FMH606 Master's Thesis 2021

Energy and Environmental Technologies

The hazard of fragments in accidental H2-air explosions

Maulen Yerbolatov

Faculty of Technology, Natural sciences and Maritime Sciences
Campus Porsgrunn

University of
South-Eastern Norway

www.usn.no

Course: FMH606 Master's Thesis, 2021

Title: The hazard of fragments in accidental H_2-air explosions

Number of pages: <x>

Keywords: Hydrogen, vapor cloud explosion, fragments

Student: Maulen Yerbolatov

Supervisor: Dag Bjerketvedt

External partner: < >

Summary:

In this thesis were shown basic toolbox of blast estimation and hazards probabilities due to fragments, written in open-source programming language, Python. Used method in blast parameters calculation is easily appliable to any other stoichiometric gaseous fuel-air mixture. A code repository is given in appendix and it is freely available to download, modify and redistribute under the GNU GPLv3 license.

Preface

This work was done from distance due to hard times with Covid-19.

With almost no proper experience in Python programming, this thesis was quite challenging for me. Also, the actual workload was underestimated by me due to working full-time. But I plan to use this interesting experience for my further development as base in programming in engineering field.

I'd like to thank Dag Bjerketvedt for discussions we had, all the help I received and his patience in working with me. Also my parents for their support, as well as my friend Dulat for his advices in script writing.

Almaty/ Aktogay, Kazakhstan, 19 of May 2021 (13th August)

Maulen Yerbolatov

Contents

Nomenclature

CFD	Computational fluid dynamics
CSD	Computational structural dynamics
DDT	Deflagration to detonation transition
HyRAM	Hydrogen Risk Assessment Models
LPG	Liquefied Petroleum Gas
QRA	Quantitative risk assessment
TNO	Nederlandse Organisatie voor Toegepast Natuurwetenschappelijk Onderzoek (English: Netherlands Organization for Applied Scientific Research), an independent research organization in the Netherlands.
TNT	Trinitrotoluene
VCE	Vapor cloud explosion

1 Introduction

1.1 Background

Hydrogen generation, transportation and storage, and hence the need to assess and control hydrogen explosions hazards is common to many energy-related fields. This highly flammable gas is good alternative to hydrocarbons due to almost no emissions, thus becomes widely spread. Nowadays hydrogen stations can be found close to populated parts of modern cities, so risks of explosion hazards are real. Many articles related to this issue can be found, but mostly they are related to direct or primary damage from accidents or to formation of overpressure and impulse from blast wave of VCE. This work related to dangers from secondary risks with direct collision of structure fragments and debris with human.

1.2 Existing software

Explosion modelling and potential damage approximation considered as part of quantitative risk assessment (QRA). Up to date there are many software tools for quantitative risk assessment (QRA), over 80 tools[1]. But most of them does not support vapor cloud explosions (VCE) or does not do necessary calculations with hydrogen as fuel, so only a few can be used for damage estimation from fragments.

HyRAM (Hydrogen Risk Assessment Models) is opensource software tool, was developed by Sandia National Laboratories for the U.S. Department of Energy (DOE) Office of Energy Efficiency and Renewable Energy (EERE) Hydrogen and Fuel Cell Technologies Office (HFTO). The toolkit uses variety of models for different types of physical and chemical effects following hydrogen release accident. If we take a look at probit functions in this software, they mostly are related to primary or direct harms from resulting processes of accidents. There are also CFD based tools, as *FLACS*, released by GexCon AS and KFX, built by ComputIT. But cost of such tools is expensive and applying every detail for proper usage of them is troublesome due to limited access to source code.
So in some cases company can do QRA calculations by making their own toolkit, not necessarily thorough, but enough for certain cases.

1.3 Hydrogen safety

Pure hydrogen (H2) is the lightest among the gases present in our atmosphere and as such is highly diffusive, 3.8 times more so than natural gas, and extremely buoyant [2]. This causes H2 to rise at a rate of up to 20m/s in otherwise ambient air. Combination of these effects makes difficult to get a combustible concentration of hydrogen without the presence of a confined space with some form of roof to keep the gas from escaping too quickly into the atmosphere. That being said, once the concentration is in the rather wide range of flammability, the gas is highly combustible with only a required ignition energy of roughly 1/10th. that of natural gas.

In this thesis mainly will be taken into account mass/volume of hydrogen in stoichiometric mixture of hydrogen-air and without any direct relation to geometry.

1.4 Goal and structure of the thesis

The goal of this work is to find suitable equations and assumptions based on related works and experiments and to write Python code to estimate blast parameters and possibilities of harm from fragments.

This work is divided into two main parts:

- Defining initial parameters and calculation of blast parameters
- Calculation of possibilities of hazards

2 Theory

Explosion is a rapid release of energy. In this work as explosion will defined energy release of flammable gas or liquid explosion. According to [3], in process industries such explosions can be divided to 7 types: a) *physical explosions*, b) *condensed phase explosions*, c) vapor cloud explosions (VCEs), d) *boiling liquid expanding vapor explosions (BLEVEs)*, e) *confined explosions with reaction*, f) *vapor escapes into, and explosions in, buildings (VEEBs)*, g) *dust explosions*. In this work, VCEs will be discussed.

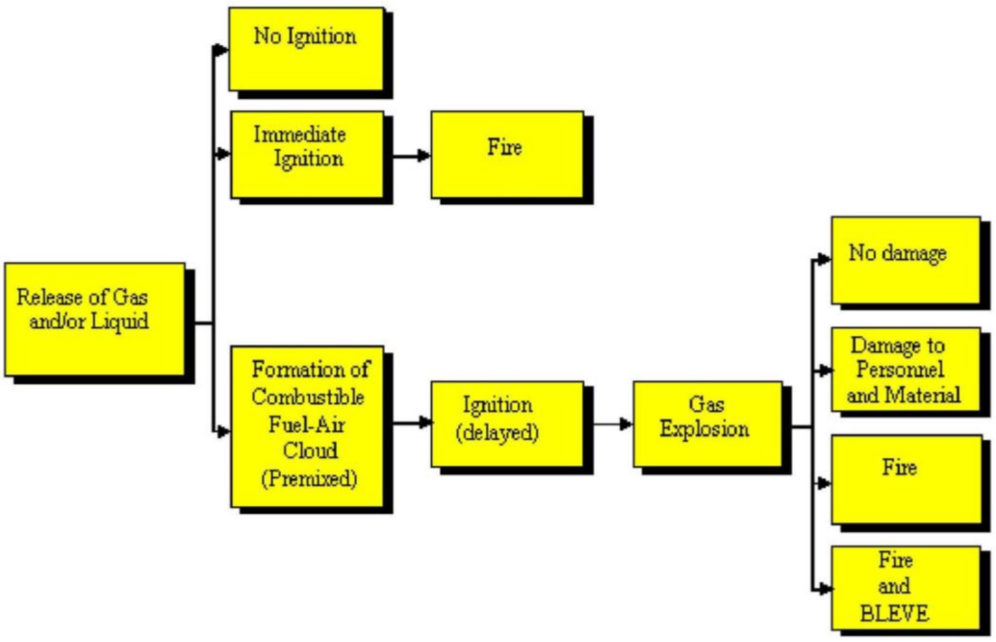

No Ignition
Immediate Ignition
Fire
Release of Gas and/or Liquid
Formation of Combustible Fuel-Air Cloud (Premixed)
Ignition (delayed)
Gas Explosion
No damage
Damage to Personnel and Material
Fire
Fire and BLEVE

Figure 2.1. A typical event tree of accidental releases of combustible gas or evaporating liquid () [1]

2.1 Deflagration, Detonation and Deflagration to detonation transition

Deflagration is process where front of reaction, in this case combustion, moves slower than speed of sound in unreacted component. According to [4], in case of deflagration flame starts as laminar, so flame propagation is dictated by molecular diffusion of mass and heat. This process of diffusion is slow, so velocity of laminar flame is approximately 3-4 m/s. Due to such slow propagation, overpressure from deflagration mostly is in order of millibars [Guidelines for Evaluating the Characteristics of Vapor Cloud Explosions, Flash Fires, and BLEVEs].

