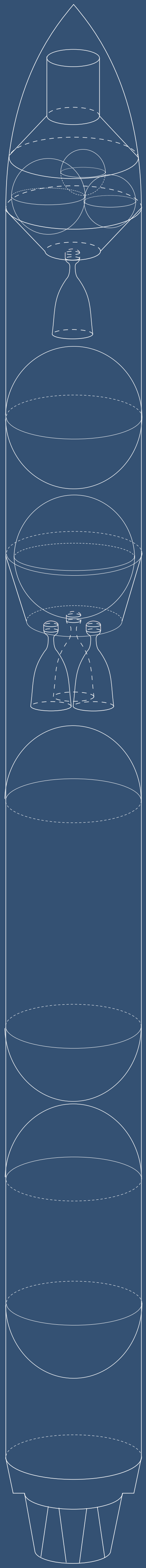
MASTER THESIS

SPACE ENGINEERING

LAUNCH VEHICLE STRUCTURAL MASS PREDICTION MODEL

W.A.R. Wildvank



Launch Vehicle Structural Mass Prediction Model
-

WILLEM ANTONIUS ROY WILDVANK

Student number: 4218523

To obtain the degree of Master of Science
at the Delft University of Technology

To be defended publicly on Thursday June 14, 2018 at 10:00 AM.

Assessment Committee:
Barry Zandbergen - Responsible thesis supervisor
Bertil Oving - Supervisor at the NLR
Prof. Eberhard Gill - Chair
Prof. Julien van Campen - Examiner from other research group

May 31, 2018

Abstract

Accurate dry mass prediction is essential in early design phases of a launch system. To predict structural mass of a launch vehicle more accurate at a conceptual design phase, a structural mass model was developed. The model predicts a launch vehicle's structural mass, thickness, center of mass and mass moment of inertia. Load carrying structures within in a conventional launch vehicle were modeled on a subsystem level by determining force and moment acting on the structure during flight. The model was developed for use in launch vehicle optimization algorithms, and thus relies on a relative small amount of calculations to obtain structural strength of parts. The model supports walls made of isotropic and orthotropic material, sandwich panels, rings and stiffeners. The model was compared to existing models and known data from existing launch vehicles. The results did indicate a good convergence for determining wall thickness when compared to values found in literature.

Keywords: mass model, launch vehicle design, launch vehicle optimization, rocket structures, buckling of rocket structures.

Overzicht

Het accuraat voorspellen van de droge massa is essentieel voor de vroege ontwerp phase van een lancheer systeem. Een model was ontwikkeld om de massa, dikte, zwaartepunt en traagheidsmoment van structurele sub-systemen van een raket te benaderen. Het model is gemaakt met het doel om het te gebruiken in optimalisatie algorithmes voor het ontwerpen van meerdere traps raketten die een vracht in een baan om de aarde of els in het zonne-stelsel te brengen. Het model ondersteunt meerdere soorten wand materialen en configuraties. Het rapport vergelijkt resultaten met bestaande modellen en data van bestaande lanceer voertuigen. Waardes berekent voor de dikte van de wanden door het model kwamen wel overeen met gevonden waardes in de literatuur. Na dit overzicht is de rest van het rapport geschreven in het engels.

Sleutel woorden: massa model, lanceer voertuig design, lanceer voertuig optimalisatie, raket structuren, knikken van raket structuren,

Foreword

First and foremost I would like to thank my supervisors, Barry Zandbergen and Bertil Oving for their guidance during the project. Their feedback was essential for the result of this research. I would also like to thank the NLR for given me the opportunity and a good environment to complete my master-thesis within her organization. I would like to thank my family and my girlfriend Ann Helen for always supporting me during my studies, I could not have done it without you.

This report contains research done by the author who is at time of writing a master student at the technical university of Delft. The research was completed in order to obtain a master of science in the field of space engineering at the university's aerospace faculty. The research was for a large portion completed at the Dutch aerospace center (NLR). The NLR provided the resources needed to complete the research such as guidance, facilities and expertise.

Contents

List of Figures

List of Tables

List of Terms

Definitions

Bulkhead Connection between two objects within a rocket stage or launch vehicle.

C++ Programming language.

Cube-sat Small standard sized cube shaped satellite

Dry stage mass Mass of a rocket stage when it is fully depleted from any propellant, e.g. main fuel, main oxidizer, RCS propellant, ullage engine propellant without the payload mass.

Inter-stage Structure connecting two stages together, which is jettisoned during separation.

Launch Vehicle All rocket stages and payload combined, i.e; all systems lifting off at Launch.

Rocket Engine All subsystems of a rocket stage that contribute to delivering the main propulsion force (not direction), excluding propellant and pressurization tanks and plumbing from the tanks to the Rocket engine.

Rocket Stage Every subsystem of a rocket stage including the inter-stage or fairing above the stage (but not the inter-stage below).

Tank End Cap End caps of a cylindrical tank which main purpose is to contain propellant.

Hull Structure between tanks or outside structure of non-load carrying propellant tanks.

Acronyms

CAD	Computer Assisted Design.
CFD	Computational Fluid Dynamics.
ESA	European Space Agency.
FEM	Finite Element Method.
GTOW	Ground Take Off Weight.
LEO	Low Earth Orbit.
LV	Launch Vehicle.
NLR	Nederlands Lucht en Ruimtevaart Centrum "Netherlands Aerospace Centre".
OOP	Object Oriented Programming.
SMILE	SMall Innovative Launcher for Europe.
TUDAT	TU Delft Astrodynamics Toolbox.
CFRP	Carbon Fiber Reinforced Plastics.
SS	Stainless Steel.
LOX	Liquid Oxygen.
RP-1	Rocket Propellant 1.
LH2	Liquid Hydrogen.

Symbols

a	Semi-major axis ellipse, linear acceleration
b	Semi-minor axis ellipse
c	Specific heat
d	Diameter
D	Flexural stiffness per unit width
E	Young modulus
F	Force
g	gravitational parameter
Gr	Grasshoff number
h	Heat transfer coefficient, height
I	Moment of inertia
J	Joule
k	Thermal conductance
K	Kelvin
l	load factor
L	Length
m	Mass, number of buckle half waves in axial direction
M	Mass, Moment
$\mathscr{M}$	Molar Mass
n	Number of buckle waves in circumferential direction
N	Axial force per unit of circumferential width
Nu	Nusselt number
P	Pressure
Pr	Prandtl number
q	Heat flux, distributed load
r	Radius
Ra	Rayleigh number
R	Ideal gas constant
S	Surface, Shear
s	Seconds,number of stiffeners, slanted length cone
t	Thickness, time
T	Temperature
u	Number of rings
v	Velocity
V	Volume
x, y, z	Cartesian coordinates
α	Semi-vertex angle of a cone, angular acceleration
β	Thermal expansion coefficient, throttling parameter
γ	Heat capacity ratio, buckling correction factor
μ	Dynamic viscosity, Poisson ratio
ν	Kinematic viscosity
ρ	Density
σ	Normal stress
τ	Moment,torque
ϕ, φ	Polar coordinates
ω	Angular velocity

Subscripts

c	Cylinder	p	Propellant, pressure
cog	Center of gravity	pay	Payload
cr	Critical	rec	Recovery subsystem
D	Drag	r	Ring
dry	Dry (mass)	s	Sphere
e	Exhaust	sm	Structural material
E	Engine	st	Stiffener
f	Fuel	t	Tank
fa	Fairing	tf	Thrust-frame
He	Helium	T	Thrust
i	Insulation	$u.p.$	Unused propellant
is	inter-stage	vac	Vacuum
ox	Oxidizer	$yield$	Yield (stress)

Chapter 1

Introduction

This report is part of the master thesis phase of the Space Engineering master track at the TU-Delft. This report functions as an embodiment of the work executed and aims to fulfill the requirement for successfully completing the master phase. The topic of this research is a mass model for launch vehicles to estimate their dry mass for conceptual design phases. The research largely took place at the NLR(Nederlands Lucht en Ruimtevaart Laboratorium) in Marknesse, in cooperation with the aerospace faculty at the TU Delft.

The research project was planned and executed by the author of this report, i.e.; the student. The student was guided during the project by two supervisors:

Oving, Bertil Senior R&D engineer Small Space Systems - Nederlands Lucht en Ruimtevaart Centrum (Netherlands Aerospace Centre)

Zandbergen, Barry Lecturer Space System Engineer - TU Delft

Other than guidance the supervisors also evaluated the results produced. This evaluation also covered the determination of sufficient completion of the project(green light) together with a determination of the final grade.

The topic of this research is the development, verification and validation of a mass model for the structural mass in a launch vehicle. This Introduction chapter will continue with describing the background to the project and the problem statement to show how to research came to be. This continues with a description of the research methodology implemented, together with the scope of the project and a description of its limits. The introductory chapter will conclude with a plan of action which describes the iterations used to develop the model and a review of literature surveyed and used to develop the model.

The second chapter describes mass modeling of a launch vehicle and its subsystems. This chapter will break down the LV(launch vehicle) into several subsystems and defines these to make clear distinctions between them. The chapter also reviews how mass of subsystems other than the structural mass are modeled based upon existing mass models.

The third chapter describes the modeling of propellant and pressurization tanks. Since propellant tanks are such a large and heavy subsystem a separate chapter is dedicated to them. The chapter divides the tank further up into other subsystems and describes its design parameters.

The fourth chapter introduces determination of forces and moments acting on the launch vehicle. This includes the calculation of these forces at any point in the LV. Other than moment and forces it is also explained how the center of mass and mass moment of inertia are calculated.

The fifth chapter describes yielding criteria used to determine the thickness/geometry of the structure wall. This includes tensile stress but also extensive description of buckling criteria used for several kinds of wall structures and shapes.

The sixth chapter of the report describes the way the model is build up. This explains sequences, algorithms and methods used to determine the structural mass.

The last three chapters before the appendix show the results obtained from the model and its input variables(chapter seven). The verification process and results and an attempt at validating the model (chapter eight). And finally a conclusion which concludes the report and describes recommendations before the appendix starts.

I. Background

Before the background to the project is described please note that a large portion of background information concerning existing mass models for launch vehicles can also be found in the literature study prior to the thesis research[1] and in the proposal for this research.[2] These reports indicate the motivation for the development of a dry mass model in more detail.

Dry mass of a launch vehicle is a critical property for analyzing a rocket's performance. Together with its engine(s) exhaust velocity and propellant mass they make up the 'famous' Tsiolkovsky rocket equation or ideal rocket equation, see equation 1.1. In this equation M_p is the propellant mass, M_{dry} is the dry mass, v_e is the exhaust velocity and Δv is the change in velocity (Delta v) achieved after expulsion of all the propellant in the rocket. Delta v is often used to describe performance needed to perform certain (orbital) maneuvers, an example of this is the transition from earth's surface to LEO (low earth orbit), this is often stated as a delta v value of around $10[km/s]$.[3]

$$\Delta v = v_e ln\left(\frac{M_p + M_{dry} + M_{pay}}{M_{dry} + M_{pay}}\right) \tag{1.1}$$

Previous studies completed by master students at the TU Delft have focused on the modeling and optimization of launch vehicles. These models use various design parameters for the design of launch vehicles which include engine performance, trajectory, propellant choice and an estimation of dry mass. These models are than optimized for a certain payload and orbit to minimize the total launch vehicles mass. This is done by trajectory analysis and input of LV parameters. Dry mass estimation in these models is achieved by empirical relations taken from several studies, e.g. Zandbergen.[4]

[1]W.A.R. Wildvank. "Reusing Rocket Stages - A Literature Survey". MA thesis. TU Delft, 2017.

[2]Roy Wildvank. "Master thesis proposal - Launch Vehicle Structure Mass Model". In: (2017).

[3]B.T.C. Zandbergen. *AE1222-II: Aerospace Design and Systems Engineering Elements I*. 2015.

[4]BTC Zandbergen. "Aerospace Design and Systems Engineering Elements I, Part Launcher Design and Sizing". In: (2011).

Figure 1.1: *SMILE logo*

One of these studies was conducted by van Kesteren[5] who conducted research on solid propellant launch vehicles which are launched from the ground versus similar launch systems that are launched by releasing them from an aircraft at high altitude/velocity. Another similar research project was conducted by Miranda[6] which compared hybrid engine based vehicles. These studies used cost models to estimate launch cost for these vehicles, i.e. TRANSCOSTS.[7] These cost estimations are largely empirical relations in which propellant mass of the vehicle is an important parameter. Both studies used a multi-disciplinary optimization tool named Pagmo developed by ESA to optimize launch vehicle concepts to minimize GTOW (ground take of weight). The models in these studies were integrated into TUDAT (TU Delft Astrodynamics Toolbox) which is a large library written in C++ with various tools for astrodynamic calculations such as orbit determination.

The project is executed at the NLR and has overlapping areas with the SMILE project, a project in which the NLR has great involvement. SMILE stands for SMall Innovative Launcher for Europe and its goal is to develop a small and affordable launcher for relative small and light payloads. The project is a Horizon 2020 project spanning thirteen companies and institutes to design a small launch vehicle (around $50[kg]$ to LEO) to supply the small satellite and cube-sat market with a dedicated launcher. In current day, smaller satellites are often launched in combination with larger payloads (piggybacking), this cheap option often comes with restricted choice of orbit and launch date. Supplying the market with a dedicated launcher gives owners of smaller satellites more control over orbit injection and other launch related mission parameters. Launch costs is an important parameter for the SMILE project to "compete" with piggyback launches. Although a dedicated launcher does not directly compete with piggyback rides it cannot be significantly more expensive since the upsides will then not weigh up against the cost increase.[8]

This research project uses data from the SMILE project to verify the created model. This data is mostly in the form of FEM(finite element method) data and helps compare the model to more refined computer simulation.

[5]M. van Kesteren. "Air launch versus ground launch". MA thesis. TU Delft, 2013.

[6]F. Miranda, B.T.C. Zandbergen, and D. Dirkx. "Design Optimization of Ground and Air-Launched Hybrid Rockets:" 2015. URL: `Item%20Resolution%20URL%20http://resolver.tudelft.nl/uuid:832aa36d-041b-4896-97a7-837f91cdc5d6`.

[7]Dietrich E Koelle. *Handbook of Cost Engineering for Space Transportation Systems (revision 1) with Transcost 7.1: Statistical-analytical Model for Cost Estimation and Economical Optimization of Launch Vehicles*. TransCostSystems, 2003.

[8]AJP Van Kleef et al. "Innovative Small Launcher". In: (2015).

II. Problem Statement

The master theses projects mentioned in the previous section use models to calculate an optimized LV configuration. A large number of variables go into these models that relate to several aspects of a launch vehicle. These vary from flight trajectory to engine performance to aerodynamic analysis to mass modeling. These models are constructed to be helpful in the conceptual phase of a launcher design as well as to give insight in the difference between various launcher configuration. The configurations are optimized to minimize GTOW. Multi-disciplinary optimization should thus give a good indication of the size and mass of a LV, or at least a proper distinction between various concepts.

As mentioned in the previous section; current mass models are almost solely based upon empirical relations. In itself there is nothing wrong with the usage of empirical relations for approximating mass. An advantage of empirical data is that it is grounded in reality and can be very helpful to quickly determine a value once the relationships are established. A downside to using empirical data in the launch vehicle industry is often lack of available data. The combination of the total number of existing launch vehicles and availability of data for existing launchers makes it difficult to create precise relations. This sparse data creates relationships which often lack precision. As a consequence these formulas are inadvertently inaccurate up to a certain extent.

Another downside to approximating mass this way is the diminishing of variables which might be interesting for concept design trade-off or in researching key parameters for a LV design. An example of this can be choice of material for a certain subsystem over another material. Another example is change in geometric shape of parts to relax or intensify design parameters. An elliptical tank end cap can for example be made "flat" (high a/b) to limit the height of the tank or spherical ($a/b = 1$) to minimize wall thickness. Empirical relations do take into account key parameters for mass determination but this is in limited amount. As mentioned previously, available data is not abundant and thus dividing this data up even further to distinguish between more design parameters makes the statistical analysis sparse.

On the other side of the spectrum is detailed modeling, e.g. CAD,FEM,CFD. These methods are computational intensive and take a lot of time from the engineer, but should give accurate results. Theses methods are thus obviously helpful when one or a few concepts are chosen to be developed further and less so when the project is still trading-off various conceptual designs. This property makes these methods impractical for use in an optimization tool since calculating hundreds of concepts would take too much time and effort for the result gained.

The combination of the difficulties empirical mass model relations face and the importance of dry mass in the determination of a rockets performance gives a need for a more accurate mass model for rocket stages and launch vehicles which can be used during the conceptual phase of LV design. This is easier said than done since a LV consists of a large amount of widely different subsystems which all contribute to its overall mass. It would also be ambitious to attempt to have modeled all these subsystems within this research project.

A decision was made to focus on the structural components of a LV. This decision is based upon the assumption that structural mass is a subsystem that contributes significantly towards the total dry mass. Another reason was the assumed large effect structure mass has on changing design parameters of the LV or one of its subsystems. An example of this is propellant choice; propellant choice affects performance of the engine and thus thrust, tank volume and thus moment of inertia and propellant mass and thus thrust to weight ratio. All parameters also affect the required strength of the structure and thus its weight, even though it does not directly change its own parameters.

Structural mass was also assumed to be able to be modeled without implementing excessive empirical relations nor computational intensive calculations. This left the problem of creating a mass model for structural subsystems of a LV with refined accuracy over empirical methods but significantly less resource hungry than existing, more accurate methods such as finite element method.

III. Problem Analysis

Extending the problem statement, this section focuses on analyzing the problem to further specify the needs of the stakeholders. This section is build up of paragraphs each analyzing the problem further by identifying needs. These needs were used to define clear requirements in the following section. To start this section, identified needs are listed:

1. The model shall only require input available at a conceptual phase of a launcher design.
2. The model shall have computational time suitable for conceptual design optimization.
3. The model shall have more input than existing mass models (flexibility).
4. The model shall be as little as possible dependent on empirical data.
5. The model shall determine structure mass more accurate than existing methods.

1 The first need deals with input which is available at a conceptual level. This statement in itself is rather ambiguous since there is no clear line of what data is and isn't available at this point in a LV development process. Furthermore, the details of this need vary from project to project. Nonetheless is it important to restrict necessary input to a certain extend. This should make the model useful in scenarios were the LV only exist at a conceptual level. A list of input parameters which were deemed **not** necessary to run the model:

- Aerodynamic pressure distributions
- Aerodynamic coefficients
- Propellant dynamics (e.g. sloshing)
- Heat transfer in a LV (cryogenic propellant and aerodynamic heating)
- Attachment methods of structural components
- Acoustic loads
- Etc.

2 The second requirement has its focus on computation resources, this needs to ensure that the model is not as computation heavy as for example a detailed FEM analysis. Computation intensity is difficult to quantify as there are different kinds of computers and computer parts which excel at different kinds of calculations. This makes it difficult to formulate this requirement in one sentence in a SMART way.

As the need states; it has to be suitable for multi-disciplinary optimization. With this computation time can be limited to practical limits. Table 1.1 shows time units like days and weeks converted to seconds. This gives a basis for the amount of seconds it may take for the model to compute one launch vehicle. An optimizer for practicality was determined to run at a maximum time of one week ($10^5[s]$), a reasonable time of one day($10^4[s]$) and if under one hour (10^3) is considered quick.

Miranda[9] mentions several numbers of generations for a given optimizer to converge to a final solution. These values go up to the around 10^4 generations for some optimization algorithms. Assuming this number as a standard value the maximum time to compute one launch vehicle can determined. Dividing the maximum computation time by the number of generations gives the answer, thus:

bad - $t_{comp} < 60.5[s]$

acceptable - $t_{comp} < 8.64[s]$

good - $t_{comp} < 0.36[s]$

Where t_{comp} is the computation time to complete one launch vehicle. It was decided that this computation time needs to be achieved on computer systems designed for personal use. This means that this requirement cannot be satisfied with computer systems which are specially designed for simulation tasks, e.g. a supercomputer. To further comply with this need depending on the users needs, the model needs to be tune-able to a certain extend. This means accuracy needs to tune-able versus computation time.

Table 1.1: *Time units converted to seconds*

Time unit	Converted to seconds
second	$1.000 \cdot 10^0$
minute	$6.000 \cdot 10^1$
hour	$3.600 \cdot 10^3$
day	$8.640 \cdot 10^4$
week	$6.048 \cdot 10^5$
month	$2.592 \cdot 10^6$
year	$3.154 \cdot 10^7$

3 The third need described in the previous paragraph states that the model shall have more input variables than existing mass models. More input variables increase the flexibility of the model by increasing the number of parameters by which various concepts can be distinguished. This in itself is easy to verify. To show the actual content of this need in more detail, table 1.2 shows input parameters determined for the model compared to various existing models which are based on empirical data. It can be observed from the table that this model uses more input parameters than any other method described in this report.

