range(1, len(t_points)):
    t = t_points[i-1]

    # Update the mass at each collapse event
    current_mass = initial_mass
    for j in range(1, 7):
        if t >= j:
            current_mass += concrete_density * slab_thickness *
effective_length

    # Calculate the applied force
    F = sum(collapse_force(t, t_collapse, F_collapse, duration) for
t_collapse, F_collapse, duration in collapse_events)
    F_applied[i] = F

    # Update the modulus of elasticity based on the force-deflection
relationship
    if x_numerical[i-1] < 0.016:  # Assuming 0.01 m as the deflection limit
for uncracked concrete
        E = E_initial
        k = k_initial
    else:
        E = E_cracked
        k = k_cracked

    if F > 25:  # If the reinforcement yields, set stiffness to zero (almost)
        k = 0.03

    # Calculate acceleration, velocity, and displacement
    a = (F - c * v_numerical[i-1] - k * x_numerical[i-1]) / current_mass
    v_numerical[i] = v_numerical[i-1] + dt * a
    x_numerical[i] = x_numerical[i-1] + dt * v_numerical[i-1]
    a_numerical[i] = a

# Calculate maximum values and their corresponding times
max_displacement = np.max(x_numerical)
max_displacement_time = t_points[np.argmax(x_numerical)]

max_velocity = np.max(v_numerical)
```

```
max_velocity_time = t_points[np.argmax(v_numerical)]

# Plot results
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
plt.plot(t_points, x_numerical)
plt.xlabel('Time (s)')
plt.ylabel('Displacement (m)')
plt.title('Dynamic Response with Nonlinear Stiffness')
plt.grid(True)

plt.subplot(2, 1, 2)
plt.plot(t_points, v_numerical)
plt.xlabel('Time (s)')
plt.ylabel('Velocity (m/s)')
plt.grid(True)

plt.tight_layout()
plt.show()

# Print maximum values
print(f'Max Displacement: {max_displacement} m at {max_displacement_time} s')
print(f'Max Velocity: {max_velocity} m/s at {max_velocity_time} s')
```