Fundamental difference between deflagration and detonation is propagation mechanisms. Detonation is process where front of reaction moves faster than speed of sound in unreacted component. In detonation reaction front propagates by shock wave, that compresses unreacted mixture. For fuel-air mixture at normal conditions propagation velocity reaches 1500-2000 m/s and peak pressure – 15-20 bar [4].

Graphical comparison between deflagration and detonation is shown on Figure 2.2. from [3]

Deflagration to detonation transition (DDT) is a process where energy from the deflagration can accumulate in the pressure wave in pipes and channels due to their geometry. When accumulated energy reaches its limits, the resulting compression of the gas leads to a detonation.

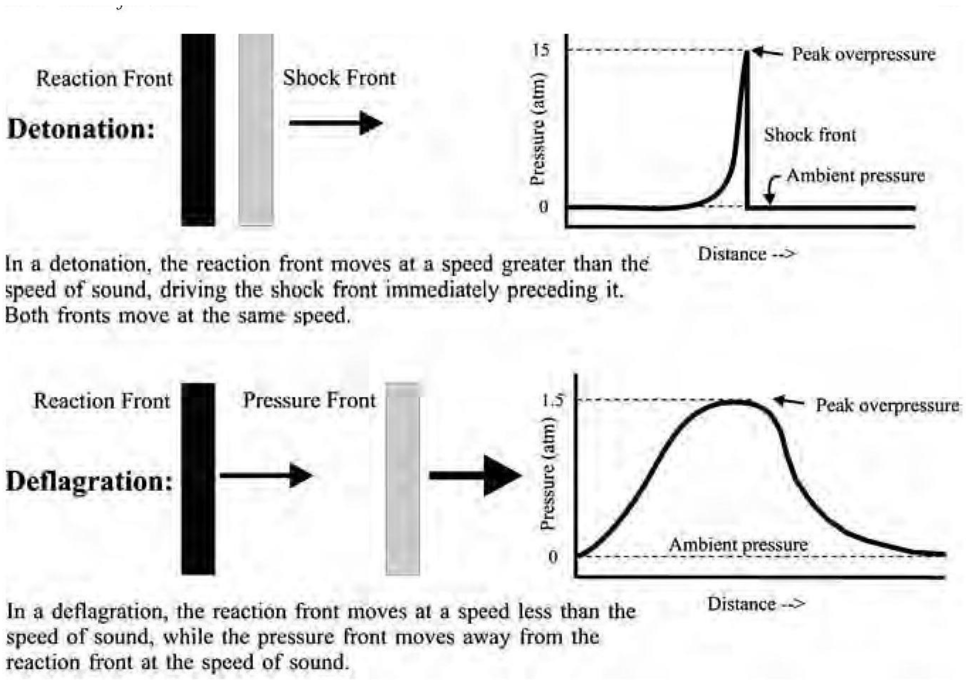

Reaction Front
Shock Front
Detonation:
15
Pressure (atm)
0
Peak overpressure
Shock front
Ambient pressure
Distance -->
In a detonation, the reaction front moves at a speed greater than the speed of sound, driving the shock front immediately preceding it. Both fronts move at the same speed.
Reaction Front
Pressure Front
Deflagration:
1.5
Pressure (atm)
0
Peak overpressure
Ambient pressure
Distance -->
In a deflagration, the reaction front moves at a speed less than the speed of sound, while the pressure front moves away from the reaction front at the speed of sound.

Figure 2.2. Comparison of detonation and deflagration gas dynamics. [3]

2.2 Vapor cloud explosion

By simply saying vapor cloud explosion (VCE) is combustion of flammable vapor with overpressure. More detailed requirements of such combustion to be classified as VCE are shown in Figure 2.3.

Flame acceleration is main part of VCE. Increase in this acceleration comes with turbulence stretches, that tears and increases flame front area. The cause of turbulence is motion of gas, due to flow turbulence as it flows ahead of flame front, while being pushed by increasing in volume combustion products or due to interaction between gas and obstacles on its way.

2.3 Explosion estimation models

To find hazards of fragments, generated as the result of explosion, overpressure and/or impulse of blast must be calculated. Due to complicate calculations and dependance of initial conditions to each other, three empirical methods are widely used to estimate blast parameters. They are: TNT equivalency, TNO multi-energy, Baker–Strehlow-Tang.

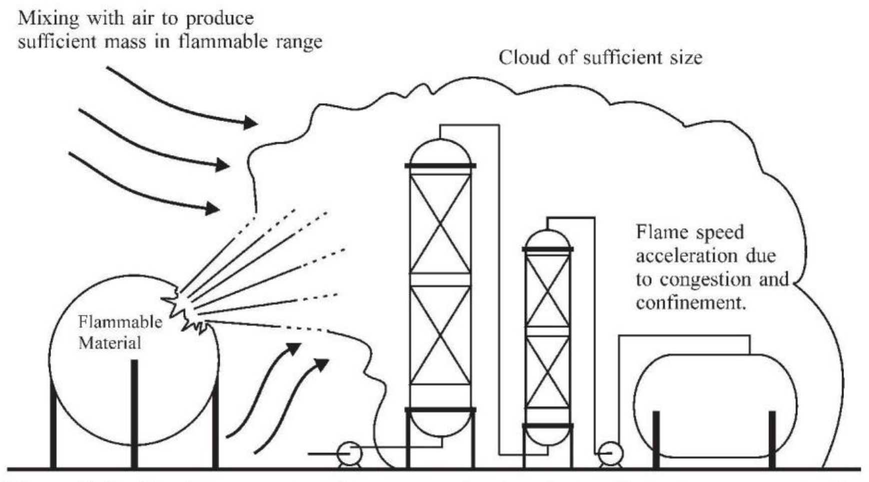


Figure 2.29. Conditions required for a vapor cloud explosion. An ignition source is also required, but these are abundant.

Figure 2.3. Graphical interpretation of required conditions for VCE. []

According to [4], nowadays there are actually many derivative methods, based on beforementioned three.

2.3.1 TNT Equivalency Method

The TNT equivalency model is very simple and is based on the assumption of equivalence between the flammable substance involved in the VCE, and an amount of TNT, in terms of its potential blast effects. The equivalent mass of TNT is found by following equation:

$$W = \frac{\eta m E_c}{E_{TNT}}, \tag{2.1}$$

where W - equivalent mass of TNT (mass),

η - empirical explosion efficiency (unitless),

m - mass of flammable gas (mass),

E_c - heat of combustion of the flammable gas (energy/mass),

E_{TNT} - energy of explosion of TNT (energy/mass).

But this method has some shortcomings such as: does not consider effects of confinement and congestion, assumes detonation, difficult to estimate center of the explosion.

2.3.2 TNO Multi-energy Method

TNO multi-energy is based on assumption that explosion more dependent on the level of confinement and less on the type of fuel. Severe blast effects are produced by the portions of vapor in partially confined or obstructed regions. The remaining unconfined parts will simply burn out without seriously contributing to the blast. Blast itself is assumed to be result of hemispherical steady flame speed, stoichiometric hydrocarbon-air explosion at ground level.