4 The fourth need states; the model shall depend on statistical data as little as possible. This is a need that is difficult to verify. This need was implemented to combat the lack of available data other methods suffer from. To verify this, a sensitivity analysis has to be carried out to understand how much empirical relations affect the outcome of the model. In other words; error values of empirical relations used in the model can not have significant impact upon the final result.

5 The fifth need states that the structure mass has to be determined accurately. Providing supporting evidence for the accuracy is part of validating the model. The need stems from the goal to increase accuracy over existing models. This should be achieved if the value can stay within 10% of its actual real life value. Since this will be an increase over existing models which operate with larger error values.

[9]Miranda, Zandbergen, and Dirkx, see n. 6.

Table 1.2: *Comparison of input for calculation of dry or structural mass*

Design variable	This research	Zandbergen*	Akin	Apel
M_p	✓	✓		✓
M_{pl}	✓	✓		
# engines	✓	✓		
$F_{T.vac.}$	✓	✓	✓	
S_{fa}	✓	✓	✓	
M_{ox}	✓	✓	✓	
M_f	✓	✓	✓	
S_t	✓		✓	
M_{press}	✓	✓	✓	
Stage height	✓		✓	
Stage diameter	✓			
Expansion ratio			✓	
ΔV				✓
I_{sp}				✓
V_t	✓			✓
M_{rec}				✓
$M_{u.p.}$				✓
σ_{yield}	✓	✓		
ρ_p	✓			✓
ρ_{sm}	✓			
Buckling structure	✓			
Tank geometry	✓			
Stage geometry	✓			
Thrust frames	✓			
Payload frames	✓			
Inter-stages	✓			
$F_{inertia}$	✓			
F_D	✓			

* Zandbergen has more detailed mass models with more input parameters, but these focus on engine mass and not on structure mass.

IV. Research Methodology

This thesis research tries to follow research methodology as describes in book by Verschuren.[10] The research methodology applied can be read in more detail in the proposal for this research. This section will briefly summarize the methods applied.

The research objective Converting the problem formulated in the previous section to a clear objective was done to reflect on the success(or lack thereof) of the project after completion. A clear objective can be used to determine if the research was successful. The main objective was formulated in a way that it was restricted to one sentence. This was done to avoid a long and unclear objective which could be harder to verify. Its goal was also to create a good foundation upon which further requirements were build:

Develop and verify a launch vehicle structural mass model with increased accuracy and flexibility compared to existing conceptual models without significantly

[10]P.J.M. Verschuren et al. *Designing a research project*. The Hague : Eleven International Publishing, 2010.

compromising computation time.

Model Requirements: As mentioned; to avoid ambiguous interpretation of the research objective further requirements for the model were formulated. These requirements are, when applicable formulated in a SMART way (Specific, Measurable, Achievable, Realistic, Timed). Formulating requirements in such a way helps the verifying process tremendously and increases its usefulness. Described below are the main requirements for the structural mass model based upon the research objective.

1. Structural mass shall be calculated within $\pm 10\%$ of actual values.
2. Computation time shall be tune-able versus computation accuracy.
3. TUDAT compatibility
4. TUDAT independent

Requirements listed here were derived from earlier described needs in the previous section. Chapter 9 describes the verification of these requirements.

V. Research Framework

The research framework is illustrated in figure 1.2. The left side of the illustration shows knowledge which is necessary for proper implementation of the mass model. These sources of knowledge are combined to create the model displayed in the center of the graph. The model is than compared/confronted with other existing mass models which are displayed above and below the model. The result of the confrontation is simply stated as the "result of the analysis". This analysis compares both models on their respective performance based on the requirements for this model. This is done to illustrate the possible advantages and disadvantages of the new model. The figure mentions Hutchinsons model which has not been described up unto this point in the report. Hutchinsons model is a structural mass model similar to the model described in this report. It was thus deemed valuable towards the research to compare Hutchinsons model with this model. Although the model shares a similar approach there are certain differences between the two. Unfortunately this model was eventually dropped from comparison to this research due to time constraints.

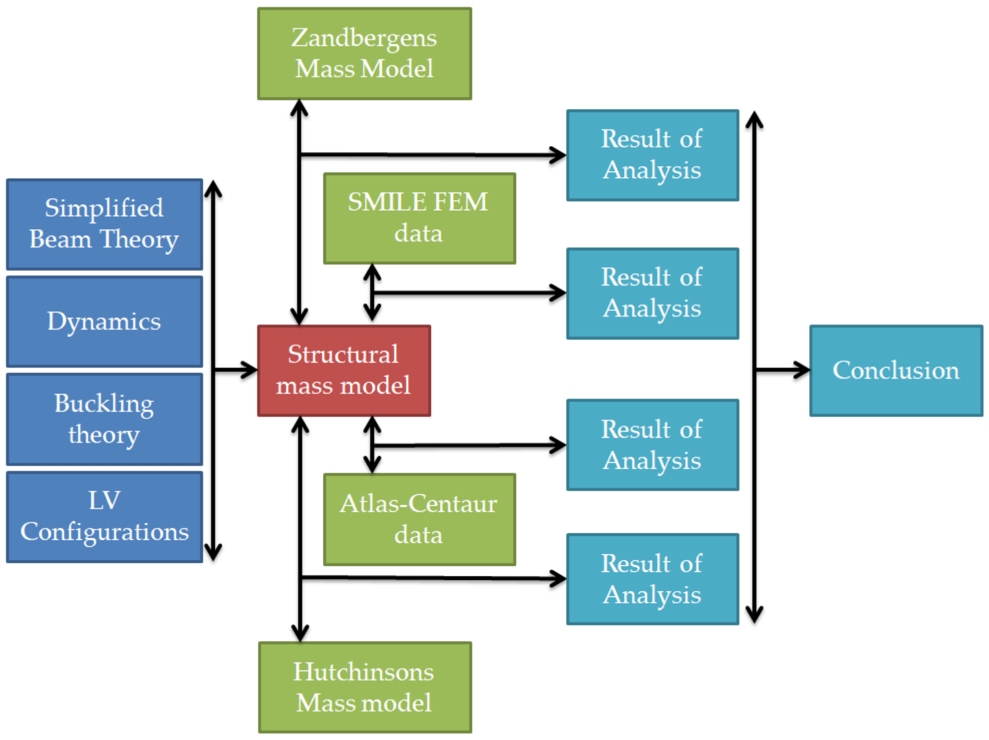
VI. Scope and Limitations

The report divides the project into components modeled and modeling methods implemented, to describe the scope of the project. To give a clear overview over what is and isn't modeled, lists are presented which indicate components or methods taken into consideration.

Components The scope of the project shall entail a structural mass model of a rocket stage or launch vehicle. This means modeling of the following structural components:

- Propellant tanks
- Pressurizer tanks
- Thrust frames
- Payload frame
- Fairing
- Inter-stages
- Rocket stage hull

Figure 1.2: *Research framework*



The following subsystems were incorporated to model structure mass but were not modeled in itself:

- Engines(s) [Mass, length, diameter]
- Avionics
- Payload

Modeling The following loads and yield criteria are taken into account by the model.

- Tensile yield
- Buckling yield
- Thrust
- Drag
- Inertial effects
- Lateral Forces and Moments
- Ullage gas pressure
- Hydro-static pressure

VII. Plan of Action

This section describes the process on which the model was build. The process of building the model was to be build using iterations. This means the model started with a relatively simple version and iterated towards a version in which it was to be a more refined with extra features. The model versions and their features are listed in table 1.3.

version 1: The first version took into account vertical loads and internal tank pressure. This means other than internal pressure no lateral loads were considered at this point. The idea behind this is that internal pressure and buckling criteria due to axial compression are the driving forces which determine the thickness of the hull. At this point only the most heavy subsystems(objects) are considered for modeling and due to simplicity the model limits itself at this point to a single stage rocket with a liquid engine. This is the version which was completed at the mid-term review point and thus on which the results in the mid-term report are based.

version 2: This version took into account lateral thrust created by thrust vectoring and thus took into account bending related stresses. This means the moment along the z axis of the LV was also modeled at this point. Comparing the results of the second and first model version showed the significance of bending loads and its effect on the structure mass. This version also incorporated support for multiple stages to make it easier to model a full launch vehicle. The addition of pressurization subsystems and thrust frames was a step towards refining the model with the goal of increasing its accuracy.

version 3: With the addition of multiple load-cases the project tried to fit more within the proposed concepts of the SMILE project. In addition was support for composite materials added. This created a possibility to test/compare the model with FEM numbers from the liquid concept and to test a whole flight trajectory instead of a single load-case. A proposed method of modeling heat transfer can be read in the Propellant Tank Mass Modeling chapter under the relevant section. Heat transfer under these circumstances is very complicated however and it has to be investigated further if the desired accuracy can be obtained by making certain assumptions to make the problem more simple. This was necessary to keep the scope of this project within its bounds to be able to be completed within the set time.

version 4: The fourth and final version tried to implement more refinement by adding object which mass can be approximated by existing empirical relations. The model can; other than adding these masses to the already calculated mass, give these objects a location which affect loads on structural parts. This effect can be small but this will be determined by analysis of the results. Another important addition will be bundling configurations. This allows the model to strap together multiple engines or pressure tanks and measure its effect compared to a different number of larger or smaller objects. Additionally the version introduced hydro-static loading in the axial direction. This addition was a refinement and will test its significance on the structure mass.

Table 1.3: *Summarized features of different model version.*

VERSION	Loads	Objects	Configurations
1	Vertical thrust Internal pressure Vertical acceleration Vertical drag	Propellant tanks Engines Fairing Interstages	Liquid Single Stage
2	Lateral thrust Moment	Pressure Tanks Pressurization group Thrust Frames	Multiple Stages
3		Payload-frame	Multiple-loadcases
4	Hydrostatic	Avionics	Bundling

VIII. Review of Literature

A short review of the literature used and mentioned in the project and report.

i. Mass Models

Determination of engine mass and avionics is directly used in the model. The formula's used to determine engine mass are taken from Zandbergen. For LOX, RP-1 engines the following equation was used:

$$M_e = 1.104 \cdot 10^{-3} F_T + 27.702 \tag{1.2}$$

And for H2, LOX engines:

$$M_e = 0.00514 \cdot F_T^{0.92068} \tag{1.3}$$

The mass modeling for avionics also comes from Zandbergen, this is defined as vehicle equipment bay (VEB).

$$M_{VEB} = 0.345 M_{dry}^{0.703} \tag{1.4}$$

Other equations by Zandbergen used for verification in this project are as follows. for dry stage mass two equations are used, one for semi-cryogenic and storable rockets stages.

$$M_{dry} = 0.0668 M_p + 1499 \tag{1.5}$$

And one for fully cryogenic stages:

$$M_{dry} = 0.0893 M_p + 1628 \tag{1.6}$$

Zandbergen also describes equations for construction mass. This is defined as a stage's dry mass minus the engines' mass. These equations are also two fold. One equation is for cryogenic and storable propellant combinations:

$$M_{const} = 0.022 M_p + 2090 \tag{1.7}$$

And the other equation is for fully cryogenic stages:

$$M_{const} = 0.0461 M_p + 1870 \tag{1.8}$$

ii. Atlas-Centaur

"Flight-Performance of Atlas-Centaur AC-13, AC-14, AC-15"[11] is a report which describes flight performance of early Atlas-Centaur versions in detail. Since this report is referenced a lot in this report it is briefly discussed here.

The launch vehicles discussed in the report use very traditional design as far as that applies in the launch vehicle industry. This means traditional construction material such as steel instead of more modern approach to construction materials such as CFRP (Carbon fiber reinforced plastic).

[11] Lewis' staff. *Flight performance of Atlas-Centaur AC-13, AC-14, AC-15 in support of the surveyor lunar landing program.* Tech. rep. NASA, Lewis Research Center, 1974.

The vehicles discussed use load carrying tanks which use internal tank pressure to counteract the structures nature to buckle which leads to extreme thin structures. This is one of the reason why this design is interesting for the model. Since it is likely that failure mode of certain systems changes with change in load-case.

Chapter 2

Launch Vehicle Mass Modeling

The body of this report starts with a description of LV mass modeling. Several mass models for launch vehicle design are already mentioned in the previous chapter under literature review and the methodology of these methods will not be discussed here. What will be discussed here is breaking down the problem towards modeling the mass of the structural subsystems of a launch vehicle by detailing mass modeling of the entire launch vehicle.

In order to do this, it is important to evaluate of what kinds of systems and subsystems a launch vehicle consists. Once all systems and subsystems are described an approximation can be made of each of its masses. These (sub)system masses can be added together to calculate the total dry mass of the LV. This breakdown of masses of (sub)systems is called a mass breakdown structure.

The first breakdown before a detailed mass breakdown structure is constructed is dividing the launch vehicle up into stages (obviously, some launch vehicle might consist of only one stage and this then thus does not apply). A rocket stage is defined as follows: Every subsystem of a rocket stage including the inter-stage or fairing above the stage (but not the inter-stage below).

This means that the payload was considered separate of the stage and is its own entity. Figure 2.1 shows this by displaying an example of a mass breakdown of a LV. The payload separate from the last stage in which it is encapsulated. It is however displayed on the same "level" as parts which are subsystems of a stage. This is done to indicate that the payload is not broken down further. It has a mass, a center of mass location and a specified geometry. Further breakdown of the payload was not deemed necessary for structural mass modeling.

For stages which are not the final stage, the stage includes the inter-stage which connects the the stage to the stage above it. Figure 2.1 shows a more detailed breakdown of the last stage. The fairing and payload frame subsystem are colored red since they only occur for the last stage. If a stage is not the final stage then the fairing and payload frame subsystems are replaced by the inter-stage subsystem.

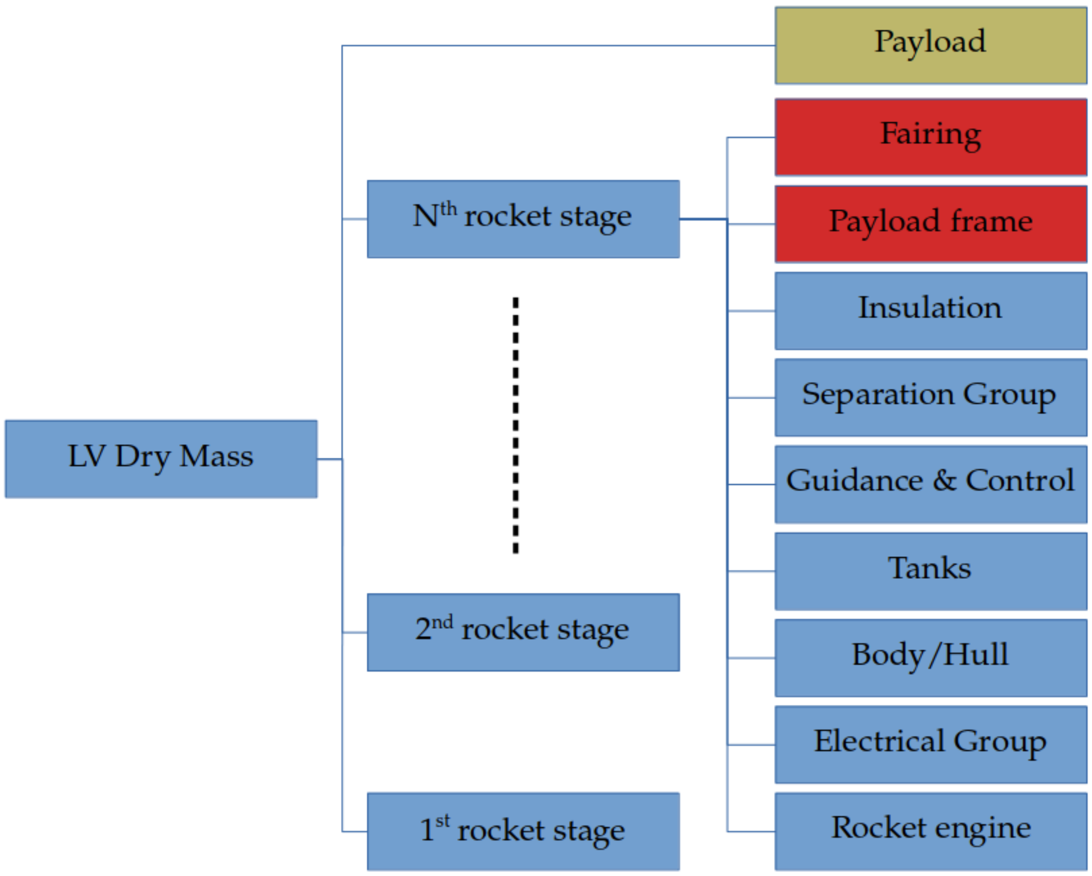


Figure 2.1: *Example of top level mass breakdown structure.*

I. Mass Breakdown Structure

Breaking down the mass of a rocket stage in separate subsystems makes it easier to access the work at hand. Figure 2.1 shows and example of a top level mass breakdown used in reference[1] for the Atlas-Centaur launch system. What (sub)system belongs to which stage can vary from project to project. The division used in this project is described in the previous chapter.

To introduce clarification in the report the mass breakdown which is applied to this project is described. This starts, just like in figure 2.1 with dividing the launch vehicle into separate stages. From here the dry rocket stage mass is divided into five sections; body, engine, tank, electronic group and pressure group. This can be seen in figure 2.2 which display the mass breakdown structure used in this project. There can be multiple tanks in a rocket stage but for simplicity and clarity only one is described in the figure.

Figure 2.2 distinguishes (sub)systems by color to clarify their role in the model. The red boxes represent the main system groups of the rocket stage, these boxes are also slightly larger compared to the other. The green boxes represent sub-systems which are modeled by calculating their mass without interference of empirical relations. These green sub-systems embody the work done in this model and where it distinguishes itself from existing mass models. Purple boxes represent sub-systems which are not modeled through analysis of this research. This does not mean the mass of these sub-systems is not taken into account, but it means that mass of these sub-systems depends on existing empirical relations and/or user input.

The body group is divided into six parts which details are described in figure 2.2. One of these sub-systems is not calculated trough structural analysis. This is the bulkhead sub-system. The definition of the bulkhead sub-system in this report are parts used to connect two or more structural parts or other sub-systems to each other. Between tanks in rocket stages this is often a thickened ring to attach the hull to the tank by using for example; nuts and bolts,welding,etc. It should be noted that the caps of the tanks do not belong to the bulkhead group. This is done on purpose since its required skin thickness and therefore mass can be calculated with structural analysis. This was the reason to omit this part from the bulkhead group, since it keeps a clear distinction between the green and purple sub-systems.

The engine system group contains all sub-systems contributing to the main propulsive force of the rocket. This systems and/or its subsystems mass' is empirically determined as can be seen from figure 2.2. These relations are described further into this chapter.

The Tanks as a system are divided into three green systems and three purple, empirical systems. Since propellant tanks are such a massive part of the overall structure of a LV they got their own chapter. The next chapter shall thus describe their mass calculations in more detail.

The last two systems; the electronics and pressure group were large buildup of parts which require empirical relations in order to calculate their mass. The exception of this are the pressure tanks which volumes and mass are calculated based on user input. This method is described in the tank modeling chapter since pressure tank modeling bears great resemblance to propellant tank modeling.

[1]Lewis' staff, see n. 11.