Theory

Firstly, volume of vapor cloud V_c, if only mass was given, and dimensionless scaled distance $\bar{R}$ are needed to be calculated:

$$V_c = Q_{ext}/(\rho \cdot c_s) \tag{2.2}$$

$$E = V \cdot 3{,}5\ MJ/m^3 \tag{2.3}$$

$$\bar{R} = \frac{R}{(E/P_0)^{1/3}} \tag{2.4}$$

where Q_{ext} – the quantity of gas, kg,

ρ – density of gas, kg/m^3,

c_s – the stoichiometric concentration, %vol,

E – combustion energy involved of stoichiometric mixture hydrogen-air, J,

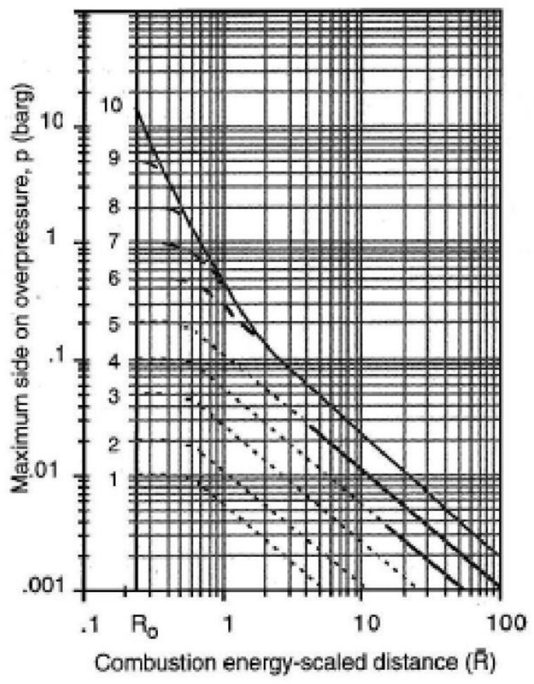
P_0 – ambient pressure, Pa,

Following equations are used to calculate positive phase duration and peak side-on overpressure in relation to relative distance:

$$t_+ = \frac{\bar{t}_+ \cdot (E/P_0)^{1/3}}{c_0} \tag{2.5}$$

$$\Delta P_S = P_0 \cdot \Delta \bar{P}_S \tag{2.6}$$

$$I = \frac{1}{2} \cdot t_+ \cdot \Delta P_S \tag{2.7}$$



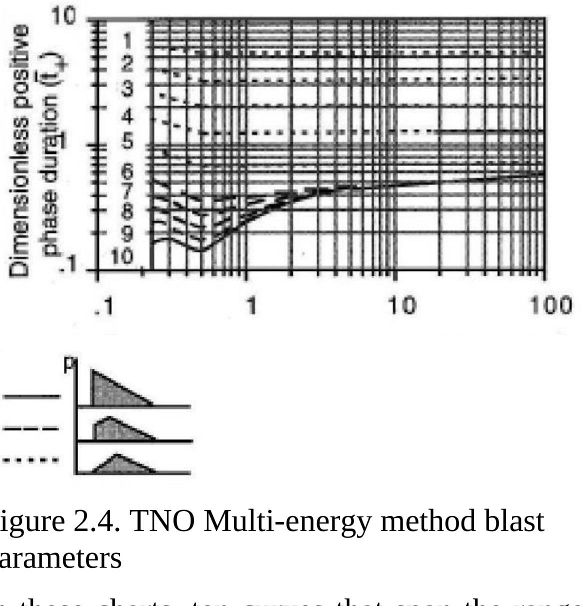


Figure 2.4. TNO Multi-energy method blast parameters

In these charts, ten curves that span the range of severities from mild deflagrations to detonations have been shown. Each curve is assigned an integer that indicates its severity. Thus, curve 1 means mild deflagration and 10 stands for detonation.

2.3.3 Baker–Strehlow-Tang Method

BST method has similarity with TNO multi-energy method and uses set of correlation curves of dimensionless pressure in relation to scaled distance. This method is based on constant flame velocity, so strength of blast is proportional to maximum flame speed achieved in cloud. baker et al. (1994) suggests from experimental data that flame speed can be correlated in relation to fuel reactivity, density of obstacles and confinement as parameters. In Table 2.1. shown possible combination of beforementioned parameters based on different forms of flame expansion. Flame velocities in table are represented in form of Mach number, relation of local flow velocity and local speed of sound, and values are maximum flame speed for each case.

Table 2.1. Flame speed in Mach number for Soft Ignition Blast (Baker, 2003) [3]

1D Flame Expansion Case		Obstacle density		
		High	Medium	Low
Reactivity	High	5.2	5.2	5.2
	Medium	2.27	1.77	1.03
	Low	2.27	1.03	0.294

2D Flame Expansion Case		Obstacle density		
		High	Medium	Low
Reactivity	High	DDT	DDT	0.59
	Medium	1.6	0.66	0.47
	Low	0.66	0.47	0.079

2.5D Flame Expansion Case		Obstacle density		
		High	Medium	Low
Reactivity	High	DDT	DDT	0.47
	Medium	1.0	0.55	0.29
	Low	0.50	0.35	0.053

3D Flame Expansion Case		Obstacle density		
		High	Medium	Low
Reactivity	High	DDT	DDT	0.36
	Medium	0.50	0.44	0.11
	Low	0.34	0.23	0.026

2.4 Fragment formation

Hazards from gas explosions includes high temperature, overpressure, dynamic pressure, blast wave. More of concern are fragments and debris from applied overpressure due to being more hazardous to people [8]. Human injury from blast debris depends on many factors. These include velocity, angle of impact, size, density, mass and the portion of the body involved.

The main force behind fragments and debris formation is load on object(-s) from blast wave, so two conditions are taken into consideration: 1) the loading, acting on surface of object and 2) response of object to the loading (displacement, breaking down, etc.).

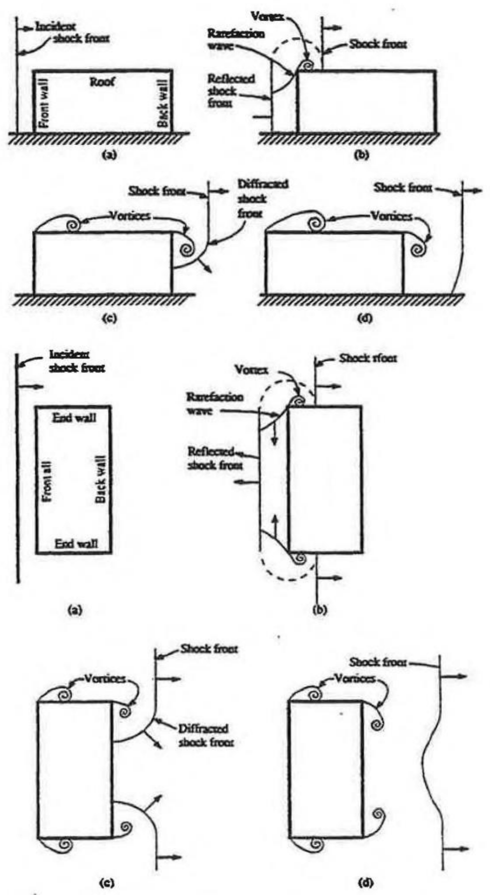


Figure 2.5. Schematic representation of the disturbance of the blast caused by a structure. []

Fragments generated from explosion can be divided to two types. *Primary fragment* denoted fragments of container or casing of explosive source. If the source is high explosive, then casing usually raptures to many small fragments with velocities up to several kilometers per second. *Secondary fragments* are fragments from nearly located objects, that get damaged due to overpressure and impulse of blast wave.

2.5 Glass fragments

Glass is a popular building material used in many structures. It is very brittle and fragile compared to most of other building materials, so it is very vulnerable to extreme loads such as explosion. According to [], glass shard missiles cause most of injuries on human body and casualties.

A large number of experiments have been carried out on glass windows under blast loading. Nevertheless, for security concern, most of these tests are still confidential and not for public access.

For simplicity, pressure at which window-pane might break is taken in calculation as 2 kPa and average value with 50% of windows broken as 5 kPa. These values are found as results of many experiments.

But in reality, there are also many parameters that determines the actual pressure at which window-pane breaks.[9]

Brittleness of glass is affected by how window was handled, cleaned and were exposed to wind. These parameters might show the damage degree on glass surface as microcracks.

Static fatigue due to continuous load on window-pane also decreases glass strength. Addition to this load, angle and distance from explosion influences on breakage of window.

3 Software setup

By term of “calculator” in this work going to be defined Python code, written in purpose to estimate VCE of stoichiometric hydrogen-air mixture and probability of further hazardous effects on structure and human.

3.1 Structure.

The calculator consists of 2 files, *Blast_calculator.py* and *probit.json* and 2 folders, *overpressure* and *duration*. *Blast_calculator.py* is base script of this calculator with defined functions. *Probit.json* includes probit values for each percent of possibility as shown in Table 3.1. Folders *overpressure* and *duration* have 10 different .json format files in each. In these files are written values of dimensionless side-on overpressure and dimensionless positive phase duration for each 10 blast strengths of TNO Multi-energy method.

Table 3.1. Relationship between probabilities and probits [5]

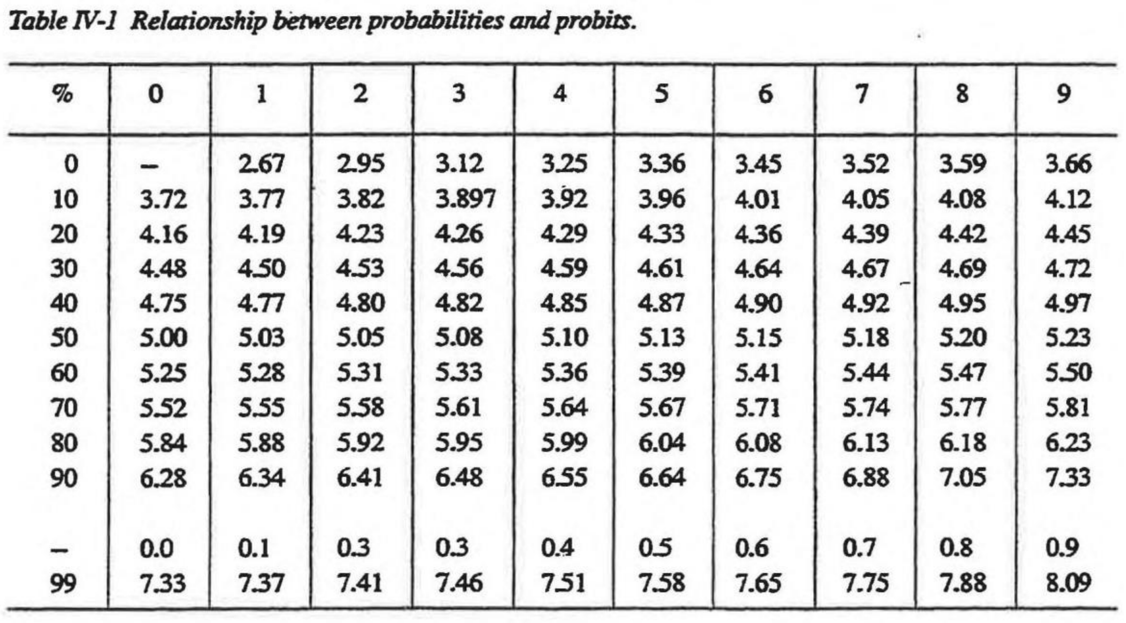


Table IV-1 Relationship between probabilities and probits.

%	0	1	2	3	4	5	6	7	8	9
0	–	2.67	2.95	3.12	3.25	3.36	3.45	3.52	3.59	3.66
10	3.72	3.77	3.82	3.897	3.92	3.96	4.01	4.05	4.08	4.12
20	4.16	4.19	4.23	4.26	4.29	4.33	4.36	4.39	4.42	4.45
30	4.48	4.50	4.53	4.56	4.59	4.61	4.64	4.67	4.69	4.72
40	4.75	4.77	4.80	4.82	4.85	4.87	4.90	4.92	4.95	4.97
50	5.00	5.03	5.05	5.08	5.10	5.13	5.15	5.18	5.20	5.23
60	5.25	5.28	5.31	5.33	5.36	5.39	5.41	5.44	5.47	5.50
70	5.52	5.55	5.58	5.61	5.64	5.67	5.71	5.74	5.77	5.81
80	5.84	5.88	5.92	5.95	5.99	6.04	6.08	6.13	6.18	6.23
90	6.28	6.34	6.41	6.48	6.55	6.64	6.75	6.88	7.05	7.33
–	0.0	0.1	0.3	0.3	0.4	0.5	0.6	0.7	0.8	0.9
99	7.33	7.37	7.41	7.46	7.51	7.58	7.65	7.75	7.88	8.09

In Python code defined following functions:

- to sort within arrays to find closest numbers (in “+” and “-” directions) for later usage in interpolation formulas;
- to calculate heat of combustion of stoichiometric hydrocarbon – air mixture from mass;
- to calculate scaled distance from Eq. 2.4;
- to open corresponding .json files for overpressure and positive phase duration for given distance or to whole way of blast to available in graphs values in relation to scaled distance;
- to calculate impulse of blast as given in Eq. 2.7;

Software setup

- to plot values of overpressure, impulse and positive phase duration in ordinate and scaled distance in abscissa;
- to find some necessary values as fragment velocity to use in further probit functions calculations;
- to calculate probit functions of damage to structure (building), probability of windows breakage and possibility of survival of human after been hit by fragment of different mass given by user.

Probit functions are taken from [5] because such works with derived functions for damage to buildings, fragments formation are very few.

The probit for structural damage is given in Eisenberg []:

$$Y = -23.8 + 2.92\ln{(P_s)} \quad (3.1)$$

The probit for structural damage is given in Green book [] for 1% probability of breakage at pressure 1 kPa (for old windows):

$$Y = -11.97 + 2.12\ln{(P_s)} \quad (3.2)$$

The probit for survival possibility is given in Green book [] in dependance to mass of fragment and velocity:

$$Y = -13.19 + 10.54\ln{(V_0)} \quad (3.3)$$

4 Results

As example to check how does written Python code performs were done 2 types of calculations. Firstly, were done comparison of side-on overpressure, impulse and P-I graph for 4 different masses of hydrogen, 0.1, 1, 10 and 100 kilograms of hydrogen for detonation. Second comparison were done for window breakage probability for 3 different blast strengths, detonation (according to multi-energy method's graph is 10), 1 bar deflagration (7) and 0.1 bar deflagration (4). For easier understanding of these calculations were used Matplotlib library in Python programming language. Due to lack of experience in working with Python, I have made different scripts for plotting.

The result of side-on overpressure - distance, impulse – distance and overpressure – impulse relations for 4 masses is shown in Figure 4.1 , Figure 4.2 and Figure 4.3, respectively.

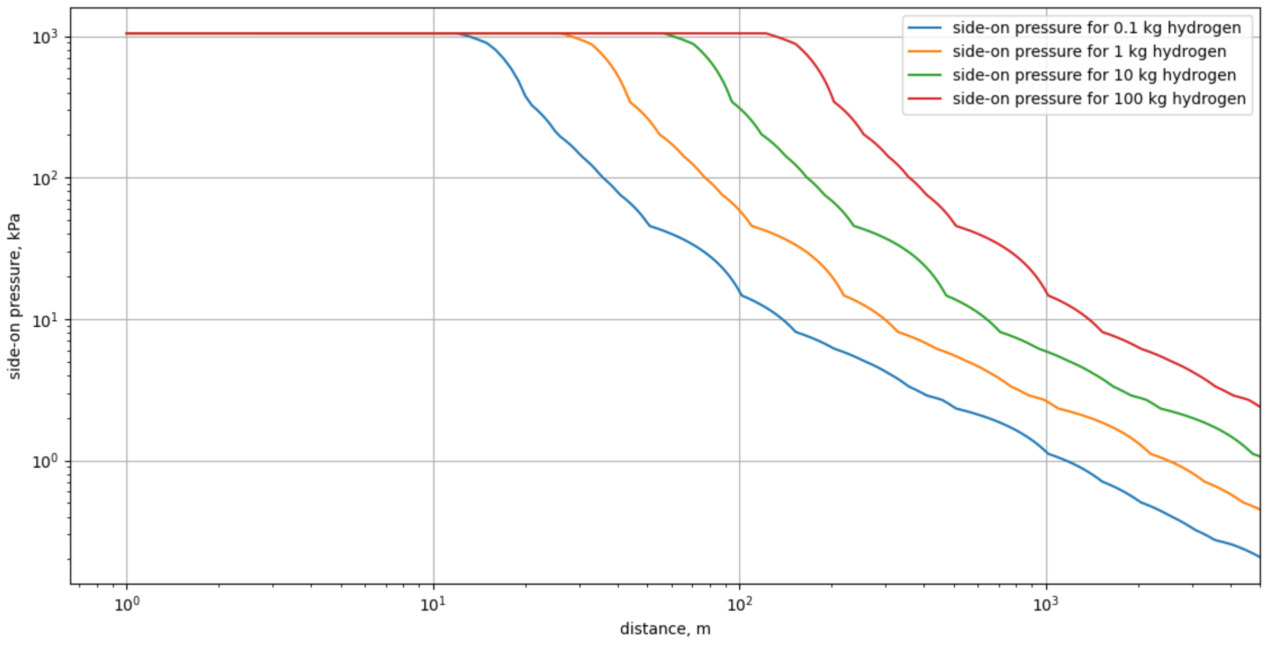


Figure 4.1. Side-on overpressure and distance relation in log scale.