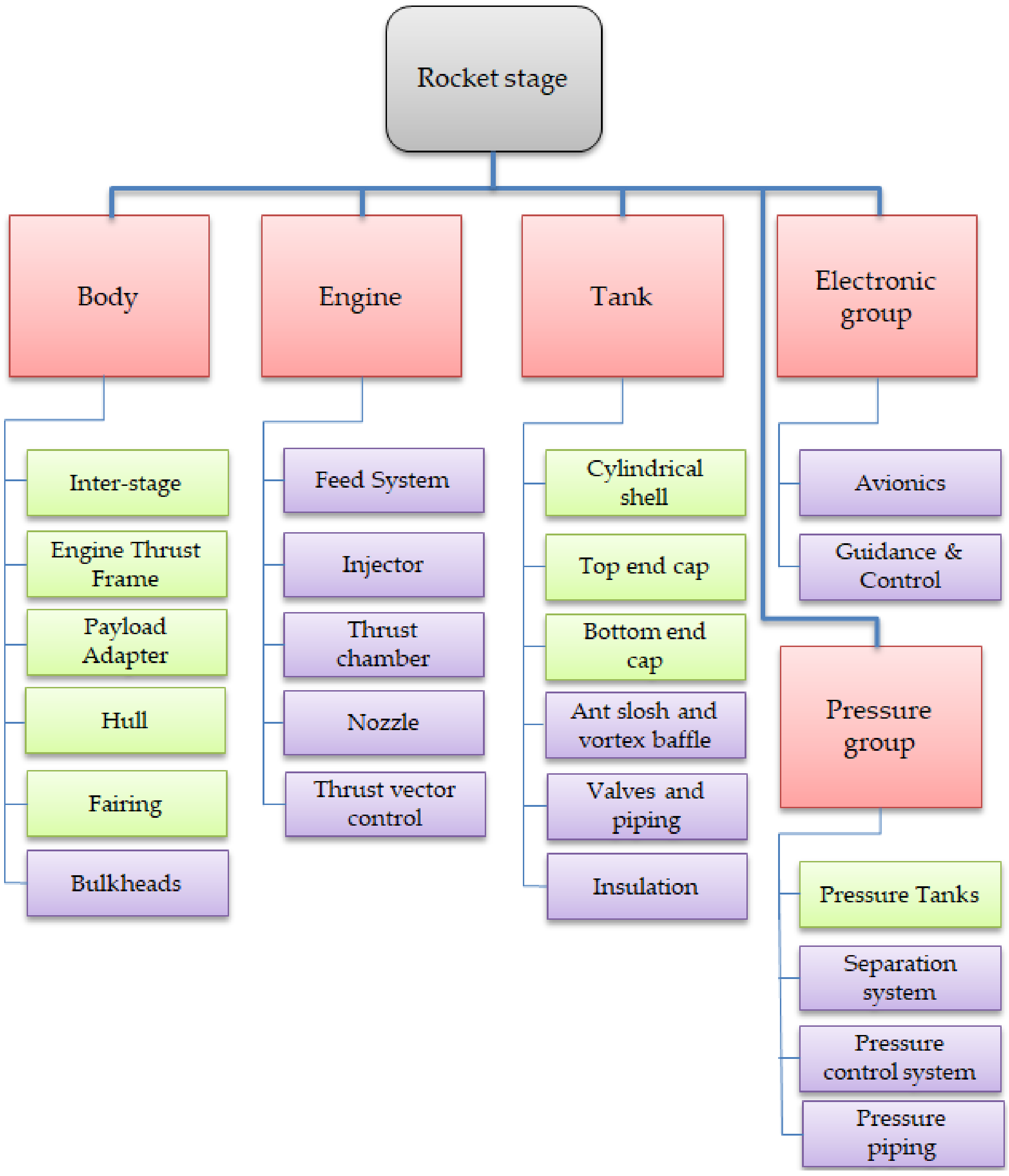


Figure 2.2: *Mass breakdown structure of model.*

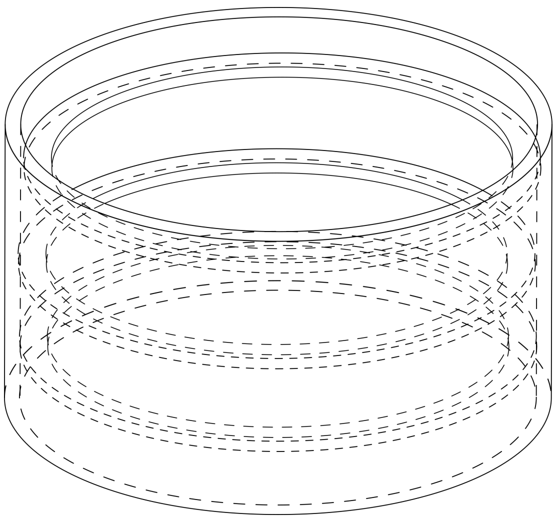


Figure 2.3: *Example of a hull object or inter-stage object in the model. One layer skin with three rings.*

II. Inter-stage and Hull

First subsystem described which mass is solely calculated by user input is the hull. The reason hull objects and inter-stage objects are combined into one section is that they are modeled in the same way. Hull objects are a structural manifold which can connect two or more tanks to each other. Hull and inter-stage objects are modeled as cylinder with a certain length L and diameter d. The cylinder can have stiffeners and rings to increase its strength as well as multiple layers of skin to facilitate sandwich panels or layers of orthotropic material. Figure 2.3 shows an example of how an inter-stage or hull object was modeled. The example is a cylinder with a single layered skin and three rings, this is obviously a fairly basic example. The body can for example also be conical when connected stages have different diameters.

To calculate mass of these objects one has to know the thickness of each layer, the length and diameter of the object and the geometry of possible rings and stiffeners. the volume of each ring/stiffener/layer has to also be multiplied by the density of the parts material. This calculation is described by equation 2.1 where N is the number of skin layers, s is the number of stiffeners,S_{st} is stiffener area, h_r is the height of the rings and u is the number on rings in the cylinder. The equation indicates it calculates mass of an inter-stage (M_{is}) but was also used for mass of hull objects.

$$M_{is} = 2\pi d L \sum_{j=1}^{N} t_j \rho_j + sL\rho_{st}S_{st} + \frac{\pi \rho_r}{4}(d^2 - (d - h_r)^2)t_r u \tag{2.1}$$

In order to calculate mass; thickness of each layer, ring geometry and stiffener geometry have to be known. The model uses a set input for ring and stiffener geometry after which required skin thickness in calculated. This method is also applied when a sandwich wall is desired. A set sandwich core thickness is defined after which the required thickness of the skin is calculated.

Since inter-stage and hull objects are mainly stressed in compression buckling analysis was used to determine their thickness. This buckling analysis had an axial force as input as well as

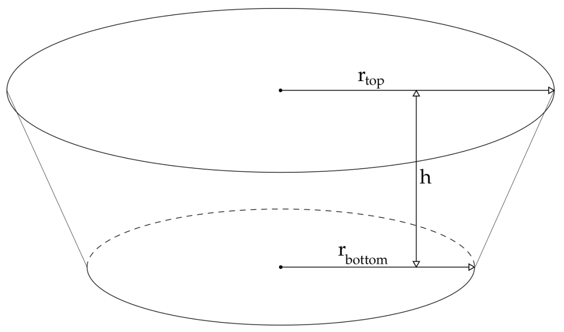


Figure 2.4: *3D sketch of a conical frustum.*

a moment which was applied to the object. How these force and moment were calculated can be read in the chapter about force and moments.

III. Thrust-frame and Payload adapter

Thrust and payload-frames are modeled by assuming their shape as a conical frustum. This means they have a top radius, bottom radius and height to constrict their geometry. An example of this shape can be seen in figure 2.4. To avoid confusion, the top radius r_{top} is defined to always be the larger radius and the bottom radius r_{bottom} to always be the smaller one. From now on thrust-frame is used to indicate this conical frustum object, however the same things apply to the payload-frame. The only difference is that the payload frame is generally "upside down" in a normal launcher configuration in the coordinates of the LV with respect to the thrust-frame(s).

The method used to calculate the mass of the thrust-frame is similar to that of the inter-stage object described earlier. This means surface is multiplied by its thickness. Surface for a conical frustum is defined by the following equations.[2] First the slanted length of the thrust-frame s is calculated :

$$s = \sqrt{(r_{top} - r_{bottom})^2 + h^2} \tag{2.2}$$

Now the surface can be calculated with the following formula:

$$S_{tf} = \pi(r_{top} + r_{bottom})s \tag{2.3}$$

Once the surface is calculated thrust-frame mass M_{tf}, was calculated as follows:

$$M_{tf} = S_{tf} \sum_{j}^{N} \rho_{j=1} t_j \tag{2.4}$$

Where N is the number of layers in the wall and with t and ρ being thickness and density respectively.

In order to calculate mass one thus needs to know the top and bottom radius and the height of the thrust-frame. To calculate the geometry of the thrust-frame, its location in the rocket stage has to be defined. This location was defined to be between the cylindrical engine and the elliptical propellant tank above the engine.

[2]Eric W. Weisstein. *Conical Frustum*. 2017. URL: `mathworld.wolfram.com/ConicalFrustum.html`.

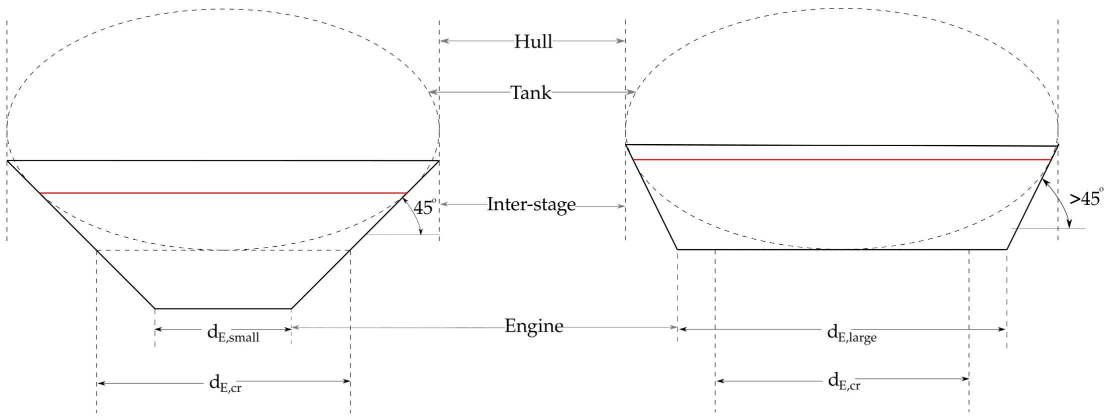


Figure 2.5: *Thrust frame configurations in 2D*

Since thrust-frames are loaded in compression and will thus generally fail due to buckling the decision was made to **not** let the angle of the thrust frame be lower than 45^o, this should ensure proper buckling analysis to take place. Another decision was made is for choice of configuration of the thrust-frame. Two configurations can be chosen from in the model. The first configuration is where the thrust-frame is attached to the hull of the rocket which can be seen in both the left and right side in figure 2.5. The second configuration is where the thrust-frame is connected to the propellant tank at the point where the angle of the thrust-frame and elliptical tank are equal. This second configuration is indicated by a red line in figure 2.5.

Figure 2.5 shows thrust-frame geometries for different engine sizes. The left side shows a configuration with a small engine (or bundle of engines) diameter. This means the engine diameter is smaller than the critical diameter d_{cr}. The right side shows a configuration with a larger engine diameter ($d_E > d_{cr}$). The angle of the thrust-frame becomes larger when the critical engine diameter is exceeded. All in all four main configurations can thus be distinguished:

1. small engine, tank attached
2. large engine, tank attached
3. small engine, hull attached
4. large engine, hull attached

To distinguish between small and large engine configurations the critical diameter has to be calculated. To do this the point on the ellipse where its angle is 45 degrees has to be defined. This translates to a slope value of 1 or -1. The ellipse shape of the tank is defined as follows:

$$\frac{x^2}{a^2} + \frac{z^2}{b^2} = 1 \tag{2.5}$$

or:

$$z = \pm\frac{b}{a}\sqrt{a^2 - x^2} \tag{2.6}$$

To calculate the coordinates for the slope, equation 2.5 has to be differentiated with respect to x to obtain z'. This was done by implicit differentiation.

$$\frac{d}{dx}\left(\frac{x^2}{a^2}\right) + \frac{d}{dx}\left(\frac{z^2}{b^2}\right) = \frac{d}{dx}(1) = 0 \tag{2.7}$$

$$\frac{d}{dx}\left(\frac{x^2}{a^2}\right) = \frac{2x}{a^2} \tag{2.8}$$

$$\frac{d}{dx}\left(\frac{z^2}{b^2}\right) = \frac{zz'}{b^2} \tag{2.9}$$

Hence:

$$z' = -\frac{xb^2}{za^2} \tag{2.10}$$

Since the value of z' is known($z' = 1 \;||\; z' = -1$) it can be filled in, in equation 2.10. This creates the following value for z_{cr}:

$$z_{cr} = -\frac{x_{cr}b^2}{a^2} \tag{2.11}$$

This can be filled in into equation 2.6.

$$-\frac{x_{cr}b^2}{a^2} = \frac{b}{a}\sqrt{a^2 - x_{cr}^2} \tag{2.12}$$

rearranging and simplification results in the following equation.

$$x_{cr} = \pm\frac{a^2}{\sqrt{1 + \frac{b^2}{a^2}}} \tag{2.13}$$

z_{cr} can than also be found by putting the value of x_{cr} into equation 2.6. With x_{cr} one can determine if the engine diameter is considered large ($d_E > 2x_{cr}$) or small ($d_E < 2x_{cr}$).

Configuration 1 First considering a small engine where the thrust-frame is attached to the tank. The angle for this configuration is set at 45 degrees and bottom radius is known as half the engine diameter. This leaves top radius which is equal to x_{cr} which can be calculated as can be seen from the equation above. The only variable left which defines the thrust-frame is its height. The height can easily be calculated from the known variables since an isosceles right triangle can be constructed between the slant length and height, hence:

$$h = r_{top} - r_{bottom} \tag{2.14}$$

Configuration 2 When considering large engine attached to the tank the thrust-frame has a set height. This can also be observed from figure 2.5 where the right side configuration has a large engine. The bottom of the thrust-frame coincides with the bottom of the tank, hence the height becomes:

$$h = b + z_{top} \quad when\ z_{cr} < 0 \tag{2.15}$$

This thus means that z_{top}, x_{top} and the thrust-frame angle need to be determined. Please note that unlike for small engine configuration, z_{top} and x_{top} do not coincide with z_{cr} and x_{cr} respectively. The slope line which runs tangent to the ellipse is given by equations 2.16 and

2.17, where c is a constant value. These two equation together with equations 2.10 and 2.6 form four independent equations which solve for four unknown variables; z', x_{top}, z_{top}, c. The model solves this problem numerically.

$$z_{top} = z' x_{top} + c \tag{2.16}$$

$$z_E = z' x_E + c \tag{2.17}$$

Configuration 3 & 4 Hull attached thrust-frames are in geometry similar to tank attached thrust-frames. Hull attached thrust-frames are slightly larger and have a different x_{top} and z_{top} coordinates compared to their tank attached counterparts. Since all variables from the configurations 1 and 2 can be determined, and x_{top} is equal to the stage radius by definition; ,z_{top} for hull attached thrust-frames can be determined. Equation 2.18 shows how z_{top} was calculated, the equation is derived from the line tangent to the elliptical tank shape and is based on the fact that this line is linear and hence has a constant slope.

$$z_{top} = z'(x_{top} - x_E) + z_E \tag{2.18}$$

IV. Fairing

A rocket fairing's purpose is to protect the payload from outside forces which mainly consist of aerodynamic forces during ascent. The structure thus needs to withstand pressure difference over the skin which can be significant due to the LV's high velocities. The magnitude of this aerodynamic drag is dependent on air density ρ_{air}, the velocity v, a reference surface S_{ref} and a drag coefficient C_D as can be seen in equation 2.19.[3] This drag coefficient is mainly dependent on shape of the object and is often determined experimentally, this thus makes is difficult to approximate within this model without access to experimental data. To keep the work in this research to acceptable proportions it was decided to leave aerodynamic analysis out of the model, this means that the user is expected to deliver a drag value or a load-factor from which drag can be calculated. This ensures that no complicated aerodynamic analysis has to be conducted to use the model.

$$D = 0.5 C_D \rho_{air} v^2 S_{ref} \tag{2.19}$$

Another problem that was encountered in determining structural integrity of a fairing is the pressure distribution over the fairing during flight. The pressure distribution is the actual force acting on the structure but can vary greatly from tip to base. This pressure distribution can be determined through computational fluid dynamics (CFD) but this process is computational intensive. To analyze the required thickness of the fairing in this model, simplifications have to be made.

Since aerodynamic drag is determined to be the main load on the fairing, the material experiences compression. The fairing is considered to be a thin walled structure and thus buckling was considered to be its main method of yielding. This was the first assumption made for simplification, to analyze the wall thickness of the fairing. And it was thus decided that the fairing will undergo a buckling analysis.

[3]John David Anderson Jr. *Fundamentals of aerodynamics*. Tata McGraw-Hill Education, 2010.

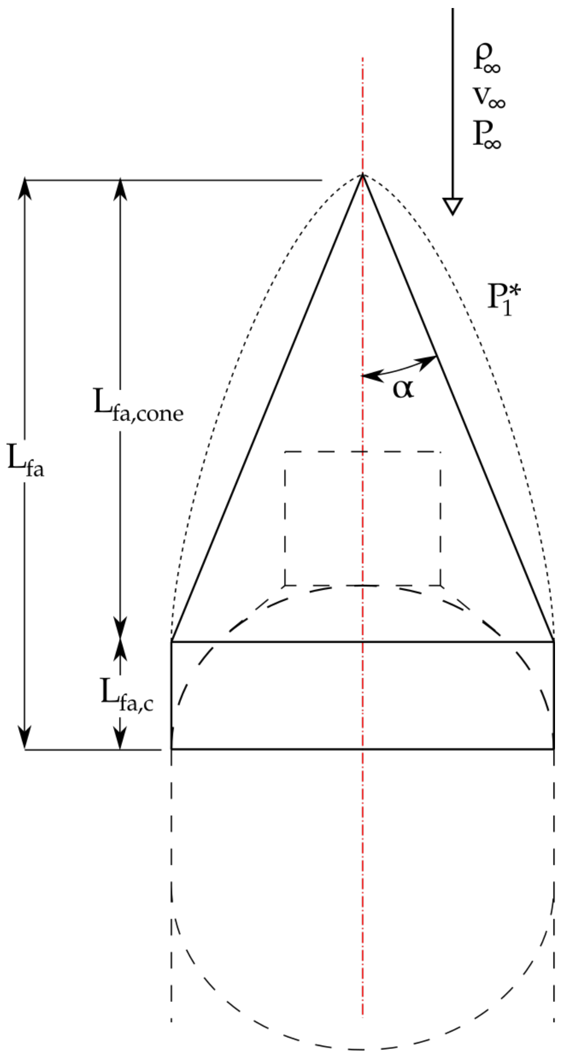


Figure 2.6: *Sketch representing the modeling of the fairing*

To analyze the buckling behavior, an assumption had to be made about the load case. The assumption was made that the fairing tip experiences a uniform pressure difference over its entire surface. This external pressure is calculated from a known drag value.

The third and final assumption to determine the required thickness of the fairing is a simplification of geometry. The assumption is made that a conical shape has similar buckling behavior under constant external pressure compared to other nose-cone shapes. This assumption permits the use of buckling analysis used for cones found in literature to be used for all fairing shapes, this assumption is displayed in figure 2.6. Where the dotted line represents the fairings real outline and the solid line represent the equivalent cone on which buckling analysis is carried out.

The model supports two basic fairing shapes a blunted cone and the so called "von Karman" shape. The first shape is a simple cone which surface is approximated by equation 2.20 where r is the radius at the base and s is the cones slant height defined by equation 2.21 where parameter h is the cone height measured from its base.[4]

$$S_{cone} = \pi r s \tag{2.20}$$

$$s = \sqrt{r^2 + h^2} \tag{2.21}$$

The second shape, the "von Karman" shape is defined by equation 2.22 and 2.23.[5] h and r is the nose-cone height and base radius respectively. z is the height measured from the nosecone tip and y is the distance from the fairing axis of revolution.

$$y = \frac{r\sqrt{\theta - \frac{sin(2\theta)}{2}}}{\sqrt{\pi}} \tag{2.22}$$

$$\theta = arccos\left(1 - \frac{2z}{h}\right) \tag{2.23}$$

The surface of the "von Karman" shape is calculated numerically by taking a small distance, dz of the line plotted by equation 2.22 and thus obtaining dy. The distance ds is then calculated with Pythagoras theorem seen in equation 2.24. This small distance is then integrated by surface of revolution to create a small surface increment seen in equation 2.25. Eventually this surface increment; Ds is used to calculated to entire surface of the "von Karman" nosecone.[6]

$$ds = \sqrt{dz^2 + dy^2} \tag{2.24}$$

$$dS_z = 2\pi y ds \tag{2.25}$$

$$S = \int dS \tag{2.26}$$

[4]Eric W. Weisstein. *Cone*. 2017. URL: `mathworld.wolfram.com/Cone.html`.

[5]Gary A Crowell Sr. “The descriptive geometry of nose cones”. In: *URL: http://www. myweb. cableone. net/cjcrowell/NCEQN2. doc* (1996).

[6]Eric W. Weisstein. *Surface of Revolution*. 2017. URL: `mathworld.wolfram.com/SurfaceofRevolution.html`.

Chapter 3

Propellant Tank Mass Modeling

I. Introduction

This chapter discusses the propellant tank mass model. It was decided to assess the propellant tank separately from the rocket stage since it is such a large integral portion of the launch vehicle and thus greatly contributes to its mass. This will give an opportunity to analyze this subsystem in more detail which will hopefully result in an overall better model. The goal of the tank model is to implement several design options which can help in the conceptual design phase of a launch system. These design options are discussed throughout the chapter and include tank rigidity and propellant temperature.

To analyze the mass of a propellant tank it is divided into several components. The main component to determine mass of the tank is the structure which function is to keep the propellant contained while maintaining structural integrity. Other components might be insulation, anti vortex/slosh baffles, valves, etc. Table 3.1 shows a mass breakdown of this subsystem. The final version of the model only takes into account the top,cylinder and bottom shell of the tank as these span the entire surface of the tank but if desired the user can add an insulation with a given density per square meter. The stresses occurring in the structure are discussed and evaluated and the lay-out of the model is shown and explained.

Table 3.1: *Tank group mass breakdown*

TANKS	Mass breakdown	remarks
Top Shell		
Cylinder Shell		
Bottom Shell		
Anti-vortex Baffle(s)		optional
Anti-slosh Baffle(s)		optional
Pipe(s) to Engine		
Pressure Relieve Valve(s)		
Insulation		optional
Liner		optional
Drain/Fill Valve(s)		

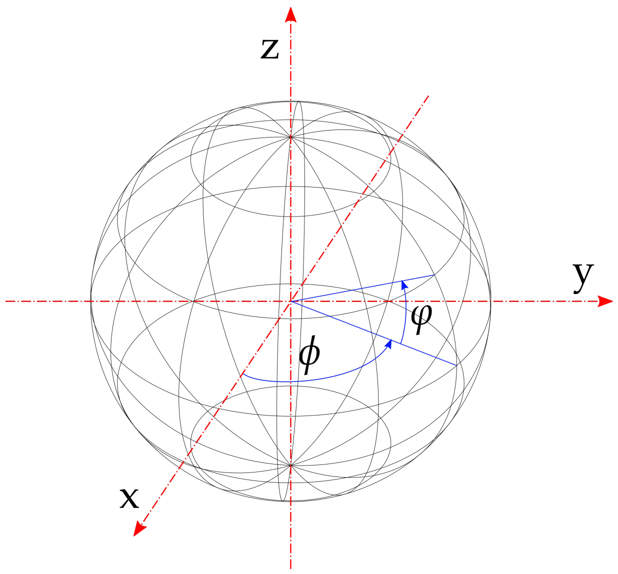


Figure 3.1: *Illustration of the reference frame used in this report.*

II. Structure

The structure of a propellant tank has as main purpose to contain the fuel and/or oxidizer in liquid of gaseous form. This means that the structure mainly consists of the tank wall which at least needs to be strong enough to withstand the internal pressure of the tank. To determine the mass of the tank structure an assumption can be made that the tank is a thin shell structure. This means mass of this shell can be determined taking the tank surface, thickness and material density as seen in equation 3.1. Initially the tank is assumed to be cylindrical with spherical end caps.

$$Tank\ shell\ mass : M_{shell} = \iint\limits_S t_t \rho_t \, dS \tag{3.1}$$

Stresses in the tank can be determined from thin shell theory[1] and can be seen in equation 3.2 and 3.3. The stresses here are defined for a spherical tank/tank-section and a cylindrical section. To get an overview of the reference frame and variables used ,figure 3.1 is plotted below.

$$Sphere : \sigma_\varphi = \sigma_\phi = \frac{P_t r_t}{2t_t} \tag{3.2}$$

$$Cylinder : \sigma_\phi = \frac{P_t r_t}{t_t} \tag{3.3}$$

Rearranging these equations gives a formula for required tank thickness[2] at a given internal pressure, material yield stress and tank radius:

$$Sphere : t_t = \frac{P_t r_t}{2\sigma_{yield}} \tag{3.4}$$

[1]Warren Clarence Young and Richard Gordon Budynas. *Roark's formulas for stress and strain*. Vol. 7. McGraw-Hill New York, 2002.

[2]Required tank thickness without incorporating a safety factor

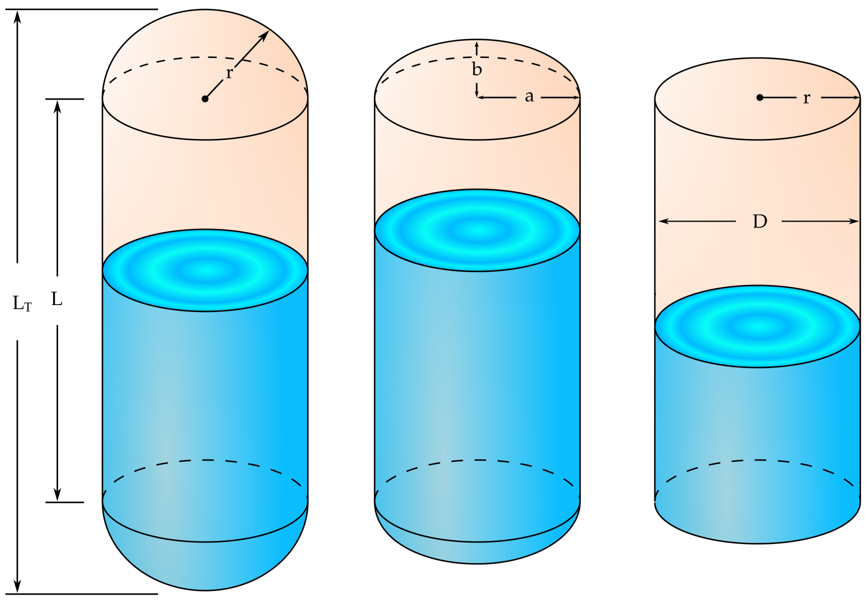


Figure 3.2: *Different tank shapes, from right to left: Cylinder, Cylinder with elliptical caps, Cylinder with spherical caps.*

$$Cylinder : t_t = \frac{P_t r_t}{\sigma_{yield}} \tag{3.5}$$

To determine the mass of the structure the tank surface also needs to be calculated, the surface equations for a cylinder with open ends and a sphere are shown in equation 3.6 and 3.7.

$$Sphere : S_s = 4\pi r_t^2 \tag{3.6}$$

$$Cylinder : S_c = 2\pi r_t L \tag{3.7}$$

For a cylindrical tank with spherical end caps, equations 3.6 and 3.7 need to be combined:

$$Tank\ Surface : S_t = 4\pi r_t^2 + 2\pi r_t L \tag{3.8}$$

Equations 3.4, 3.5 and 3.8 show that tank mass depends on tank radius and length[3]; r_t and l_c respectively. Tank radius and length might however not be the best input parameter at a conceptual level. Better parameters would be tank diameter, and propellant mass since they relate better to aerodynamic properties and rocket performance. Propellant mass can be linked to propellant volume, which is assumed to be equal to tank volume for this discussion about structures; equation 3.9.

$$V_t \approx V_p = M_p\ \rho_p \tag{3.9}$$

Equation 3.9 links the propellant mass to the tank volume and is defined in equation 3.10.

$$Tank\ Volume : V_t = \frac{\pi}{6} d_t^3 + \frac{\pi}{2} d_t^2 L \tag{3.10}$$

[3]Length of the cylindrical section of the tank

Thus from Diameter, Propellant mass and density the tank length can be obtained. with which the tank surface can be calculated through equation 3.8. This leaves propellant density as variable that needs to be determined in order to determine dry tank mass.

Traditionally propellant density could be taken as a constant in rocket design since propellant was often stored at room temperature or in case of cryogenic propellant close to the boiling point. More modern configurations can have a design where the propellant is stored close to its freezing temperature to increase the propellant density. The goal of this design property is to lower the empty tank weight and hence increase performance. This means that propellant temperature is also a valuable input parameter to determine tank weight. Internal tank pressure also affects propellant density much like temperature but this is of an magnitude smaller than temperature. As example one can look at liquid oxygen, at a temperature of 65 $[K]$ the change in density for an increment of temperature is 4.5 $[kgm^{-3}k^{-1}]$ and 0.126 $[kgm^{-3}bar^{-1}]$ for a unit of pressure[4]. Furthermore is the range much larger for temperature storage than internal pressure. Tank pressure for the space-shuttle external oxygen tank was only 2.5 $[bar]$ as an example, while the range of storing LOX is 35 degrees Kelvin from freezing to boiling point. An assumption can thus be made that for liquid propellant the density is only a function of storage temperature.

$$\rho_p \approx \rho_p(T) \tag{3.11}$$

i. Elliptical End Caps

LV tank design often incorporates half an ellipsoid as end cap in favor of a spherical geometry. This is done to lower the height profile of the LV by partially "filling" up the empty space in its cylindrical shape. Stress in ellipsoids tanks have a separate subsection because of slightly more complex math involved to calculate the necessary thickness. The following equations also come from thin shell theory and consist of stress in the material due to internal pressure.

$$Ellipsoid : \sigma_\varphi = \frac{p_T R_1}{2t} \tag{3.12}$$

$$Ellipsoid : \sigma_\phi = \frac{p_T R_1}{t}\left(1 - \frac{R_1}{2R_2}\right) \tag{3.13}$$

With:

$$R_1 = \frac{\sqrt{a^4z^2 + b^4x^2}}{b^2} \tag{3.14}$$

$$R_2 = \frac{(a^4z^2 + b^4x^2)^{3/2}}{a^4b^4} \tag{3.15}$$

Fortunately, it is easy to solve for thickness ,t in equation 3.12 and 3.13 since it only appears once. This results in equation 3.16 and 3.17 and thus there are two solutions to the required thickness. This can be solved by incorporating a yield criterion which takes into account both principle stresses. To do this the Von Mises yield criterion is chosen and can be seen in equation 3.18 for plane stress without shear stresses. Using this criterion the required thickness can be calculated by rearranging the equation and solving for thickness, this formula is shown in equation 3.19. The derivation for this formula can be found in the appendix.

$$Ellipsoid : t_t = \frac{p_T R_1}{2\sigma_\varphi} \tag{3.16}$$

[4]Data is taken from NIST database

$$Ellipsoid : t_t = \frac{p_T R_1}{\sigma_\phi}\left(1 - \frac{R_1}{2R_2}\right) \tag{3.17}$$

$$\sigma_v = \sqrt{\sigma_\varphi^2 - \sigma_\varphi\sigma_\phi + \sigma_\phi^2} \tag{3.18}$$

$$Ellipsoid : t_t = \frac{p_T}{\sigma_{yield}}\sqrt{A^2 - AB + B^2} \tag{3.19}$$

$$A = \frac{R_1}{2} \tag{3.20}$$

$$B = R_1\left(1 - \frac{R_1}{2R_2}\right) \tag{3.21}$$

ii. Input Parameters

Choice of input parameters for the model is important, in order to accurately determine the mass of the tank shell structure:

M_p : Propellant mass

σ_{yield} : Yield stress of the material

P_t : Internal tank pressure

ρ_p : Propellant density

iii. Ullage volume

Equation 3.9 sets the propellant tank's volume equal to the propellant volume. In reality this is not the case. Propellant tanks have an initial ullage volume to accommodate for:

- Expansion of the fluid due to change in temperature.
- Avoid rapid increase of tank pressure due to a pressurization fail (valve stuck in open position).[5]
- Pressurization of the liquid propellant.

Determining required ullage volume is difficult and little to no literature on the subject was found during the research. This has to do with the fact that this depends on available mass flow from pressure relieve valves, heat-transfer, motion of the propellant within the tank, interaction between the gas and the liquid and other factors..[6] Even if this is all solved this might not be a good method to determine required volume without experimental tests. For this reason the ullage volume is modelled by means of empirical data. This decision strays a bit from the fact that the project does not want to depend on empirical values, but seems to be a essential in order to achieve realistic values. Values obtained from reference[7] report the following values for ullage volume of the Centaur upper stage:

- liquid hydrogen - 1.4 % of total tank volume
- liquid oxygen - 2.2 % of total tank volume

[5]A Hedayat, KC Knight, and RH Champion Jr. "Transient Analysis of X-34 Pressurization System". In: (1998).

[6]Viatcheslav V Osipov et al. "Dynamical model of rocket propellant loading with liquid hydrogen". In: *Journal of Spacecraft and Rockets* 48.6 (2011), pp. 987–998.

[7]Lewis' staff, see n. 11.

iv. Load Cases

This section will discuss the load cases under which the structural part of the tank in itself operates. This means the stresses which are occurring by loads directly applied to the tank. As discussed in previous sections, the tank shell experiences stress from storing the propellant at a pressure difference ($Tankpressure > Environmentpressure$) compared to the environment in which the tank is placed. This induces a tensile stress on the whole body of the tank. This property has been used to decrease the weight of the tank for load cases that include compresive stress. The Atlas booster is an example of such a design philosphy where the tank structure is a so called "balloon tank". This means the tank uses internal pressure to retain its shape to keep it from buckling. This caused different minimum pressure requirements for different load-cases, i.e. differnt stages during operation.[8] Compression loads on a launch vehicle can induce buckling behavior during flight, but this will be treated in the structure modeling and full rocket stage modeling chapters which will include the tank section.

Self-Buckling: The scenario described above is an example of possible self-buckling. This is a situation in which a structure cannot retain the stresses of its own weight under compression. In the case of a propellant tank the tank has to be kept at a minimum internal pressure to avoid buckling. This is design choice made by the engineer. The design can incorporate a very thin balloon tank which has to be kept under pressure at all times or a rigged tank design which can support its own weight. Stress in the material caused by their own weight can be found in the appendix.

Hydro-static Pressure: Hydro-static pressure occurs when a column of liquid is experiencing acceleration due to gravitation or acceleration of the column. Fluid at the top of the column exerts a pressure on the fluid below which creates a pressure difference across the column. This pressure difference over a column also exist in liquid propellant tanks, the effect increases with a longer tank length and stresses caused by this effect are described in the appendix.

Thermal Shock: Thermal shock is caused by a high temperature gradient in a material which creates stress due to a delta in the thermal expansion of the material. This kind of stress is not evaluated by the model because of the difficulty to calculate wall temperatures in a rather straightforward manner. This is more discussed in the insulation section of this chapter.

III. Pressurization Tanks

Pressurization tanks supply pressure to the systems in the launch vehicle and are often present in a rocket stage in one form or another. In liquid and hybrid stages these tanks often regulate pressure of the propellant tanks, i.e.; oxidizer and fuel tanks. This means these tanks can often be quite substantial in dry mass since they often contain pressures up to hundreds of [bar].

Determining the mass of pressurization tanks is similar to propellant tanks in the sense that the tanks wall thickness can be determined through determination of the internal pressure, radius and material yield stress. material and internal pressure can be taken as design variables. This leaves the radius of the tank and this requires knowledge about the required tank volume. This is something that needs to be determined and hence this section discussing this part separately.

[8]Lewis' staff, see n. 11.

To determine the tank volume the ideal gas law is used, equation 3.22. In this equation P is pressure, V is volume, M is mass of the gas, R is the ideal gas constant, T is the temperature of the gas and $\mathscr{M}$ is the molar mass.

$$PV = \frac{MRT}{\mathscr{M}} \tag{3.22}$$

To determine the volume we need to set the pressure and temperature of the gas in its container. As an example an internal pressure and storage temperature can be taken.

- Internal pressure $= 200[bar]$
- Storage temperature $= 15[K]$

A gas often used to pressurize systems is Helium, this gas is ideal for pressurization because of its inert property and it remains gaseous at extremely low temperatures. The molar mass of Helium is:

- molar mass $= 4[g/mol]$

This analysis calculates the necessary volume and mass of the Helium by determining the space it needs to fill when the propellant tanks of the rocket are empty and at which pressure it needs to keep these tanks. Using this method the necessary Helium mass to fill the propellant tank can be determined. To continue the example; the necessary Helium for a fuel and oxidizer tank are determined.

$$M_{He,fueltank} = \frac{V_{fueltank} P_{fueltank} \mathscr{M}}{RT} \tag{3.23}$$

$$M_{He,oxtank} = \frac{V_{oxtank} P_{oxtank} \mathscr{M}}{RT} \tag{3.24}$$

Since the helium tank needs to retain a positive pressure delta compared to the systems it needs to pressure, the pressure in the helium tank need to be larger or equal to the tank with the second highest pressure (after the He tank of course). Let's assume:

$$P_{oxtank} > P_{fueltank} \tag{3.25}$$

then:

$$M_{min,presstank} = \frac{V_{presstank} P_{oxtank} \mathscr{M}}{RT} \tag{3.26}$$

finally the total helium mass can be determined:

$$M_{He,total} = M_{He,fueltank} + M_{He,oxtank} + M_{min,presstank} \tag{3.27}$$

After rearranging and simplifying this leads to:

$$V_{presstank} = \frac{V_{fueltank} P_{fueltank} + V_{oxtank} P_{oxtank}}{P_{presstank} - P_{oxtank}} \tag{3.28}$$

With the volume of the pressurization tank the radius and thus the tank wall thickness can be calculated. It is important to note that equation 3.28 describes tank volume for isothermal expansion of the helium gas. This meas that is it assumed the temperature of the helium stays equal to that of its initial storage temperature. This is not necessarily the case, lower pressurizer weight can be achieved by implementing a heat exchanger to heat the helium gas before pressurizing the fuel and oxidizer tanks. If this is the case equation 3.28 becomes equation 3.29, where T_{new} is the temperature of the helium after it is heated.

$$V_{presstank} = \frac{(V_{fueltank}P_{fueltank} + V_{oxtank}P_{oxtank})}{(P_{presstank} - P_{oxtank})} \frac{T_{presstank}}{T_{new}} \tag{3.29}$$

Another method of in which the temperature can be increased is if an isentropic expansion is assumed. In this case the new temperature can be calculated with equation

$$T_{new} = T_{initial} \left(\frac{P_{new}}{P_{initial}} \right)^{\dfrac{\gamma - 1}{\gamma}} \tag{3.30}$$

IV. Insulation

A Calculation of necessary insulation thickness was not incorporated in the model. A user of the program can however add an insulation layer to a wall and give a prefixed density. The reason insulation thickness in not calculated is described in this section.

Cryogenic storage of propellant requires insulation to keep the propellant cool to retain the benefits of the higher density the liquid phase of the propellant provides. Common used cryogenic propellants in LV design are for example liquid methane, LOX and LH. Cryogenic propellant was briefly discussed in the previous chapter about structures since its temperature affects its density and therefore the size of the structure. This section will try to approximate the mass of the insulation. It is thought insulation mass can be approximated quite accurately because it covers the entire surface of the tank which is a known parameter. The only other parameters which are necessary to determine the mass of the insulator is its density and thickness as can be seen in equation 3.31. Subscript i indicates insulation here, t is the thickness, ρ is the density and M is the mass. This is integrated over the tank surface S. Equation 3.31 assumes the insulation layer can be approximated as a thin shell similar to which was done to the structural component.

$$Mass\ of\ Tank\ Insulation : M_i = \iint\limits_S t_i \rho_i \, \mathrm{d}S \tag{3.31}$$

Density of the insulator is chosen as an input parameter in the model. The reason for this is to create an easy way to compare weight difference of various insulators in a conceptual design phase. Another input parameter is the insulator's thermal conductivity coefficient; k. The thermal conductivity coefficient in $[watt\ k^{-1}\ m^{-1}]$ determines how well (or how bad) the material transfers heat due to conduction. To have a proper analysis this value must also be known for the structural material used, since heat is also transferred through the tank shell.

With these input parameters the required thickness of the insulator has to be determined. This is not a trivial analysis since it requires additional input of which some are design requirements. This can be explained by the operational conditions of a propellant tank. In conventional set-ups the tank is often filled with propellant on the launch pad. This means that the operation starts with an empty tank which is filled until its maximum capacity is reached. The LV can however also experience ground hold which also affects the total heat transferred to the propellant. This in combination with aerodynamic heating during flight and forced convection of wind when grounded. A paper which discusses modeling the filling process of the space shuttle external tank by Osipov et al. describes this process in detail.[9] The

[9]Osipov et al., see n. 6.

discussion about heat transfer assumes a steady-state system onwards, and thus ignores initial temperature of the structure and insulation. The assumption is made that the materials will quickly change temperature due to the high temperature delta and their relative thin profile.

i. Tank Wall Heat Transfer

Heat transfer through the tank wall is a combination of conduction,convection and radiation. This process in shown in figure 3.3. As can be seen the propellant temperature is kept at a colder temperature than the outside air temperature. The linear curves in the shell and insulation show that this heat transfer is controlled by conduction. Equations 3.32 and 3.33 show the heat flux; q $[Wm^{-2}]$, through the respective material for an assumed one dimensional case.[10] The parameter k $[Wm^{-1}K^{-1}]$ in the equation stands for thermal conductance.

$$Shell\ Conduction : q = -k_{shell}\frac{dT}{dx} = -k_{shell}\frac{T_2 - T_1}{x_2 - x_1} \tag{3.32}$$

$$Insulation\ Conduction : q = -k_i\frac{dT}{dx} = -k_i\frac{T_3 - T_2}{x_3 - x_2} \tag{3.33}$$

$$x_2 - x_1 = t_s \tag{3.34}$$

$$x_3 - x_2 = t_i \tag{3.35}$$

Heat flux between the propellant and the tank shell is dictated by convection as well as conduction. This can also be seen in figure 3.3 where steeper temperature gradients are shown for conduction amplified by convection. Convection essentially includes the impact of fluid movement on heat transfer.