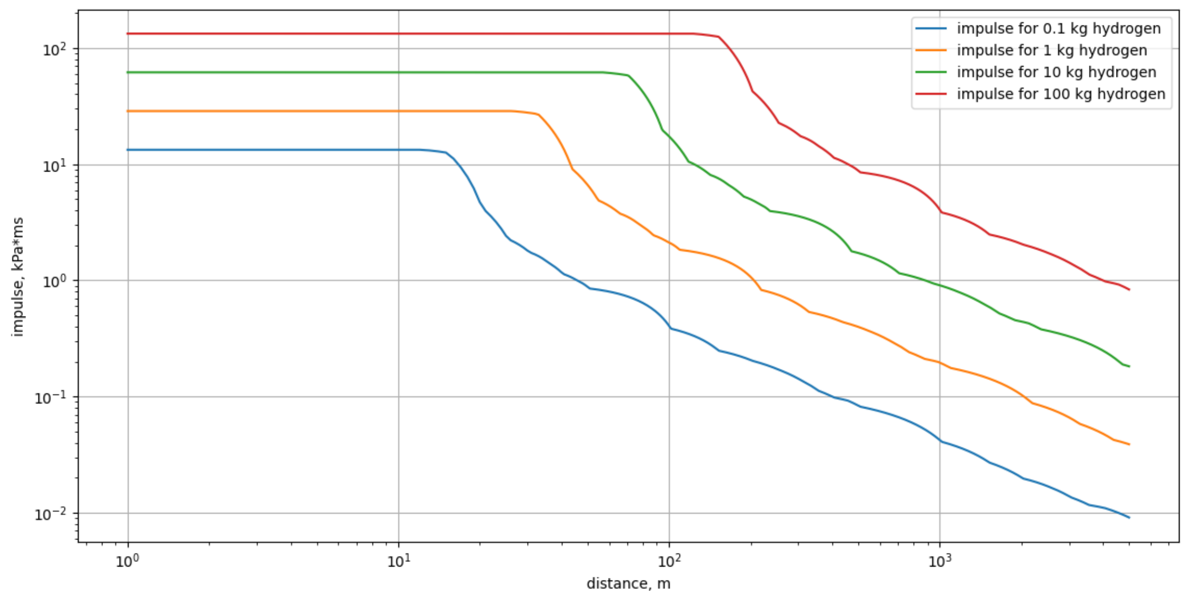


Figure 4.2. Impulse and distance relation in log scale.

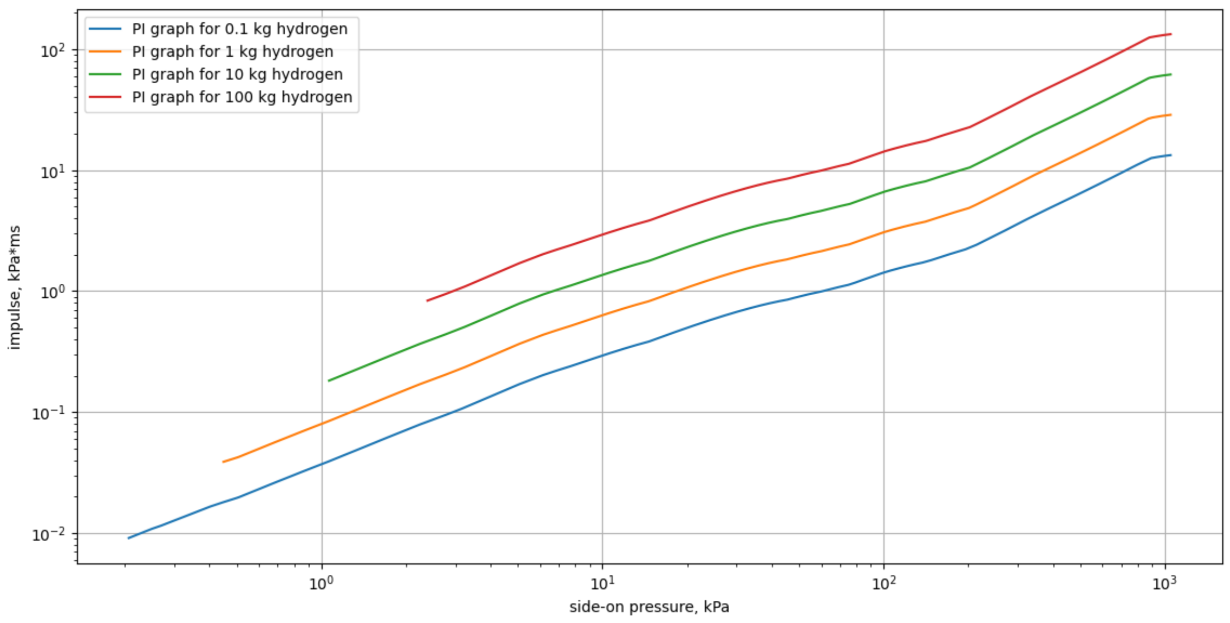


Figure 4.3. Overpressure – impulse (P-I) graph.

As can be seen from graphs, change in overpressure is directly proportional to mass. Mass of gas in order to be stochiometric defines gas cloud sizes. Difference comes from different cloud volumes and energy of explosion which is related to scale distance calculations.

According to damage estimations on common structures [8], 2 kPa is assumes as “Safe distance” due to probability 0.95 of no serious damage below this pressure value with projectile limit, some damage to house ceilings and 10% window glass breakage. So from Figure 4.1. can be found safe distances for each mass and they are: for 0.1 kg of hydrogen is

~ 620 m, for 1 kg of hydrogen – 1700 m., for 10 kg of hydrogen – 3000 m. and for 100 kg – more than 5000 m.

This was one way to estimate safe distance. Another way is to use overpressure-impulse (P-I) graph. According to 'Green book' [5], there are 3 major thresholds with following lower limits: 1 threshold/or minor structural damage – P = 4.6 kPa and I = 110 Pa·ms; 2 threshold/or major structural damage - P = 17.5 kPa and I = 290 Pa·ms; 3 threshold/or partial demolition. 50 or 75% of walls destroyed or unsafe - P = 40 kPa and I = 460 Pa·ms. From Figure 4.3 after applying these limits can be seen that distance of 5000 m in calculations are not enough for 100 kg of hydrogen to find safe zone. By using P-I graph with combination of overpressure – distance or impulse – distance can be determined approximate limits of three thresholds for each mass of fuel.

From impulse values can be obtained approximate distance of flight of fragment via assuming mass of fragment. But due to unknown parameters as angle of flight, surface area, body part, etc. this estimations are too vague, so estimated distance from P-I diagram is more reliable.

The result of comparison of window breakage for 3 different types of explosions is shown in Figure 4.4.