Heat flux for with convection can be seen in equation 3.36 and shows that heat transfer coefficient; h $[Wm^{-2}K^{-1}]$ is necessary to calculate heat flux. To calculate the heat transfer coefficient the Nusselt number (equation 3.37) can be approximated. The Nusselt number is the ratio between heat transfer with convection over the heat transfer with just conduction. To make this parameter dimensionless it is multiplied by a characteristic length which is the height of the wall which is in contact with the propellant.

$$q = h(T_p - T_1) \tag{3.36}$$

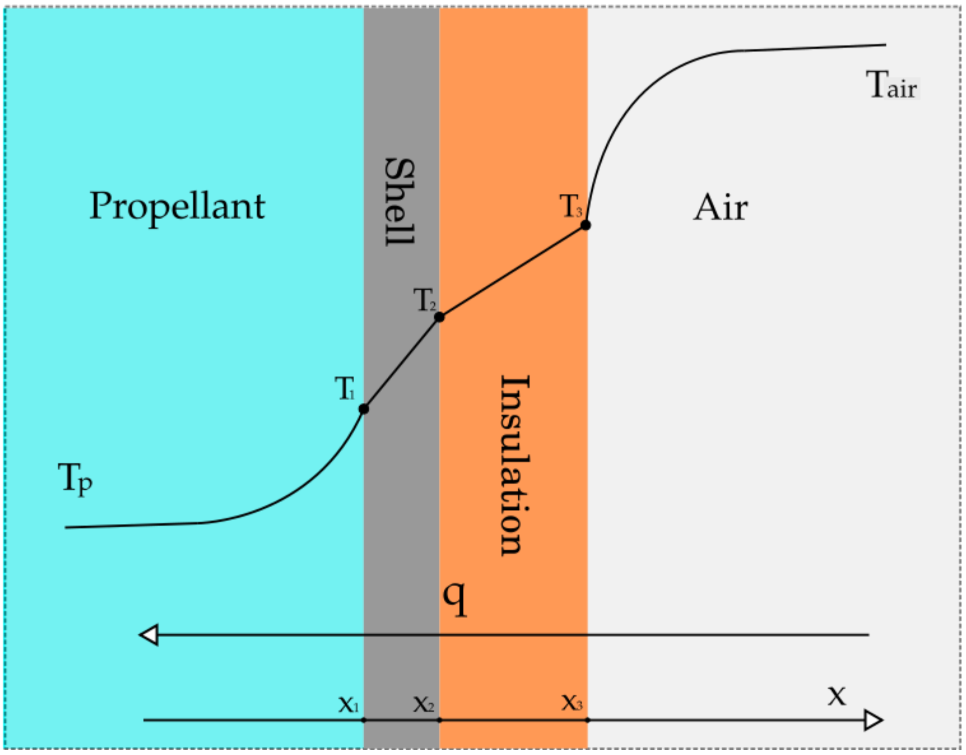
$$Nu_L = \frac{hL}{k} \tag{3.37}$$

To approximate the Nusselt number the condition of the heat transfer has to be determined. In the case of the convective heat transfer between the propellant and the wall, natural convection on a vertical wall can be assumed. This means that the movement of the fluid is induced by the heating of the fluid which makes it move upwards due to gravitational accelatration. Churchhill and Chu suggest the relationship in equations 3.38 to 3.42 to approximate Nusselt's number for free(natural) convection along a vetical plate.[11] Where Nusselt, Prandtl, Rayleigh and Grasshoff number are Nu, Pr, Ra and Gr respectively. c_p, μ and k are specific heat, dynamic viscosity and thermal conductivity. And in equation 3.42; g, β, ν

[10]J.H. Lienhard IV and J.H. Lienhard V. *A Heat Transfer Textbook*. 4th. Version 2.11. Cambridge, MA: Phlogiston Press, 2017. URL: `http://ahtt.mit.edu`.

[11]Stuart W Churchill and Humbert HS Chu. "Correlating equations for laminar and turbulent free convection from a vertical plate". In: *International journal of heat and mass transfer* 18.11 (1975), pp. 1323–1329.

Figure 3.3: *Schematic illustration of heat transfer in the tank wall.*



and L are acceleration, thermal expansion coefficient, kinematic viscosity and height of the cylinder wall respectively.

$$Nu_L = 0.68 + 0.503(Ra_L \cdot \Psi)^{0.25} \ : \ Ra_L < 10^{10} \tag{3.38}$$

$$\Psi = \left(1 + \left(\frac{0.492}{Pr_L}\right)^{9/16}\right)^{-16/9} \tag{3.39}$$

$$Pr_L = \frac{c_p \mu}{k} \tag{3.40}$$

$$Ra_L = Pr_L \cdot Gr_L \tag{3.41}$$

$$Gr_L = \frac{g\beta L^3 (T_1 - T_p)}{\nu^2} \tag{3.42}$$

As can be seen in equation 3.38 this relation is only valid for Rayleigh numbers below 10^{10}. This is where problems arise for this implementation into the model. A quick calculation for LOX shows that Rayleigh numbers are in the order of 10^{12} - 10^{17}. This makes these formulas unusable for usage in cryogenic tanks. Table 3.2 shows an example calculation with liquid oxygen. It was therefore chosen to let the model define a certain insulation thickness as input instead since further calculation of required insulation thickness was deemed outside of the scope of the project due to its complexity.

Table 3.2: *Rayleigh number example calculation for LOX, data taken from NIST database.*

Input		Results	
c_p	$924[J/kg/K]$	Pr	0.75
μ	$1.13 \cdot 10^{-5}$	Gr	$4.5 \cdot 10^{13}$
k	0.014015	Ra	$3.43 \cdot 10^{13}$
g	$9.81[m/s^2]$		
β	$0.00692[1/K]$		
L	$5[m]$		
T_1	$143[K]$		
T_p	$97[K]$		
ν	$2.92 \cdot 10^{-6}[m^2/s]$		

ii. Ice Buildup

The external wall of the propellant tank can experience ice buildup when storing cryogenic propellant while being grounded. Moisture in the air condenses and freezes stuck to the surface prior to lift-off. This is an undesirable situation as it creates additional mass which is lifted of the ground which affect the performance of the LV. An example is the centaur upper stage (AC-14) with an accumulated ice buildup of $28/[kg]$ of which $24/[kg]$ is shaken off/ ablated during launch. This is around approximately 1.3% of the stage's dry weight.[12]

V. Anti Slosh and Vortex Baffles

Anti slosh and vortex baffles haven't been modeled, the proper modeling of necessary structure involves CFD (Computational Fluid Dynamics). This is obviously not in the scope of this project and would go against the requirement of limiting computational resources since CFD is in general a big consumer of computational resources. Literature on methods to evaluate sloshing behavior through less complex computations are found to be scarce. The project had three different directions concerning anti slosh and vortex baffles.

- Model sloshing and vortex behavior through simple computations and approximate necessary baffles
- Model sloshing and vortex baffle mass by guesstimates of their size and dimension
- Not model anti slosh and anti vortex baffle

The final option is mentioned (even though it is not desirable) since the project is time restricted and if there are not enough resources than this might be unavoidable.

[12]Lewis' staff, see n. 11.

Chapter 4

Determination of Forces & Moments

I. Introduction

This chapter shows the methods which were used to determine forces and moments which are acting on the LV for a given load case. Compression and momentum loads are important to determine buckling criterion for structures, this is described in the next chapter. First the determination of these forces and moments is described.

First forces acting on the launch vehicle were described together with a description of load cases. This is followed up with a section containing information about how compression force at a specific point was determined. The chapter is concluded with a section which describes how moment at a specific location was determined in the model.

II. Forces and load cases

Two main forces acting on the launch vehicle during flight are shown in figure 4.1. These forces are plotted as two main forces in opposite direction; aerodynamic force(F_a) and thrust force(F_T). These two main forces are not necessarily exactly opposite of each other but one of the figure's goals is to illustrate the existence of compression loads in the structure. And this is obviously achieved by having two forces acting in opposite direction while pointing towards each-other. Other forces not illustrated are gravity and lateral thrust and aerodynamic forces.

A load case where the two main forces are not exactly opposite creates a momentum around the center of mass of the launch vehicle. An example of this is lateral thrust created by combustion instability, throttling or thrust vectoring. This moment induces an angular acceleration and creates compression and tension stresses in the structure.

Trust To start with the thrust force F_T; thrust in reality is created by the pressure distribution on the surface of an engines thrust chamber and nozzle manifold. For simplicity the model sees these pressure distributions as a point force with a magnitude and direction.

This point force is determined by the engine parameter which indicates maximum thrust, number of engines, thrust angle and a throttling parameter β. The model calculated maximum thrust of a single stage with the following equation:

$$F_{T,tot} = \sum_{i=1}^{N} F_{T,i} \tag{4.1}$$

Where N is the number of engines and $F_{T,i}$ is the maximum thrust of i^{th} engine. The model can thus handle multiple engines but is restricted to global movement and throttling of them. What is meant by that can be described with the following equations which indicate determination lateral ($F_{T,x}$) and axial thrust ($F_{T,z}$).

$$F_{T,x} = F_{T,tot} sin(\theta)\beta \tag{4.2}$$

$$F_{T,z} = F_{T,tot} cos(\theta)\beta \tag{4.3}$$

In these equations, θ is the thrust angle and β is the throttling parameter. Thrust angle is measure between the z axis and the direction of the thrust force, e.g. a thrust angle of zero degrees would result in a lateral force which is also zero. Throttling parameter β is defined as having a positive value equal to or between one and zero:

$$0 \leqslant \beta \leqslant 1 \tag{4.4}$$

This thus mean that individual engines of one stage cannot have different thrust angles within the model, i.e. they always point in the same direction. This also accounts for throttling; all engines of a stage have to be throttle to the same value at a certain time-frame. The model does not support difference in throttle value at a specific time.

Aerodynamics Aerodynamic forces in figure 4.1 are the accumulation and integration of the pressure distribution over the skin of the rocket. This results in a point load with a magnitude and direction. This method of displaying aerodynamic forces comes in handy when discussing things like flight and orbital mechanics. This method is however less practical for structural analysis.

As discussed in the previous chapter under the fairing section: calculation of aerodynamic drag and lift is difficult without extensive aerodynamic analysis or computational simulation. These two reasons were key factors for the decision of a different approach to determining aerodynamic forces.

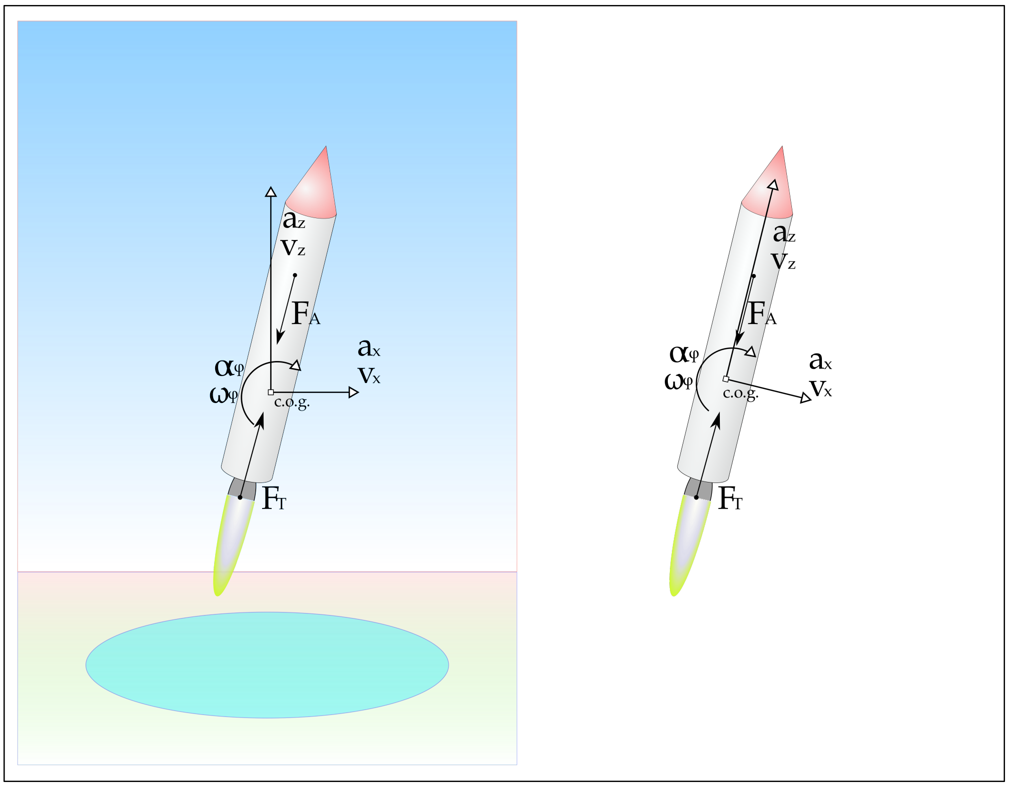
The decision was made to apply load factor as a possible input for a load case as well as longitudinal aerodynamic forces directly. This has the disadvantage that lateral aerodynamic forces (lift) get removed from the model. This sacrifice in capability was deemed acceptable since launch vehicles often fly close to a zero degree angle of attack. Load factor l was defined as follows:

$$l = \frac{F_{T,z} - F_D}{M_{wet}} \tag{4.5}$$

Where $F_{T,z}$ is the thrust in axial direction, F_D is the drag force and M_{wet} is the launch vehicles wet mass. The advantage to having a load factor as input was its value is known somewhat accurately and is often set to a maximum value as a requirement for a launch vehicle, since this value affects structural integrity of the launch vehicle but also its payload. Aerodynamic drag can than be calculated if all other values are known:

$$F_D = F_{T,z} - lM_{wet} \tag{4.6}$$

Figure 4.1: *Indication of difference between earth based (left) and vehicle based (right) reference frame.*



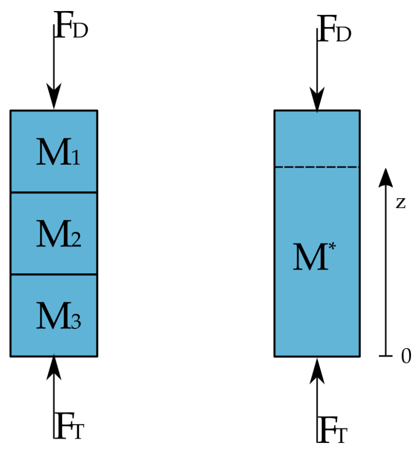
Wet mass of the launch vehicle is obviously not a known variable since dry mass is not known. Wet mass can be approximated however since propellant mass is in general much larger than dry mass. Also a first approximation was made for dry mass by putting the design criterion for propellant tanks at design pressure and to give other structural parts a default thickness. This way the model can iterate towards a solution by updating wet mass and in turn drag force.

At a certain points during the ascent trajectory of the launch vehicle the structural load case has to be determined. Some variables which are included in this load case were already mentioned:

- total thrust $[N]$
- thrust vector $[deg]$
- load factor $[-]$ /drag $[N]$
- throttle $[-]$
- propellant left in the tank $[kg]$

One last variable which has not been mentioned yet is propellant left in the tank. This is a variable important to rockets since their weight dramatically decreases during flight. For example when a load factor is used to calculate aerodynamic drag and thus compression loads and structure mass, the wet mass has to be approximated. It is thus relevant for the load case how much propellant is left. It was thus decided that this variable is also taken as input when calculating required thickness for a specific load case.

Figure 4.2: *Variation of compression loads through a structure in increments (left) and continuous (right).*



III. Compression force at a specific point

When thrust and drag are known the model needs to determine the compression force at a location z due to the axial thrust and aerodynamic drag. This value is used to determine the required thickness of the wall at this position by determining its yield criterion.

This would be trivial if these two forces had the same value which then creates a load case with a load factor of zero. This would have as consequence that the compression load would be constant throughout the launch vehicle, i.e. it would be the same at every location z. This is however rarely if ever the case since the rocket needs to accelerate at a significant rate in order to achieve orbit. This creates a situation where a positive load factor is often a property of a load case. Compression loads thus come to vary with location z and to achieve these values the mass distribution of the LV has to be known.

The reason mass distribution has to be known to calculate compression loads is due to acceleration of the structure. This means that the sum of all forces in the axial direction is not equal to zero. This is illustrated in figure 4.2 and equation 4.7. Equation 4.7 shows the case for a structure which is divided into incremental mass elements this is shown in the left side of figure 4.2.

$$\sum F_z = (M_1 + M_3 + M_3)a_z \neq 0 \tag{4.7}$$

Assuming the thrust force is larger than the drag force. To calculate the compression load between M_3 and M_2 on the left side of figure 4.2, the mass above M_3 has to be calculated. In this example this is trivial since it is simply:

$$M_{above3} = M_1 + M2 \tag{4.8}$$

Since the acceleration of the structure is known from equation 4.7 the force needed to accelerate M_1 and M_2 can be calculated with Newton's second law.[1] which is equal to the compression load at this location.

$$F_{load,2-3} = (M_1 + M_2)a_z \tag{4.9}$$

[1]Isaac Newton. "Principia mathematica". In: *Newton's principia* 634 (1934).

This example with increments was used to illustrate the concept of the method used to calculate the load force. When thrust force is larger than the drag forces the launch vehicle accelerates upwards. The compression load close to the thrust vector (engines) experience the largest load factor is this case. These load will be close to the magnitude of the thrust vector. The compression loads at the top close to the drag vector will be smaller and close to the drag value.

Using mass increments was deemed to be not desirable for use in the model since compression load at a specific location can not be determined. It would then only be possible to calculate the load at preset locations. It was decided to use a continuous approach so any location could be evaluated when necessary.

The continuous approach is sketched in figure 4.2 on the right side of the figure. To keep consistent notation, the bottom of the "vehicle" in the figure has a z value of zero (its origin). M^* is indicated as the mass below the dashed line (z).

The mass below the dashed line is calculated by integrating the mass over the length of the launch vehicle. The relation of this integration can be seen in equation 4.10. This equation shows that the change of mass with respect to location z needs to be known to calculate M^*. The model determines this parameter and integration numerically and mass below the dashed line is thus calculated.

$$M^* = \int_0^z dM = \int_0^z \frac{dM}{dz} dz \tag{4.10}$$

Once M^* is known the force required to accelerate this mass to the acceleration of the entire vehicle can be determined:

$$F^* = M^* a_z \tag{4.11}$$

Compression load becomes:

$$F_{load} = F_T - F^* \tag{4.12}$$

The most difficult part of this method is the determination of $\frac{dM}{dz}$. This is done by known location and mass of different subsystems and an assumed distribution of mass within subsystems. The methodology used is described in more detail in chapter 6, where the model is explained.

IV. Moment at specific point

Determining the moment load acting on a specific location is more complicated than the compression force due to axial loads. This is caused by additional parameters which need to be known in order to complete the calculation.

i. Force, shear and moment line

Determining the moment at a given location z on the LV requires the information of forces acting on the object. The model makes use of force, shear and moment lines which determine stress acting on the object a a location z. An example of this can be lateral thrust of the rocket engine due to thrust vectoring or a force due to wind on the launchpad. The latter can be seen in figure 4.3. The figure shows the wind which is represented as an equally distributed load

Figure 4.3: *Approximation of force on LV due to wind on the launchpad.*

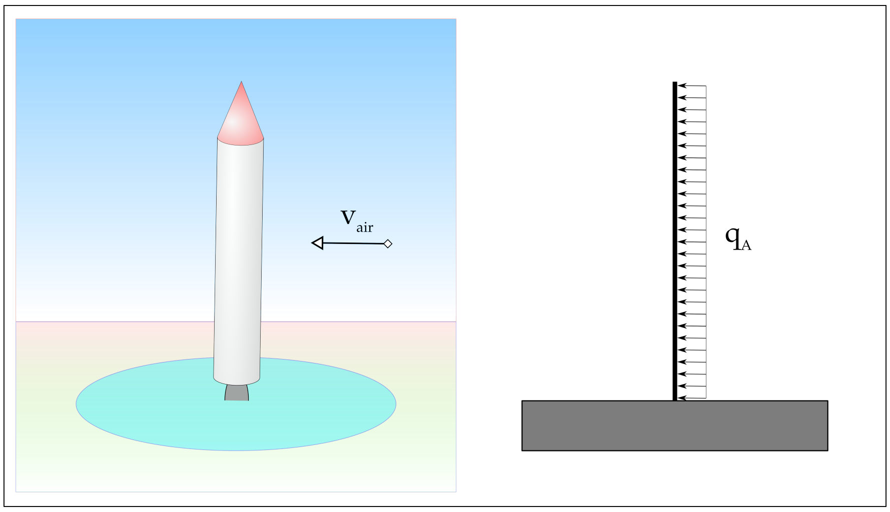
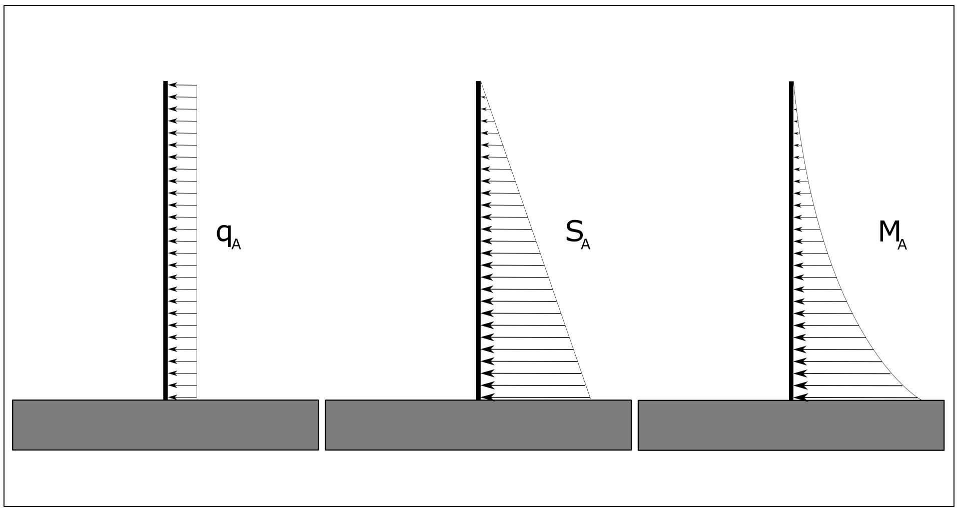


Figure 4.4: *Example of the implementation of the force, shear and moment line.*



across the lenghth of the LV. This representation is a force line, where the LV is anchored to the launchpad and thus creating a cantilever beam. This approximation is solely to determine the moment and shear force of a particular location. From this force line the shear and moment line can be determined by integrating. Integrating the force line once forms the shear line and integrating the force line twice becomes the moment line. This process is illustrated in figure 4.4. More information about determining the shear and moment line can be found in the appendix of the report, this is the same method which is incorporated into the model.

ii. Rocket in flight

As can be seen from the previous section, the moment at a specific location for a rocket on the launch pad can be calculated in a rather straight forward method. All forces should be known and most importantly is the sum of all forces equal to zero since there is no acceleration taking place.