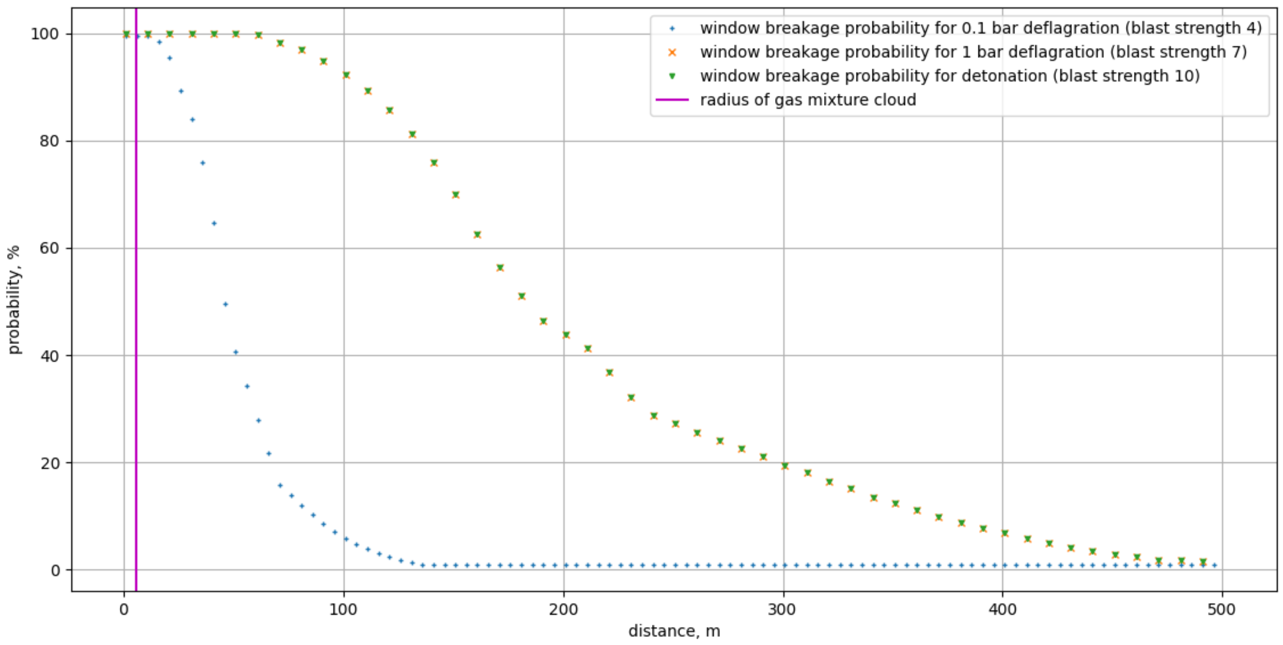


Figure 4.4. Windows breakage probability by distance

The results for 1 bar deflagration and detonation are same and overlap due to small distance of difference for blast strength higher than 6-7 and at certain distance blast becomes independent of its initial explosion energy.

5 Conclusion

The very difference of explosion between VCE and high explosive or missile is difference in duration, which defines impulse of explosion. Also, in comparison to other natural gases, explosion from hydrogen mixture produces very high values of overpressure [7], so applied equations are enough only to show approximate effects of fragments, for better and complete simulation different hazard estimation models should be added, as well as more details.

High overpressure produces many little in size and mass fragments, so simplified calculations as in this work are not enough without experiments to find behavior of hydrogen-air mixture in VCEs.

References

[1] S. Lewis, “An Overview of Leading Software Tools for QRA,”.

[2] Encyclopedia Britannica. In: (). url: https://www.britannica.com/science/hydrogen/Reactivity-of-hydrogen.

[3] S. Mannan, “Lees' Loss Prevention in the Process Industries” (Fourth Edition)

[4] Bjerketvedt et. al, “Gas explosion handbook”. In: Journal of Hazardous Materials, 1997

[5] Methods for the determination of possible damage. In: CPR16E. The Netherlands Organization of Applied Scientific Research; 1989.

[6] Guidelines for chemical process quantitative risk analysis, center for chemical process safety. American Institute of Chemical Engineers; 2000.

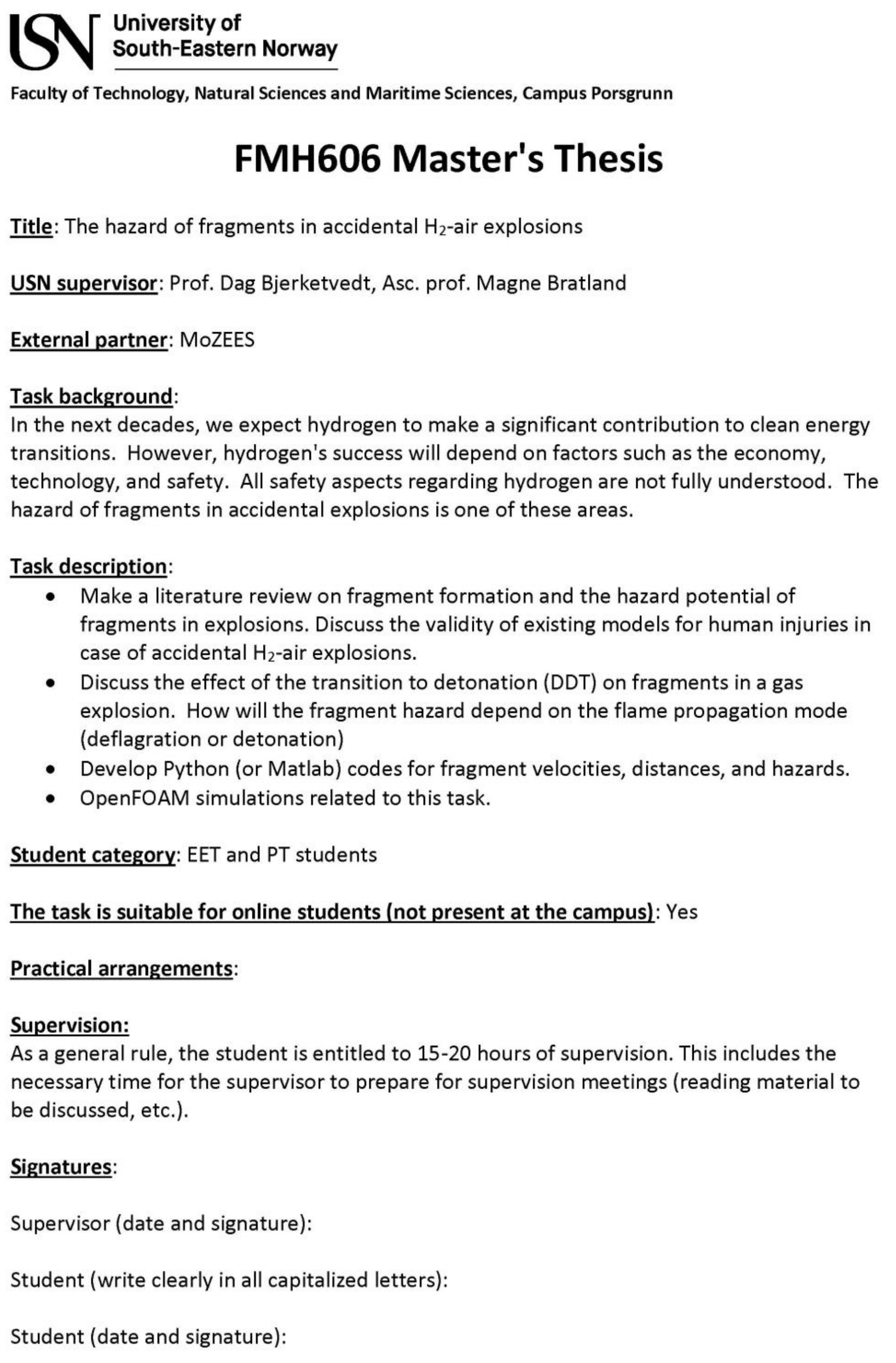
[7] Lian et. al, “Methods for estimating fragment hazard in gas explosion”. In: Journal of Hazardous Materials, 2020

[8] Daniel A. Crowl, “Fundamentals of Fires and Explosions,” in *Understanding Explosions*, John Wiley & Sons, Ltd, 2003, pp. 9–112. doi: https://doi.org/10.1002/9780470925287.ch2.

[9] D. K. Pritchard, “Breakage of glass windows by explosions,” *Journal of Occupational Accidents*, vol. 3, no. 2, pp. 69–85, May 1981, doi: 10.1016/0376-6349(81)90001-8.

Appendix A. Thesis task description

Appendix A. Thesis task description



University of
South-Eastern Norway

Faculty of Technology, Natural Sciences and Maritime Sciences, Campus Porsgrunn

FMH606 Master's Thesis

Title: The hazard of fragments in accidental H_2-air explosions

USN supervisor: Prof. Dag Bjerketvedt, Asc. prof. Magne Bratland

External partner: MoZEES

Task background:
In the next decades, we expect hydrogen to make a significant contribution to clean energy transitions. However, hydrogen's success will depend on factors such as the economy, technology, and safety. All safety aspects regarding hydrogen are not fully understood. The hazard of fragments in accidental explosions is one of these areas.

Task description:
- Make a literature review on fragment formation and the hazard potential of fragments in explosions. Discuss the validity of existing models for human injuries in case of accidental H_2-air explosions.
- Discuss the effect of the transition to detonation (DDT) on fragments in a gas explosion. How will the fragment hazard depend on the flame propagation mode (deflagration or detonation)
- Develop Python (or Matlab) codes for fragment velocities, distances, and hazards.
- OpenFOAM simulations related to this task.

Student category: EET and PT students

The task is suitable for online students (not present at the campus): Yes

Practical arrangements:

Supervision:
As a general rule, the student is entitled to 15-20 hours of supervision. This includes the necessary time for the supervisor to prepare for supervision meetings (reading material to be discussed, etc.).

Signatures:

Supervisor (date and signature):

Student (write clearly in all capitalized letters):

Student (date and signature):

Appendix B. Software repository

duration	Folder contains values from TNO Multi-energy positive phase duration curves, 10 files, each per blast strength.
overpressure	Folder contains values from TNO Multi-energy overpressure curves, 10 files, each per blast strength.
Blast_calculator.py	Main Python script with all the functions.
probit.json	Contains values to convert probits to possibility percentage.

GitHub repository link:

https://github.com/YMD2/master_thesis.git

Appendix C. Python code

This code might be updated to fix possible issues after submitting Master Thesis, so latest version will be available in Github repository from Appen. B

```
import math
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import argparse
import json
import sys
import bisect

atmospheric_pressure_default = 101250
a = 340 #m/s
P_a = 101325 #Pa
path = os.getcwd()

def sorting(arr, var):
    pos = bisect.bisect_left(arr, var)
    if pos == 0:
        return arr[0]
    if pos == len(arr):
        return arr[-1]
    before = arr[pos-1]
    after = arr[pos]
    return [after, before]

def energy(mass):
    volume = mass*11.2*100/29.5
    return 3500000*volume

def scaled_dist(dist=None):
    if dist==None:
        n = 100/((P_a/energy(mass))**(1/3))
        dist = np.arange(1, n+1, 1, dtype=float)
    scaled_distance_coeff = ((P_a/energy(mass))**(1/3))
    scaled_distance = np.multiply(dist, scaled_distance_coeff)
    return scaled_distance

def overpressure(dist=None):
    P_file = json.load(
        open(path + "\\overpressure\\{}.json".format(blast_strength), 'r'))
    P_df = pd.DataFrame(P_file)
    P_arr = P_df.to_numpy()
    P_arr=P_arr.astype(np.float32)

    if dist == None:
        k = np.interp(scaled_dist(), P_arr[:, 0], P_arr[:, 1])
        overpressure = np.multiply(k, P_a)
    else:
        s = sorting(P_arr[:,0], dist)
        ar0 = P_arr[:, 0]
        ar1 = P_arr[:, 1]
        v=[]
        for i in s:
            ind=np.where(ar0 == i)
            v.append(ar1[ind])
        k = v[0] + (scaled_dist(dist)- s[0])* (v[1]-v[0])/(s[1]-s[0])
        overpressure = np.multiply(k, P_a)
    return overpressure

def duration(dist=None):
    T_file = json.load(
        open(path + "\\duration\\{}.json".format(blast_strength), 'r'))
    T_df = pd.DataFrame(T_file)
    T_arr = T_df.to_numpy()
    T_arr = T_arr.astype(np.float32)

    calc1 = ((energy(mass)/P_a) ** (1/3)) * 1/a
    if dist == None:
        l = np.interp(scaled_dist(), T_arr[:, 0], T_arr[:, 1])
        positive_phase_duration = np.multiply(calc1, l)
    else:
        s = sorting(T_arr[:, 0], dist)
        ar0 = T_arr[:, 0]
        ar1 = T_arr[:, 1]
        u=[]
        for i in s:
            ind=np.where(ar0 == i)
            u.append(ar1[ind])
        l = u[0] + (scaled_dist(dist) - s[0]) * (u[1]-u[0])/(s[1]-s[0])
        positive_phase_duration = np.multiply(calc1, l)
    return positive_phase_duration

def impulse(dist=None):
    impulse = np.multiply(0.5*overpressure(dist), duration(dist))
    return impulse

def plotting(x, y, ax=None, plt_kwargs={}):
    if ax is None:
        ax = plt.gca()
    ax.plot(x,y,**plt_kwargs)
    return(ax)

def probit_for_structure_damage(dist):
    sd = scaled_dist(dist)
    ovp = overpressure(sd)
    imp = impulse(sd)
    if ovp <= 4600 and imp <= 110:
        V = ((4600/ovp)**3.9)+((110/imp)**5)
        return 5 - 0.26*np.log(V)
    elif ovp >= 40000 and imp >= 460:
        V = ((40000/ovp)**7.4)+((460/imp)**11.3)
        return 5 - 0.22*np.log(V)
    else:
        V = ((17500/ovp)**8.4)+((290/imp)**9.2)
        return 5 - 0.26*np.log(V)

def window_breakage(dist):
    sd = scaled_dist(dist)
    ovp = overpressure(sd)
    return -11.97+2.12* np.log(ovp)

def frag_velocity(dist, mass_of_fragm):
    sd = scaled_dist(dist)
    ovp = overpressure(sd)
    imp = impulse(sd)
    P_r = np.empty_like(ovp)
    P_r.append(2*ovp + ((2.4*ovp**2)/(0.4*ovp)+2*1.4*P_a))
    imp = (P_r/ovp)*imp
    return imp/mass_of_fragm

def probit_for_fragment_hazard(velocity, mass_of_fragm):
    if mass_of_fragm>0.001 and mass_of_fragm < 0.1:
        s=mass_of_fragm*(velocity**5.115)
        return -29.15+2.10*np.log(s)
    elif mass_of_fragm > 0.1 and mass_of_fragm < 4.5:
        s = 0.5*mass_of_fragm*(velocity**2)
        return -17.56+5.30*np.log(s)
    elif mass_of_fragm > 4.5:
        return -13.19+10.54*np.log(velocity)

def percentage(probit):
    pb_file = json.load(open(path + "\\probit.json", 'r'))
    pb_df = pd.DataFrame(pb_file)
    pb_arr = pb_df.to_numpy()
    pb_arr = pb_arr.astype(np.float32)
    prb = np.interp(probit, pb_arr[:, 0], pb_arr[:, 1])
    return prb

if __name__ == '__main__':
    m = input("Enter mass of hydrogen: ")
    if m.isnumeric()==True:
        mass = float(m)
    else:
        print("Wrong input. Mass will be set as 1 kg by default")
        mass = 1
    blast_strength = input("Enter blast strength (from 1 to 10): ")
    if blast_strength.isnumeric() == False and blast_strength > 10:
        print("Wrong input. Blast strength will be set to 10 by default")
        blast_strength = 10
    else:
        pass
    d = input("Enter distance between wall and explosion center: ")
    if d.isnumeric() == True:
        dist = float(d)
        pb1 = round(percentage(float(probit_for_structure_damage(dist))), 3)
        if pb1<=99.9:
            print(f'The probability of damage to building itself is {pb1}%')
        elif pb1>99.9:
            print('The probability of damage to building itself is higher than
 99.9%')
        pb2 = round(percentage(float(window_breakage(dist))),3)
        if pb2 <= 99.9:
            print(f'The probability of windows breakage is {pb2}%')
        elif pb2 > 99.9:
            print('The probability of windows breakage is higher than 99.9%')
        mof = input(
            "Enter mass of fragment to estimate its velocity in kilograms: ")
        if mof.isnumeric() == True:
            mass_of_fragm = float(mof)
            velo = frag_velocity(d, mof)
            pb3 = round(percentage(
            float(probit_for_fragment_hazard(velo, mof))), 3)
        if pb3 <= 99.9:
            print(
                f'The probability of survival after collision with fragment wi
th mass {mass_of_fragm} is {pb3}%')
        elif pb3 > 99.9:
            print(
                f'The probability of survival after collision with fragment wi
th mass {mass_of_fragm} is higher than 99.9%')
    else:
        print("Wrong input")
        pass
    pl=input("Are plots of overpressure and impulse necessary? y/n ")
    if pl.lower == 'y':
        x = scaled_dist()/((P_a/energy(mass))**(1/3))
        plotting(x, overpressure()/1000, title='side-on overpressure, kPa')
        plt.plot(x, duration(), title='positive phase duration, s')
        plt.plot(x, impulse()/1000, title='impulse, kPa*s')
        plt.grid()
        plt.show()
    elif pl.lower == 'n':
        sys.exit(0)
```