This is different for a rocket in flight, there are accelerations. Axial acceleration was dealt with in a previous section which analyzed the compression loads. and lateral accelerations where deemed to be approximately zero for when a zero degree angle of attack trajectory is implemented. This leaves angular acceleration to be discussed and the effect it has on the structural mass of the LV.

Angular acceleration is induced by lateral thrust, either due to thrust vectoring, combustion instability or other forces. This lateral thrust generates a moment (τ) around the center of mass ($z_{c.o.g.}$) on the launch vehicle. Where its magnitude can be determined as follows:

$$\tau = F_{lat} z_{c.o.g.} \tag{4.13}$$

In order to calculate angular acceleration α, Newton's second law is used.[2] This relation is shown by equation 4.14. In this equation I_{xx} is the mass moment of inertia around the x axis, with the x axis placed at the location of the center of mass $z_{c.o.g.}$.

$$\tau = \alpha I_{xx} \tag{4.14}$$

For the model it was assumed that the launch vehicle is axis symmetric around its z axis. This means that:

$$I_{xx} = I_{yy} \tag{4.15}$$

and:

$$I_{xy} = I_{yx} = I_{zx} = I_{xz} = I_{yz} = I_{zy} = 0.0 \tag{4.16}$$

Therefore no tensor calculations are necessary to complete the calculation in the model. In order to calculate the mass moment of inertia I, defined by equation 4.17. It can be observed that the center of gravity (mass) has to be determined, in order to calculate the mass moment of inertia. Since rotation act around its axis.

$$I_{xx} = \int_0^M (z - z_{c.o.g.})^2 dM \tag{4.17}$$

Please note that equation 4.17 assumes that the launch vehicle is long and somewhat thin. This is means that it is assumed that every every x and y coordinate for a given z coordinate has approximately the same value. Hence, the distance from the center of gravity is purely expressed in the z dimension.

iii. Center of gravity

Calculation of the position of the center of gravity will be discussed in more detail in this section. The LV is assumed to be axis symmetrical for this calculation which means that it is assumed that its x and y coordinates of the center of gravity (mass) are 0 (zero). This makes the problem less complicated by restricting the problem to one dimension, namely the z direction. This confines the problem to equation 4.18 for a body and equation 4.19 for a collection of particles,i.e. point masses. M in this equation is the total mass and m_i is the mass of the i^{th} particles.

$$z_{cog} = M^{-1} \int_0^M z dm \tag{4.18}$$

[2]Newton, see n. 1.

$$z_{cog} = M^{-1} \sum_{i=1}^{N} m_i z_i \tag{4.19}$$

Both equations are used in the model. The way this works is that parts/subsystems are created as objects which C.O.G. and mass are calculated separately by using equation 4.18. These objects are then treated as point masses and summed up to determine the total center of gravity by using equation 4.19.

For full rocket stage modelling this calculation has to take into account the mass of the propellant which is left at a certain time or load case. This variable naturally has an effect on the height of the propellant(fluid level) in its tanks. This height is unknown but and has to be calculated in order to determine the C.O.G. of the propellant in the tank. This can be achieved by a similar equation to 4.18 and is shown in equation 4.20. This equation in itself has no z value(only mass), this means that mass has to be rewritten to a function with z in it. The way this done depends on the geometric shape and is described for several shapes in the appendix. To illustrate this the method for a cone is written out here in equations 4.21 to 4.24. This works because the mass is known as an input parameter and height h can thus be calculated.

$$M = \int_0^M dm \tag{4.20}$$

$$dm = \rho dV \tag{4.21}$$

$$dV = \pi r^2 dz \tag{4.22}$$

$$r = tan(\alpha) z \tag{4.23}$$

thus:

$$M = \pi \rho tan(\alpha)^2 \int_0^h z^2 dz \tag{4.24}$$

Once the center of gravity has been determined the mass moment of inertia can be calculated. The model used an numerical integration method to achieve this, similar to the center of mass calculation. Both the integrals can be tuned within the model. This is done to give the user power over precision versus calculation time.

iv. Angular acceleration

Once the moment acting on the LV has been determined (τ) as well as the mass moment of inertia around the x axis, the angular acceleration of the object can be determined using 4.14. This means that the the entire structure undergoes this angular acceleration (α) if an infinite stiff structure or time solution is assumed. It was determined that the structure cannot be assumed to be infinitely stiff but it was decided to implement a time independent or so-called static solution. This means that for simplification dynamic loading is not taken into account in the model. A simplified way of looking at a static solution is that the moment τ is performed for a long enough duration for the entire vehicle to achieve the same angular acceleration α.

Since moment and forces cannot be evaluated in an accelerating object, the model needs to implement fictitious forces. This is done in a similar manner which was described for axial forces. In order to do this for angular acceleration it was decided to move the reference frame.

Reference Frame In order to solve the equations to calculate stress in the structure accelerations need to be converted into fictitious forces. This is achieved by moving the orgin of the reference frame to the center of gravity of the LV. This is paired with rotation of the reference frame equal to the rotation of the LV. In other words; this creates a reference frame in which the LV is not moving but is however experiencing forces due to accelerations, i.e. fictitious forces. This in essence creates a structure that is not moving relative to the reference frame and thus experiences static loading.

The decision to move the reference frame towards the center of gravity is based on the assumption that the structure rotates around this point during flight. This also means the structure does not experience and force due to angular acceleration or angular velocity at this point. This makes it easier to convert these properties into fictitious forces. A disadvantage of this method is that the C.O.G. (center of gravity) shifts significantly for a rocket in flight and this shift thus has to be taken into account. This means that the model had to determine the C.O.G. for every unique load case.

Calculation of the C.O.G. is done by calculating the C.O.G of every object and multiplying theses values with their position in the original reference frame. This is summed up and divided by the total mass to get the C.O.G. This was explained in the previous section about center of gravity.

Fictitious Force The magnitude of the fictitious forces due to angular acceleration at a certain location z had to be determined. Since angular acceleration, location of the C.O.G., and location z are known the lateral acceleration can be determined by the following equation.

$$a_x = \alpha * (z - z_{cog}) \tag{4.25}$$

when lateral acceleration is determined the fictitious force counter acting the acceleration due to the structures mass can be determined in a similar method to equation 4.11.

$$dF^* = dM^* a_x \tag{4.26}$$

In equation 4.26 the value dM^* stands for the mass for an increment of the structure. To calculate this the model uses a function with calculation the change of mass with respect to the z coordinate ($\frac{dM^*}{dz}$). The equation thus becomes:

$$dF^* = \frac{dM^*}{dz} a_x dz \tag{4.27}$$

Rearranging the equation where q stand for distribute load with a unit of $[N/m]$:

$$\frac{dF^*}{dz} = q^*(z) = \frac{dM^*(z)}{dz} a_x(z) \tag{4.28}$$

With knowledge of the fictitious force at every location the moment acting at a certain location can be determined by adding the moment all fictitious forces apply to a certain location.

$$\tau(z) = \int_0^z q^*(z)(z - z_{cog}) dz \tag{4.29}$$

This then gives the moment acting on the structure at a given location z.

Chapter 5

Determination of Yield Criteria

This chapter discusses the limiting loads to which the structure was held. These limiting loads have to be above the loads being applied to the LV in order for the structure to maintain structural integrity. The model thus has to give structural walls sufficient thickness to withstand these limiting loads.

I. Stresses

In order to determine yielding forces and moments in a given structure one has to look at stresses occurring during these loads. Stresses occurring due to axial loads are quite easy to obtain for a beam, since it is simpley the total force devided by the area of the beam.

$$\sigma_F = \frac{F}{S} \tag{5.1}$$

For thin walled cylinders this becomes.

$$\sigma_F = \frac{F}{2\pi r t} \tag{5.2}$$

Euler-Bernoulli beam method Euler-Bernoulli beam method was applied in a rather straight-forward method to determine stress occurring due to bending forces. The method to determine stress is focused around equation 5.3. This equation determines the stress at a cross-section of the beam. Where τ is the moment at that location, I is the second moment of inertia and z is the distance from the neutral axis. This neutral axis is parallel to axis around which the moment applies its force. In the case of symmetrical geometry, (e.g. a circle) the neutral axis will coincide with the moment axis.

$$\sigma_\tau = \frac{\tau z}{I} \tag{5.3}$$

Stresses due to axial load and moment can simply be combined to obtain the total stress.

$$\sigma_{total} = \sigma_F + \sigma_\tau \tag{5.4}$$

i. Composite material

Composite materials are supported by the model. These composite materials are modeled as layers of isotropic or orthotropic material which have properties in the z and x direction. The model was designed to build up a wall with layers of different material if desired. And the model is thus flexible in changing parameters related to the structural wall.

An example of this can be a sandwich panel which consist of an carbon fiber reinforced plate, aluminum honeycomb and another plate of carbon fiber reinforced plastic. This wall will then have different elastic constant when the wall is considered one composite material. The method by how theses parameters are calculated can be found in the next section about buckling.

Orthotropic material properties are expressed in the global reference frame of the LV (z and x). This made is possible to evaluate stresses occurring in the material and analyze buckling behavior. Composites are however not always layered parallel to these global coordinates. An example of this is carbon fibre reinforced plastic lamina. Since this material can take any direction parallel to the wall surface, i.e. $0 - 180[deg]$.

Since properties of these kinds materials are often only given in their local coordinate system (for CFRP tape: parallel and perpendicular to the fibers), their properties have to be converted to the global coordinate system. This is done with the following formulas:[1]

$$E_z = \frac{E_1}{m^4 + \left(\frac{E_1}{G_{12}} - 2\mu_{12}\right)n^2m^2 + \frac{E_1}{E_2}n^4} \tag{5.5}$$

$$E_x = \frac{E_2}{m^4 + \left(\frac{E_2}{G_{12}} - 2\mu_{12}\right)n^2m^2 + \frac{E_2}{E_1}n^4} \tag{5.6}$$

$$\mu_{zx} = \frac{\mu_{12}(n^4 + m^4) - \left(1 + \frac{E_1}{E_2} - \frac{E_1}{G_{12}}\right)n^2m^2}{m^4 + \left(\frac{E_1}{G_{12}} - 2\mu_{12}\right)n^2m^2 + \frac{E_1}{E_2}n^2} \tag{5.7}$$

$$\mu_{xz} = \frac{\mu_{21}(n^4 + m^4) - \left(1 + \frac{E_2}{E_1} - \frac{E_2}{G_{21}}\right)n^2m^2}{m^4 + \left(\frac{E_2}{G_{21}} - 2\mu_{21}\right)n^2m^2 + \frac{E_2}{E_1}n^2} \tag{5.8}$$

$$G_{zx} = \frac{G_{12}}{n^4 + m^4 + 2\left(\frac{2G_{12}}{E_1}(1 + 2\mu_{12}) + \frac{2G_{12}}{E_2} - 1\right)n^2m^2} \tag{5.9}$$

$$n = sin(\theta) \tag{5.10}$$

$$m = cos(\theta) \tag{5.11}$$

θ in these formulas is the angle the fiber would have with respect to the global z axis. n and m are taken as temporary variables to make the equations more readable.

[1]George Z Voyiadjis and Peter I Kattan. *Mechanics of composite materials with MATLAB*. Springer Science & Business Media, 2005.

ii. Tensile failure

Previous chapter about tank design express tensile stress due to internal pressure as a function of pressure, wall thickness and radius.

$$\sigma(P, t, r) \tag{5.12}$$

This works well for isotropic materials since for thin walled structures it was assumed that the in-plane stress was equal throughout the thickness. Material yield stress could thus easily be linked to these variables, which would make determining the required thickness straight forward.

For composite materials this procedure is more difficult since various materials have different yield stresses, but more over different materials can experience different stresses at the same location due to a difference in Young modulus E. Stress obtained from pressure, thickness and radius can thus not be directly linked to material constants. This stress has to be evaluated as the average stress in the wall denoted by $\bar{\sigma}$. Which is the sum of the stresses in all layers divided by the number of layers, assuming all layers have the same thickness:[2]

$$\bar{\sigma}(P, t, r) = \frac{1}{N}\sum_{j=1}^{N}\sigma_j \tag{5.13}$$

Here N is the number of layers and j indicates the layer number. Average stress expressed in force per unit width of the laminate becomes:

$$N_x = \bar{\sigma}_x t N \tag{5.14}$$

To solve the problem one has to make an assumption whether any variable is constant for all layers. This variable is strain ϵ, which should be approximately the same for all all layers. Strain defined by stress and young modulus:

$$\epsilon = \frac{\bar{E}}{\bar{\sigma}} \tag{5.15}$$

Force per unit width and strain of a laminate can be related for symmetric balanced laminates as follows:[3]

$$\begin{bmatrix} N_x \\ N_z \\ N_s \end{bmatrix} = \begin{bmatrix} A_{xx} & A_{xz} & 0 \\ A_{zx} & A_{zz} & 0 \\ 0 & 0 & A_{ss} \end{bmatrix} \begin{bmatrix} \epsilon_x \\ \epsilon_z \\ \gamma_s \end{bmatrix} \tag{5.16}$$

Solving for strain by calculating the inverse of matrix $[A]$, $[a]$:

$$\begin{bmatrix} \epsilon_x \\ \epsilon_z \end{bmatrix} = \frac{1}{A_{xx}A_{zz} - A_{xz}A_{xz}} \begin{bmatrix} A_{zz} & -A_{xz} \\ -A_{zx} & A_{xx} \end{bmatrix} \begin{bmatrix} N_x \\ N_y \end{bmatrix} \tag{5.17}$$

$$\gamma_s = \frac{N_s}{A_{ss}} \tag{5.18}$$

The parameters of matrix A are based upon parameters of the stiffness matrix $[Q]$ of each lamina in the laminate and is defined as follows:

$$A_{ij} = \sum_{k=1}^{N} Q_{ij}^{k} t_k \tag{5.19}$$

[2]Voyiadjis and Kattan, see n. 1.

[3]M Daniel Isaac and Ori Ishai. "Engineering mechanics of composite materials". In: *New York and Oxford* (1994).

with Q_{ij} defined in the lamina reference frame:

$$Q_{11} = \frac{E_1}{1 - \mu_{12}\mu 21} \tag{5.20}$$

$$Q_{22} = \frac{E_2}{1 - \mu 12\mu_{21}} \tag{5.21}$$

$$Q_{12} = \frac{\mu_{21}E_1}{1 - \mu_{21}\mu_{12}} \tag{5.22}$$

$$Q_{66} = G_{12} \tag{5.23}$$

Since the parameters of matrix $[A]$ are defined in the global reference frame the variables Q_{ij} have to be converted from the lamina reference frame to the global reference frame. $m = cos(\theta)$ and $n = sin(\theta)$ for the following formulas.

$$Q_{xx} = m^4Q_{11} + n^4Q_{22} + 2m^2n^2Q_{12} + 4m^2n^2Q_{66} \tag{5.24}$$

$$Q_{zz} = n^4Q_{11} + m^4Q_{22} + 2m^2n^2Q_{12} + 4m^2n^2Q_{66} \tag{5.25}$$

$$Q_{xz} = m^2n^2Q_{11} + m^2n^2Q_{22} + (m^4 + n^4)Q_{12} - 4m^2n^2Q_{66} \tag{5.26}$$

$$Q_{ss} = m^2n^2Q_{11} + m^2n^2Q_{22} - 2m^2n^2Q_{12} + (m^2 - n^2)^2Q_{66} \tag{5.27}$$

Since all parameters should now be known for a given lay-up (fiber orientations), E_1, E_2, μ_{12} and μ_{21}. The strains ϵ_x and ϵ_z can be calculated.

To determine if the laminate will hold under the normal tensile loads the strain of each lamina has to be checked to make sure it does not exceed its maximum strain value. This is done by converting strain in the global coordinate system back to the local coordinate system of the lamina. Local strain are calculated using a transformation matrix $[T]$.

$$\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ 0.5\gamma_{12} \end{bmatrix} = \begin{bmatrix} m^2 & n^2 & 2mn \\ n^2 & m^2 & -2mn \\ -mn & mn & m^2 - n^2 \end{bmatrix} \begin{bmatrix} \epsilon_x \\ \epsilon_z \\ 0.5\gamma_{xz} \end{bmatrix} \tag{5.28}$$

if ϵ_1 or ϵ_2 exceed ϵ_1^{yield} or ϵ_2^{yield} respectively for any lamina in the laminate, then the laminate was determined to fail under this load. These values can be determined with the following formulas. In these formulas σ_1 and σ_2 can be replaced by the respective material yield stresses. Since equations 5.29 and 5.30 show that strain is depended on stress in both directions, solutions exist where maximum strain is not exceeded while maximum directional stress is exceeded.

$$\epsilon_1 = \frac{E_1}{\sigma_1} + \mu_{12}\frac{E_2}{\sigma_2} \tag{5.29}$$

$$\epsilon_2 = \frac{E_2}{\sigma_2} + \mu_{21}\frac{E_1}{\sigma_1} \tag{5.30}$$

This leads to a more iterative problem where for a certain ply thickness and lay-up, one can check if material yield stresses are exceeded. This approach does however not fit well with the requirements of the model to calculate required ply thickness without using excessive amounts of cpu resources. A more suitable approach would be the determination of a maximum average stress for a laminate which cannot be exceeded (see equation 5.14). Since this section only affects failure under tensile stress it was decided to forfeit the second part of equations 5.29 and 5.30. This leads to the following simplified formulas for maximum allowable strain.

$$\epsilon_1^{yield} = \frac{E_1}{\sigma_1^{yield}} \tag{5.31}$$

$$\epsilon_2^{yield} = \frac{E_2}{\sigma_2^{yield}} \tag{5.32}$$

Equations used for the composite material calculations above were taken from Vitudadhus & Kattan[4] and Isaac & Ishai.[5]

II. Buckling

A significant portion of stresses occurring in the structure of a LV are compression stresses. This is due to the engine thrusting the LV forward while aerodynamic forces combined with the LV mass counteract this motion. Another cause of compression can be bending of the structure due to thrust vectoring of the rocket engine or again, due to aerodynamic forces. Compression stresses can be dealt with in a similar manner to tensile stresses by using a material yield stress. This however only applies if, and only if no buckling behavior takes place due to compression of the material. Buckling is a failure mode in which the material deforms due to instability under compression. This phenomena is more significant in long and thin structures where it can be the main mode of failure. It is thus obvious that buckling criteria need to be discusses when modeling structural components since this fits the description of a LV.