Due to lack of experience in making functions with several *args and **kwargs, there written two different scripts only for plotting.

1. To plot side-on overpressure, impulse and P-I graph for different masses as arguments in functions:

```
import math
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import json
import matplotlib.pyplot as plt

a = 340  # m/s
P_a = 101.325  # kPa
path = os.getcwd()
blast_strength = 10
dist = np.arange(1, 5001, 1, dtype=int)

def energy(mass):
    volume = mass*11.2*100/29.5
    return 3500000*volume

def scaled_dist(mass):
    scaled_distance_coeff = ((P_a/energy(mass))**(1/3))
    scaled_distance = np.multiply(dist, scaled_distance_coeff)
    scaled_distance=np.where(scaled_distance < 0.24, 0, scaled_distance)
    return scaled_distance

def overpressure(mass):
    P_file = json.load(
        open(path + "\\overpressure\\{}.json".format(blast_strength), 'r'))
    P_df = pd.DataFrame(P_file)
    P_arr = P_df.to_numpy()
    P_arr=P_arr.astype(np.float32)
    k = np.interp(scaled_dist(mass), P_arr[:, 0], P_arr[:, 1])
    overpressure = np.multiply(k, P_a)
    return overpressure


def duration(mass):
    T_file = json.load(
        open(path + "\\duration\\{}.json".format(blast_strength), 'r'))
    T_df = pd.DataFrame(T_file)
    T_arr = T_df.to_numpy()
    T_arr = T_arr.astype(np.float32)

    calc1 = ((energy(mass)/P_a) ** (1/3)) * 1/a
    l = np.interp(scaled_dist(mass), T_arr[:, 0], T_arr[:, 1])
    positive_phase_duration = np.multiply(calc1, l)
    return positive_phase_duration

def impulse(mass):
    impulse = np.multiply(0.5*overpressure(mass), duration(mass))
    return impulse

'''df = pd.DataFrame({'distance': dist})
df['for 0.1 kg'] = overpressure(0.1)
df['for 1 kg'] = overpressure(1)
df['for 10 kg'] = overpressure(10)
df['for 100 kg'] = overpressure(100)
print(scaled_dist(10))
df.to_excel('Table of results of side-on overpressure.xlsx')'''

plt.loglog(dist, overpressure(0.1), label='side-
on pressure for 0.1 kg hydrogen')
plt.loglog(dist, overpressure(1), label='side-on pressure for 1 kg hydrogen')
plt.loglog(dist, overpressure(10), label='side-
on pressure for 10 kg hydrogen')
plt.loglog(dist, overpressure(100), label='side-
on pressure for 100 kg hydrogen')
plt.xlabel('distance, m')
plt.ylabel('side-on pressure, kPa')
plt.legend()
plt.grid()
plt.show()

plt.loglog(dist, impulse(0.1),
           label='impulse for 0.1 kg hydrogen')
plt.loglog(dist, impulse(1), label='impulse for 1 kg hydrogen')
plt.loglog(dist, impulse(10), label='impulse for 10 kg hydrogen')
plt.loglog(dist, impulse(100),
           label='impulse for 100 kg hydrogen')
```

```
plt.xlabel('distance, m')
plt.ylabel('impulse, kPa*ms')
plt.legend()
plt.grid()
plt.show()

plt.loglog(overpressure(0.1), impulse(0.1), label='PI graph for 0.1 kg hydroge
n')
plt.loglog(overpressure(1), impulse(1), label='PI graph for 1 kg hydrogen')
plt.loglog(overpressure(10), impulse(10), label='PI graph for 10 kg hydrogen')
plt.loglog(overpressure(100), impulse(100), label='PI graph for 100 kg hydroge
n')
plt.xlabel('side-on pressure, kPa')
plt.ylabel('impulse, kPa*ms')
plt.legend()
plt.grid()
plt.show()
```

2. To plot window breakage probability from probit function for different blast strengths as arguments in functions:

```
import math
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import json
import matplotlib.pyplot as plt

a = 340  # m/s
P_a = 101325  # kPa
path = os.getcwd()
mass = 10
dist = np.arange(1, 501, 1, dtype=int)

def energy():
    volume = mass*11.2*100/29.5
    return 3500000*volume

def scaled_dist():
    scaled_distance_coeff = ((P_a/energy())**(1/3))
    scaled_distance = np.multiply(dist, scaled_distance_coeff)
    return scaled_distance

def overpressure(blast_strength):
    P_file = json.load(
        open(path + "\\overpressure\\{}.json".format(blast_strength), 'r'))
    P_df = pd.DataFrame(P_file)
    P_arr = P_df.to_numpy()
    P_arr = P_arr.astype(np.float32)
    k = np.interp(scaled_dist(), P_arr[:, 0], P_arr[:, 1])
    overpressure = np.multiply(k, P_a)
    return overpressure

def window_breakage(blast_strength):
    probit = -11.97+2.12 * np.log(overpressure(blast_strength))
    pb_file = json.load(open(path + "\\probit.json", 'r'))
    pb_df = pd.DataFrame(pb_file)
    pb_arr = pb_df.to_numpy()
    pb_arr = pb_arr.astype(np.float32)
    prb = np.interp(probit, pb_arr[:, 0], pb_arr[:, 1])
    return prb

radius_of_cloud = 0.24/((P_a/energy())**(1/3))

'''df = pd.DataFrame({'distance': dist})
df['scaled distance'] = scaled_dist()
df['for explosion strength 4'] =  window_breakage(4)
df['for explosion strength 7'] = window_breakage(7)
df['for explosion strength 10'] = window_breakage(10)

df.to_excel('Table of results of windows breakage.xlsx')'''

plt.plot(dist, window_breakage(4), '+', label='window breakage probability for
 0.1 bar deflagration (blast strength 4)', ms=3,markevery=5)
plt.plot(dist, window_breakage(7),'x', label='window breakage probability for
1 bar deflagration (blast strength 7)',ms=4,markevery=5)
plt.plot(dist, window_breakage(10),'v', label='window breakage probability for
 detonation (blast strength 10)',ms=3,markevery=5)

plt.axvline(x=radius_of_cloud, label='radius of gas mixture cloud', c='m')
plt.xlabel('distance, m')
plt.ylabel('probability, %')
plt.legend()
plt.grid()
plt.show()
```