Buckling can be a complex to model phenomena which in current day is often simulated by a computer using finite element methods. These simulations are however not available for the model in this report as it would go against the research objective to limit computation time. It is therefore necessary to use more classical methods which determine a yield criterion analytically. The formula used to determine the buckling criteria are taken from reference[6] and.[7] In the first reference Peterson describes various buckling criteria for thin walled cylinders, and a full overview of all buckling formulas used can be found in the appendix. This chapter will describe the method used for a cylinder solely under axial compression which is represented by equation 5.33. In this equation D is the flexural stiffness per unit width and is described in equation 5.34. Here is E defined as young modulus, t as the shell thickness and μ as the materials Poisson ratio. L is the length of the cylinder and k_z is the buckling coefficient and described by equation 5.35. In this equation m is the number of buckle half waves in the axial direction, β is the buckle aspect ratio seen in equation 5.36, with n being the number of buckle waves in the circumferential direction and r being the radius of the cylinder shell. γ is a coefficient to adjust theoretical buckling results to realistic values. Z is a curvature parameter given by equation 5.37.

$$N_z = k_z \frac{\pi^2 D}{L^2} \tag{5.33}$$

[4]Voyiadjis and Kattan, see n. 1.

[5]Isaac and Ishai, see n. 3.

[6]JP Peterson, P Seide, and VI Weingarten. "Buckling of thin-walled circular cylinders". In: (1968).

[7]VI Weingarten and P Seide. "Buckling of thin-walled truncated cones". In: *NASA Space Vehicle Criteria (Structures), NASA SP-8019, Washington DC* (1968).

$$D = \frac{Et^3}{12(1-\mu^2)} \tag{5.34}$$

$$k_x = m^2(1+\beta^2)^2 + \frac{12}{\pi^4}\frac{\gamma^2 Z^2}{m^2(1+\beta^2)^2} \tag{5.35}$$

$$\beta = \frac{nL}{\pi m r} \tag{5.36}$$

$$Z = \frac{L^2}{rt}\sqrt{1-\mu^2} \tag{5.37}$$

As can be observed from the formula's on the next pages these equations are quite complex. More importantly however is that they are not computation heavy for a computer compared to FEM analyses. The value of N_z is expressed in force per unit distance. This distance is equal to force per unit length of the circumferential distance from the cylinder. To calculate the required thickness the model relies on a simple, robust solver for the critical load. The compression load is given as input and the problem is solved by using a bi-section method to find the root of the following equation, where d is the diameter of the cylinder:

$$N_z(t) * \pi * d - F_{compressive} = 0 \tag{5.38}$$

This simple algorithm which gives the required thickness of the cylinder shell for a given force. Similar methods are used for other buckling scenarios including cylinders or cones with internal pressure, bending loads or a combination of these. Different formulas are used for orthotropic and sandwich materials, but they follow a similar methodology, they are described in the orthotropic cylinders section of this chapter.

Moment Since a moment can also be applied to the structure of the vehicle the buckling behavior of this load also had to be analyzed. This is performed in an actual quite simple manner by multiplying the maximum axial load per unit width of circumference by 0.75 to obtain the maximum load due to momentum.[8]

$$N_{x,m} = 0.75N_x \tag{5.39}$$

where $N_{x,m}$ is the maximum load per unit width of circumference due to a moment acting on the structure. Since axial stress due to a moment can be calculated as follows.

$$\sigma_z = \tau r \tag{5.40}$$

with r as the cylinder radius, and:

$$N_{x,m} = \sigma_z / t \tag{5.41}$$

where t is the thickness of the wall. maximum applied moment thus becomes:

$$\tau_{max} = 0.75N_x t r \tag{5.42}$$

[8]Peterson, Seide, and Weingarten, see n. 6.

Combining forces Since the structure will often experience both moment and axial loads simultaneously the combination of both yield criteria had to be considered. This was done by assuming a linear distribution between the maximum moment and axial force.[9] This means that both loads cannot fractional be larger than one:

$$1 = \frac{\tau}{\tau_{max}} + \frac{N_x}{N_{x,max}} \tag{5.43}$$

i. Orthotropic cylinders

This section describes the buckling formulas used for orthotropic cylinders. The resulting formulas carry a similar result to equation 5.33. They are however all inclusive, which means they can take into account sandwich walls, stiffeners and rings. The formulas presented are taken from Weingarten and Seide.[10]

Axial compression Weingarten and Seide describe buckling load per unit of length of circumference for orthotropic cylinders in the following manner.

$$N_z = \left(\frac{L}{m\pi}\right)^2 \frac{\begin{vmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{vmatrix}}{\begin{vmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{vmatrix}} \tag{5.44}$$

Equation 5.44 is only valid for a number of circumreferential buckling waves (n) which is equal or larger than four. L is the length of the cylinder, m is the number of axial buckling half-waves and all the terms of the matrices are defined as follows:

$$A_{11} = \bar{E}_z\left(\frac{m\pi}{L}\right)^2 + \bar{G}_{zx}\left(\frac{n}{r}\right)^2 \tag{5.45}$$

$$A_{21} = A_{12} = (\bar{E}_{zx} + \bar{G}_{zx})\frac{m\pi n}{Lr} \tag{5.46}$$

$$A_{22} = \bar{E}_x\left(\frac{n}{r}\right)^2 + \bar{G}_{zx}\left(\frac{m\pi}{L}\right)^2 \tag{5.47}$$

$$A_{13} = A_{31} = \frac{\bar{E}_{zx} m\pi}{rL} + \bar{C}_z\left(\frac{m\pi}{L}\right)^3 + (\bar{C}_{zx} + 2\bar{K}_{zx})\frac{m\pi n^2}{Lr^2} \tag{5.48}$$

$$A_{32} = A_{23} = (\bar{C}_{zx} + 2\bar{K}_{zx})\left(\frac{m\pi\sqrt{n}}{L\sqrt{r}}\right)^2 + \frac{\bar{E}_z n}{r^2} + \bar{C}_x\left(\frac{n}{r}\right)^3 \tag{5.49}$$

$$A_{33} = \bar{D}_z\left(\frac{m\pi}{L}\right)^4 + \bar{D}_{zx}\left(\frac{m\pi}{L}\right)^2\left(\frac{n}{r}\right)^2 + \bar{D}_x\left(\frac{n}{r}\right)^4 + \frac{\bar{E}_x}{r^2} + \frac{2\bar{C}_x}{r}\left(\frac{n}{r}\right)^2 + \frac{2\bar{C}_{zx}}{r}\left(\frac{m\pi}{L}\right)^2 \tag{5.50}$$

The above equations use elastic constant with a bar above them. This is to indicate that these constant are not material constants but depend on the composition of the wall as well as potential use and size of stiffeners and rings. Since many variables are used in the above equation a list was compromised of the meaning of each variable.

[9] Peterson, Seide, and Weingarten, see n. 6.
[10] Weingarten and Seide, see n. 7.

$\bar{E}_z, \bar{E}_x$ -	wall extensional stiffness
$\bar{G}_{zx}$ -	wall shear stiffness
$\bar{C}_{zx}, \bar{K}_{zx}, \bar{C}_z\bar{C}_x$ -	Coupling constants (for orthotropic walls)
$\bar{D}_z, \bar{D}_x$ -	wall bending stiffness
$\bar{D}_{zx}$ -	wall twisting stiffness

The definitions used for these elastic coefficient for othrotropic walls comes also from Weingarten and Seide.[11] The equations are listed below and were used in the model. The equations also handle possible stiffeners and/or rings. These can be distinguished by their subscripts *r* and *s* for rings and stiffeners respectively. The two parameters *b* and *d* denote stiffener and ring spacing per unit width respectively. The equations basically add parameters of various layers within the wall. *N* denotes the number of layers in the wall and *j* indicates the specified layer.

$$\bar{E}_z = \sum_{j=1}^{N} \left(\frac{E_z}{1-\mu_z\mu_x} \right)_j t_j + \frac{E_s S_s}{b} \tag{5.51}$$

$$\bar{E}_x = \sum_{j=1}^{N} \left(\frac{E_x}{1-\mu_z\mu_x} \right)_j t_j + \frac{E_r S_r}{d} \tag{5.52}$$

$$\bar{E}_{zx} = \sum_{j=1}^{N} \left(\frac{E_z\mu_x}{1-\mu_z\mu_x} \right)_j t_j \tag{5.53}$$

$$\bar{G}_{zx} = \sum_{j=1}^{N} (G_{zx})_j t_j \tag{5.54}$$

$$\bar{D}_z = \sum_{j=1}^{N} \left(\frac{E_z}{1-\mu_z\mu_x} \right)_j \left(\frac{t_j^3}{12} + t_j\tilde{y}_j^2 \right) + \frac{E_s I_s}{b} + \tilde{y}_s^2 \frac{E_s S_s}{b} \tag{5.55}$$

$$\bar{D}_x = \sum_{j=1}^{N} \left(\frac{E_z}{1-\mu_z\mu_x} \right)_j \left(\frac{t_j^3}{12} + t_j\tilde{y}_j^2 \right) + \frac{E_r I_r}{d} + \tilde{y}_r^2 \frac{E_r S_r}{d} \tag{5.56}$$

$$\bar{D}_{zx} = \sum_{j=1}^{N} \left(4G_{zx} + \frac{\mu_z E_x}{1-\mu_z\mu_x} + \frac{\mu_x E_z}{1-\mu_z\mu_x} \right)_j \frac{t_j^3}{12} + t_j\tilde{y}_j^2 + \frac{G_s J_s}{b} + \frac{G_r J_r}{d} \tag{5.57}$$

$$\bar{C}_z = \sum_{j=1}^{N} \left(\frac{E_z}{1-\mu_z\mu_x} \right)_j t_j\tilde{y}_j + \frac{\tilde{y}_s E_s S_s}{b} \tag{5.58}$$

$$\bar{C}_x = \sum_{j=1}^{N} \left(\frac{E_x}{1-\mu_z\mu_x} \right)_j t_j\tilde{y}_j + \frac{\tilde{y}_r E_r S_r}{d} \tag{5.59}$$

$$\bar{C}_{zx} = \sum_{j=1}^{N} \left(\frac{E_z\mu_x}{1-\mu_z\mu_x} \right)_j t_j\tilde{y}_j \tag{5.60}$$

$$\bar{K}_{zx} = \sum_{j=1}^{N} (G_{xy})_j t_j\tilde{y}_j \tag{5.61}$$

[11] Weingarten and Seide, see n. 7.

One parameter not specified in the previous paragraph is $\tilde{y}$. This parameter indicates the distance from a specific layer to an arbitrary reference line. This line was chosen to be on the outer wall for the model but can be chosen to be at any distance,$\tilde{y}_j$ thus becomes:

$$\tilde{y}_j = \sum_{k=1}^{j-1} t_k + 0.5t_j \tag{5.62}$$

Please note that the distance measured is from the bending axis of the layer. If $\tilde{y}_s$ or $\tilde{y}_r$ has to be determined the entire wall can be summed and half the stiffener or ring height should be added.

ii. Buckle wave numbers

As can be seen in equations 5.35 and 5.36, critical buckling load depends on buckle waves. These buckle waves are divided into axial half waves (m) which can be any positive real number and circumferential buckle waves (n) which is always an integer. This creates a scenario where the cylinder can theoretically buckle in infinite many modes, these modes all have a critical buckling load. Figure 5.1 visualizes this effect by plotting critical load versus several wave number values. The figure plots a simulation of a simple aluminum isotropic cylinder with a radius of $1[m]$, length of $2[m]$ and a thickness of $2[mm]$. The plot stops at $450[kN]$ but has much higher values around its peaks, the lowest value of $172[kN]$ which occurs at a value of 16 and 4 for n and m respectively. The figure shows a valley of low critical loads and this was noticed to be a general trend among various configurations.

Finding this minimum value is crucial to determining the buckling yield criterion. This is in practice the value at which the cylinder will experience instabilities and collapse due to compression loads. For some configurations the minimum value can be approximated. An example of this is the formula mentioned in reference[12] which states that for isotropic cylinder of moderate length the buckling coefficient can be calculated as follows:

$$k_z = \frac{4 * \sqrt{3}}{\pi^2}\gamma Z \tag{5.63}$$

This simplifies the problem significantly for this case and critical buckling load can be calculated reliably and quick. This method is however no help to stiffened or orthotropic cylinders. For theses configurations the buckle wave values have to be tuned to find the minimum load at which buckling occurs.

Optimizing Buckling waves The model has no optimization algorithm implemented for the determination of the minimum buckling load for every buckle wave. The algorithm finds the minimum value of a set of predetermined buckle wave value not unlike the calculation plotted in figure 5.1.

[12]Peterson, Seide, and Weingarten, see n. 6.

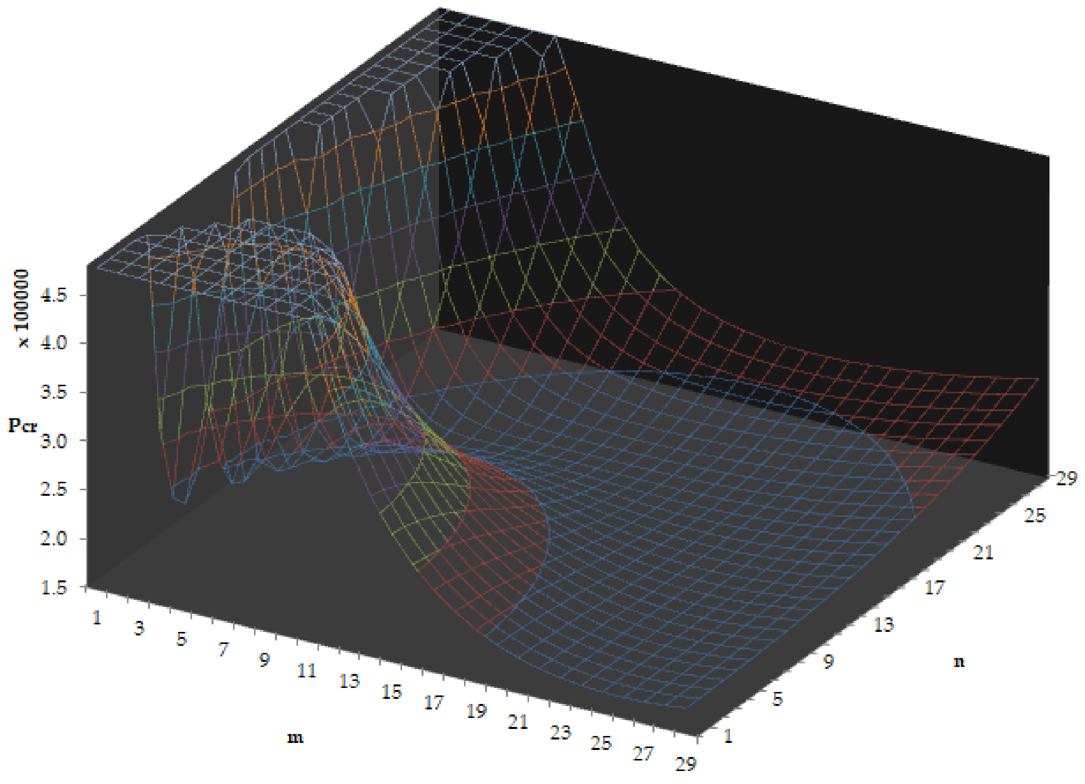


Figure 5.1: *Theoretical critical buckling load of isotropic cylinder plotted versus number of buckle half waves in axial direction - m, and number of buckle waves in circumferential direction - n.*

Chapter 6

Model Description

As described previously, the model was given shape by programming it in the TUDAT toolbox using the C++ programming language. This chapter shall give an overview and explanation of method used and the way the model was coded. This starts with an overview of the core program and gradually explains more detailed sections of the model. Methods used to calculate certain parameters are discussed by describing the method itself and how they fit into the entire program.

I. Classes

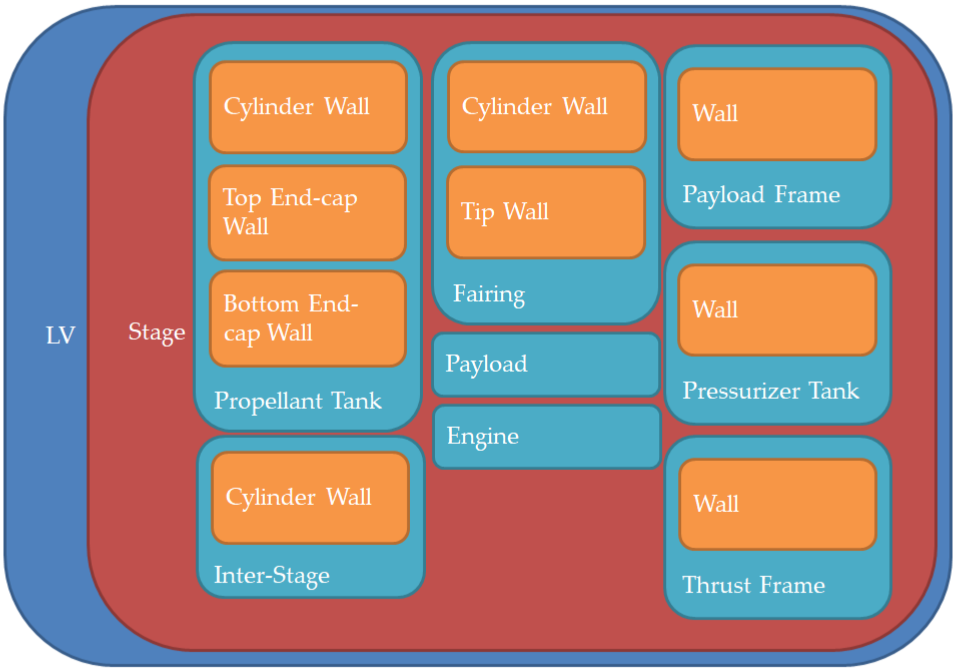
C++ is a programming language which stems from the older language C. C++ is however vastly different and in contrast to C uses classes to get stuff done. This method of programming is often called object oriented programming (OOP) and its goal is to make software development easier. The report does not explain OOP in more detail nor does it go into depth about explaining the C++ language any further. This section is meant to describe the classes created and used in the program. For further information about the language and OOP the author refers to different literature.[1]

Launch Vehicle The LV class is at the core of the program and contains all information about the LV. This ranges from the number of stages to load case parameters. The Launch vehicle object also contains a lot of other objects as members. A large object contained by the LV object is the Stage class. This class is stored in the LV object as a vector of pointers to several stage objects. This makes it easier to dynamically change the number of stages in an LV. The Stage class carries other objects in order to contains all information of a single stage. And overview of the launch vehicle class can be seen in figure 6.1. The figure shows the launch vehicle class and in this case one stage class object. This stage class then has various different objects from different classes which represent various subsystems within the rocket stage. The subsystems have another class object themselves which is called the wall class. Please note that not all class objects are represented in figure 6.1, the purpose of this figure is to give an overview. To extrapolate on the last statement, the wall class shown in the figure actually also contains one or several material objects which is also a separate class. This makes the total number of levels of object members in the LV class five.

The launch vehicle class needs to know what stages are in it and what load-case is acting on it. It needs to know how to stack build stages on top of each other and thus a rough idea on

[1]Stephen R Davis. *C++ for Dummies*. John Wiley & Sons, 2009.

Figure 6.1: *Overview of the launch vehicle class*



each stage geometry.Furthermore it needs to keep track of the mass distribution of the entire launch vehicle and forces and moment acting on the LV at each point along its z direction.

Stage The stage class needs to keep track of all subsystems within its stage. These subsystems can be categorized into two groups:

- structural subsystems
- non-structural subsystems
  - affecting stage geometry
  - not affecting stage geometry

Both groups should be well defined based on their name. In the model the first group; structural subsystems can be recognized by the fact that they have the wall class as one of its members. Changing the parameters of any structural subsystem will inadvertently change the geometry of the stage itself. For example, changing the length of a tank increase the length of the stage as well. The second group; non-structural subsystem can be further divided into two groups. This is a group which does affect stage geometry and a group which does not. The model distinguished between these groups to execute the build process correctly.

There are only two non-structural subsystems which affect the geometry of the stage. The first subsystem is the engines. This subsystem increases the length of the stage and its length has a great effect on the mass of the inter-stage. The second subsystem in the payload class, this class affects the size of the fairing and therefore its mass. All other non-structural subsystems do not affect the geometry of other subsystems. These subsystems do however have a location and mass. This means that even though they do not have an effect on geometry they do affect thickness and thus mass of structural subsystems.

Tank The tank class holds all information regarding tank geometry and other parameters such as mass, ullage volume, internal pressure. The tank class can be adapted to be integrated into the structure, this meas that for propellant tanks the choice can be made to have a double or single wall on the cylindrical section of the tank. In case of a double wall the tank object will have an extra wall object added to it. This wall will then support the external axial loads and moments acting on the structure. The inner wall will withstand the pressure exerted by the propellant and pressurization gas on the tank wall like the top and bottom end-cap.

The tank class also holds information regarding its propellant which is contained by the propellant class. An object of this class in embedded into the tank class and holds information about is density and temperature. Another aspect the propellant object holds is a minimum and maximum temperature at which to store the propellant to avoid unrealistic storage temperatures. This was mostly effective to keep cryogenic fluids below their boiling point and above their freezing point. The tank class was also designed to generate its own coordinates in its own reference frame which origin is location at the "tip" of the tank bottom end-cap.

Fairing The fairing class holds information about the shape of the nose-cone, its length and the total length of the fairing structure. The class carries functions to calculate external pressure based on a drag and its geometry as input. In order to calculate its mass the class has functions to calculate the surface. The class uses global buckling functions to determine necessary thickness of the chosen walls. The fairing class was also designed to generate its own coordinates based in its own reference frame. The origin of the reference frame is located in the center at the bottom of the cylindrical section of the fairing.

Inter-stage The inter-stage class carries its own wall properties and generates is own coordinates like the fairing, tank and more classes. It calculates is mass based upon axial load and moment as input combined with its length. The inter-stage differs from some other classes upon building its coordinates since it need information from the object above and below it to correctly make a decision about its required length to avoid collision of objects. with length, surface can be determined. From surface the required thickness is calculated with global buckling function.

Thrust-frame The thrust-frame class needs, like the inter-stage class information about the object above and below it. It communicates with the stage class to figure out the order of tank and from there retrieves information of the tank above which it sits. The class also needs to communicate with the stage class object in which it sits, about the number and diameter of the stages' engines. As described in chapter 2, the thrust-frame calculates its coordinates based upon the objects above and below but also a design decision can be made if the frame is attached to the hull of the rocket or directly attached to the propellant tank.

Thickness of the thrust-frame is determined by buckling analysis of its cone shape. The buckling force is determined from the engines to which it is attached, e.g. the thrust-frame of the second stage of a hypothetical launcher will have its necessary structural thickness based upon the thrust force of the engine(s) of the second stage. Buckling analysis is called form within the class which calls a global buckling function that requires input about the geometry, wall, etc. This input is naturally supplied by the thrust-frame object.

Payload-frame The payload-frame is similar to the thrust-frame in its functions and variables. In essence it is modeled the same with the exception that the payload-frame's features are reverted with respect to a thrust-frame's features. It's geometry is based upon the payload and the tank below it. The same design decision can be made to attach the frame to the hull or directly to the tank. The force acting on the pay-load frame comes from the mass of the payload and the force it is acting upon the frame due to inertial forces.

Payload The payload is obviously not considered an important part in modeling the structure of the LV it is thus modeled as a simple cylinder with a height radius and mass. These variables are input when initialing an object of the payload class. These parameters have the biggest effect of the geometry of the payload-frame and the fairing and the thickness(thus mass) of the payload-frame.

Engine The engine class, like the payload frame is rather simple in geometry. Engines are modeled as a cylinder with a predetermined diameter and length from object initialization. Engine mass is assumed to be linearly distributed throughout its length. Engine objects can be initialized in multiples for a single stage at once. The engines then get bundled according to optimal circle packing in a circle.[2] Engine mass can be predetermined during initialization of the class or by using empirical relations which approximate engine weight[3].[4]

Wall The wall class is essential to the model since it is incorporated into every structural subsystem mentioned above. The basic functionality of the wall class is to construct a wall from certain materials. This way the model can incorporate relative complicated wall structures while maintaining flexibility, i.e. wall parameters can quickly and easily be changed without a lot of rewriting of code.

One of the core functions of the wall class is the `addLayer` function. This function adds a layer of specified material to the wall. The `addLayer` function is overloaded[5], and can call two different functions. The simple version for isotropic material is shown below.

```
void Wall::addLayer( Materials *materials)
{
    mp.push_back(materials);
    tvec.push_back(&t);
    layers++;
}
```

This function has one pointer as a variable which is a of the material class. This material pointer is then pushed to the back of a vector containing all materials in the wall in the correct order. The second line in the function adds a pointer to the thickness `t` to the back of a vector containing the thickness of every layer of material. Lastly the number of layers is updated to add one, this way the class knows how many layers of material it has.

The wall class has a private variable called `t` which control the thickness of all structural layers. This means that all structural layers with the exception of rings and stiffeners have the same length. This was done to make implementation of composite layered materials such as CFRP easy to implement. Every layer will then thus have the same thickness.

[2]Eric W. Weisstein. *Circle Packing*. 2017. URL: `http://mathworld.wolfram.com/CirclePacking.html`.

[3]BTC Zandbergen. "Simple mass and size estimation relationships of pump fed rocket engines for launch vehicle conceptual design". In: (2015).

[4]Zandbergen, "Aerospace Design and Systems Engineering Elements I, Part Launcher Design and Sizing", see n. 4.

[5]The same function name is used to call different functions

Different functions are used to add rings or stiffeners and an extra core layer can also be added for sandwich panels. This core layer does have the ability to have a different thickness than the structural layers. The vector of pointers to the thickness was implemented to easily change thickness of the structural walls by just changing one private variable. This works since all pointers for structural walls point to the `t` private variable of the wall class.

The overloaded function can also be called with an extra floating point number:

```
void Wall::addLayer(double theta, Materials *mat)
{
    ...
    ..
    .
    Materials* matpoint = new Materials(...,...,...,...,Ez,Ex,G,mux,muz);

    mp.push_back(matpoint);
    tvec.push_back(&t);
    layers++;
}
```

This `double` floating point variable called theta, indicates the angle under which the material is placed compare to the global z-axis and the materials prime axis. This is only used for orthotropic materials such as Carbon or Glass fiber composites. The code is not shown in full since the function is quite long. The rest of the function deals with converting material constants in its local plane to global constants. The function creates a new material with the correct global coordinates and the function then points towards this for further reference.

A final responsibility the wall class carries out is the calculation of the density per square meter of wall. This function can be called from within another object containing a wall object to calculate the objecst mass. The object which carries a wall object can obviously also set/update parameters concerning thickness of the wall to the required value.

Material The material class contains material constants such as Young modulus, Poisson ratio and yield stress. Material objects were initialized in a material database header file which contains all materials used and is initialized at the start of the program.

The material class has an overloaded constructor. One constructor initializes an object for isotropic material. The other constructor function creates an object for orthotropic material where material constants can be given in different directions.

Propellants & Pressurizers Lastly there are propellant and pressurizer classes which contain data about each respectively. This data is in the form of minimum and maximum allowable temperature and their density shift within these ranges of temperatures. These classes only store data.

II. Building the launch vehicle

Building of the launch vehicle is done from within the launch vehicle class and extends itself to basically all other objects which represent subsystems of a rocket stage. The process starts of fairly basic with the function looping through its stages beginning with the first stage and working its way up. First the function calls another function within the i^{th} stage and commanding it to build the stage. After this the program calculates the height of the stage with the `calcVerticalShift(int)` function. This way the LV knows how much to shift the stage upon with the previous build stage must rest. Lastly an if statement is called to build an inter-stage for the previous stage. This function has to be called after the stage above it is build, since it relies on data from the stage above (Tank and engine(s) geometry).

```
void Launchvehicle::buildLV()
{
    double verticalShift =0.0;
    for (int i=0;  i< numberOfStages ; i++)
    {
        cout << "building stage: " << i + 1 << endl;
        stage[i]->buildStage(verticalShift);
        cout << "finshed building stage: " << i + 1 << endl;
        verticalShift = calcVerticalShift(i);

        if (i > 0)
        {
            buildInterstage(*stage[i-1],*stage[i]);
        }
    }
}
```

Building the stage in the `buildStage(double)` function is more complex and will not be fully printed here due to its length. The beginning of the function is shown below. The function starts with initializing a $3x6$ matrix called the `transformMatrix`. This function is used to determine the location of the rocket stage's subsystems in the launch vehicle coordinate system. The transform matrix' coordinates are set the vertical shift in the z direction to adjust for possible stages below the current stage which is building, the rest of the transform matrix coordinates are set to zero.

The function continuous by executing the `bundleEngines()` function, this function is discussed in more detail below. After this the real building of the stage begins by starting at the bottom of the stage and working upwards. Engine coordinates in the LV frame of reference are stored in the stage class, this way the stage class knows object coordinates in both the stage and LV reference frame. On line 10 the program loops through every engine in the stage. On line 12 the program shifts the coordinates of the engines in their own reference frame to the LV reference frame by adding the transform matrix. After the for loop every engine is places in the correct position according to user input.

The program builds the stage subsystems (engine in this case) by calling subsystem class functions of creating the subsystems own coordinates in its own coordinate system. It then linearly transforms those local coordinates to global LV or stage coordinates. Please note that the function call to get engine coordinates in the local reference frame is not shown below. This function is present within the `bundeleEngines()` function.

After the for loop coordinates of the just placed engines are called and temporary put into a matrix variable. This variable is used to create a new transform matrix on line 16 & 17. This way a new subsystem (thrust-frame) can be added on top of the engine(s).

```cpp
void Stage::buildStage(double verticalShift)
{

    Eigen::MatrixXd transformMatrix(3,6);
    transformMatrix << Eigen::MatrixXd::Zero(2,6),
                           Eigen::MatrixXd::Constant(1,6,verticalShift);

    // put engines at the bottom
    bundleEngines();
    for (int i = 0 ; i < numberOfEngines ; i++)
    {
        *(engineCoordinates[i]) = *(engineCoordinates[i]) + transformMatrix;
    }

    Eigen::MatrixXd EC = *(engineCoordinates[0]);
    transformMatrix << Eigen::MatrixXd::Zero(2,6),
                       Eigen::MatrixXd::Constant(1,6,EC(2,3));
    ...
    ..
    .
```

The rest of the `buildStage` function adds subsystems in a similar manner to the engines and updating the transform matrix as it "climbs" higher up the stage. exceptions are made to the last stage to add a payload and fairing other-wise an inter-stage is constructed within the launch vehicle class. At the end of the `buildStage` function propellant levels within the tanks are calculated. This is described in more detail below in a section dedicated to this function.

The `bundleEngines()` function is described below. This function starts by calling a engine object for its coordinates in its own local reference frame. The create "subsystem" coordinates function on line 4 exist for every subsystem. The function builds the subsystem, saves the local coordinates in the object and return the local coordinates to the caller. In this case the local coordinates are stored as a variable named EC.

The function creates a separation angle over which the engines are space, e.g. 120 degrees for 3 engines. After this the magnitude of the radius is calculated on line 7,8 and 9. This is done by optimizing circle packing within a circle to determine the distance.[6] After this every engine is looped through to shift their coordinates in the x and y plane. This is done with a special function on lines 16,17 and 18. This transformation function shifts coordinates in a polar coordinate system linearly by an angle and radius to another polar coordinate system.

[6]Weisstein, *Circle Packing*, see n. 2.

```
void Stage::bundleEngines()
{
    // ASSUMING EVERY ENGINE HAS SAME DIAMETER AND LENGTH
    Eigen::MatrixXd EC = engine[0]->createEngineCoordinates();
    double engineRadius = EC(0,1);
    double seperationAngle = 2 * pi / numberOfEngines;
    double newRadius = (engineRadius *
                 CoordTransformations::getBundleRadius(numberOfEngines))
                 - engineRadius;
    for (int i=0 ; i < numberOfEngines ; i++)
    {
        double teta = seperationAngle * i;
        double radiusShift = newRadius;
        Eigen::MatrixXd dummyMatrix = engine[i]->createEngineCoordinates();

        dummyMatrix = CoordTransformations::
            PolarToPolarLinear(dummyMatrix,radiusShift,teta);
        *(engineCoordinates[i])= dummyMatrix;
    }
}
```

III. Plotting the launch vehicle

The launch vehicle is plotted as a means to validate the build process and to a lesser extend provide supporting evidence for center of mass and propellant level calculation correctness. This is further discussed in chapter 9 of the report. This section goes into more detail about the method of which the plot function were implemented.

Unlike most other functions, the plotting function happens outside the launch vehicle class. The plotting is done within the QT 5 framework using the dialog class to achieve its requirements. The dialog class is initialized in the main function of the program using the launch vehicle object created as function variable.

```
    Dialog w(LV); // plot and show the LV
    w.showMaximized();
```

Here the LV variable is of the launch vehicle class and passes the launch vehicle in its entirety to the dialog class. Once inside the dialog class the launch vehicle's coordinates are converted by scaling in order to ensure the launch vehicle fits on the screen in its entirety.

The dialog class also contains a private variable containing the view angle under which the LV is plotted along side a scalar factor. This view angle variable determines the angle under which the LV is plotted. The coordinates of the launch vehicle are taken together with the view angle to project the three dimensional coordinates onto a two dimensional surface.

One of the advantages of these transformations stored as variables is the ability to dynamically change these variables in the plotting window. This means with implementation of a view buttons (seen in figure 6.2), the launch vehicle can be panned up, down, left and right.

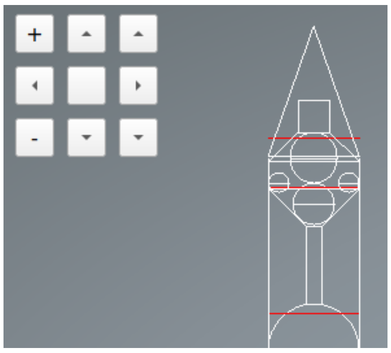


Figure 6.2: *Control buttons in the top left corner of the plotting window.*

Other functions include zooming in and out with the plus and minus buttons to view the LV in more detail. The buttons on the top and bottom right adjust the viewing angle. This in effect rotates the launch vehicle. Rotation can be used to verify the correct execution of the bundeling of engines for example.

The actual plotting of the launch vehicle is done by using functions and libraries within the QT framework (most notably QPainter) to connect lines and ellipses between coordinates of subsystems.

IV. Calculating Propellant level

Calculation of propellant levels is important for determining the center of mass and consequently determining the mass moment of inertia of the launch vehicle. The function calculating propellant levels resides within the tank class and is called during the build process. Propellant left is a floating point variable used as input from the launch vehicle class, this in passed onto the stage class to finally be passed to the tank class. The code looks as follows:

```
void Tank::calcPropellantLevel(double PL)
{
    double propellantMassLeft = PL * propellantMass;
    double dz = 0.01;
    double calcPropMass =0.0;
    double bottom;
    if (positionMatrix(2,0) <= positionMatrix(2,1))
    {bottom = positionMatrix(2,0);}
    else
    {bottom = positionMatrix(2,1);}
    double z = bottom;
    int i;
    while (propellantMassLeft > calcPropMass)
    {
        calcPropMass += contourFunction(z)*dz;
        z += dz;
        i++;
    }
    propellantLevel = z;
}
```

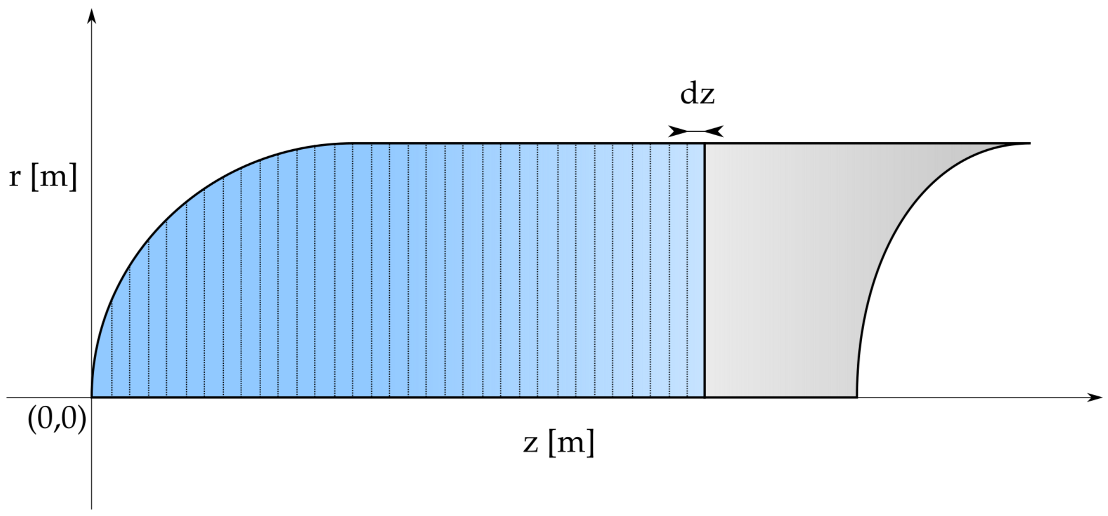


Figure 6.3: *Integration method used to calculate propellant level.*

The `double` variable PL stands for propellant left and is a value from zero to one indicating the mass fraction left in the tank. This fraction is calculated to mass left by multiplying it with the total capable propellant mass in the tank on line 3. After initializing some variable the program executes an if-else statement on line 7 to 10. This code finds the bottom of the tank, i.e. the coordinate with the lowest z value.

After determination of the bottom most part the code executes a while loop where propellant mass is numerically integrated over a distance z. When this propellant mass during integration reaches the earlier calculated mass the value of z is equal to the propellant level when assuming no errors of any kind take place.

The way integration works is through the `contourFunction(double)` on line 15. This function return the surface density of the tank at a certain location z. This is done by creating a function which calculates the contour of the tank. This can be represented by a graph with radius r on the vertical and z distance on the horizontal axis, as seen in figure 6.3.

Figure 6.3 shows calculation of the propellant level of a tank with elliptical tank cap where the propellant level is in the cylindrical part. The model can handle elliptical and conical caps which can both be face outward (left cap in figure) or facing inward (right side of figure).

After a function for the contour is constructed a the value is integrated by surface of revolution. This means the code calculates the surface of the cylinder and multiplies that by the density of the propellant to obtain the surface density.

V. Calculating center of mass

Calculation of the location of the center of mass (c.o.m. or c.o.g.) is done with a function held by the launch vehicle class. The core function is listed below. The algorithm starts with calling a function which updates the total mass of the launch vehicle. This step is necessary to make sure masses of all relevant objects are checked to see if they have been updated. Then after initialization of some temporary variables calculation of the c.o.m. starts by integrating over the length of the object (z direction).

```
1  void Launchvehicle::calcCOM()
2  {
3      updateWetMass();
4      COM = 0.0;
5      double calcMass = 0.0;
6      double height = stage[numberOfStages-1]->fairingCoordinates(2,3);
7      double dz = height/Settings::COMCalculationSlices;
8      double MPM=0.0;
9      for (double z = 0 ; z < height ; z+= dz)
10     {
11         MPM = MassPerMeter(z+(0.5*dz));
12         calcMass += MPM;
13         COM += (MPM*z);
14     }
15     calcMass = calcMass * dz;
16     COM = COM * dz / wetMass;
17 }
```

At line 11 in the illustrated code the function calls the `MassPerMeter(double)` function. This function returns derivative of the mass function if it was plotted against the height of the launch vehicle. This algorithm comes from equation 6.1 which describes the location of the center of mass.[7]

$$z_{cog} = \frac{1}{M_{wet}} \int_0^{height} z\rho dz \tag{6.1}$$

This formula is approximated by the algorithm with numerical approximation. Where N is the number of slices over which the launch vehicle is divided and dz is defined as the height divided by N.

$$z_{cog} = \frac{dz}{M_{wet}} \sum_{i=1}^{N} rho_i(z_i)(z_i + 0.5dz) \tag{6.2}$$

The algorithm also calculates mass of the launch vehicle through this method to check to error in the calculation when comparing it to the wet mass of the launch vehicle. This is an indication of the accuracy of the integration method.

VI. Calculating moment of inertia

Calculating the mass moment of inertia (m.o.i.) is done with the equations shown in chapter 4 of this report. To clarify the main equation is shown below.

$$I = \int_0^M (z - z_{com})^2 dM = \int_0^z (z - z_{com})^2 \frac{dM}{dz} dz \tag{6.3}$$

The parameters $\frac{dM}{dz}$ is already a function discussed earlier and is incorporated into the program as the `MassPerMeter(double)` function. Approximating the integral with numerical integration leads to the following code:

[7]Eric W. Weisstein. *Geometric Centroid*. 2017. URL: `http://mathworld.wolfram.com/GeometricCentroid.html`.

```
void Launchvehicle::calcI()
{
    MOI =0.0;
    double height = stage[numberOfStages-1]->fairingCoordinates(2,3);
    double dz = height / Settings::MOICalculationSlices;
    for (double z = 0 ; z < height ; z+= dz)
    {
        MOI += MassPerMeter(z+(0.5*dz)) * (z - COM)*(z- COM);
    }
    MOI = MOI *dz;
}
```

The center of mass function has to be called before mass moment of inertia is calculated since this function relies on its outcome, i.e. the location of the c.o.m.. Calculation of mass along side of the mass moment of inertia was deemed not necessary since the center of mass calculation is similar in giving a value for the error in these calculations.

VII. Calculating Mass

Finally calculation of mass is discussed in this chapter. This is naturally to main goal of the program and starts with an iteration in the main function of the program. as previously discussed this iteration is necessary to converge to a solution since the load case may change after updating the mass of the launch vehicle. The `calcStructureMass()` function is called from the launch vehicle class (line 7). This is where the actual structural mass is calculated.

```
1
2       for (int i = 0; i < n ;i++)
3       {
4           cout << "-------- Mass Iteration " << i+1 << "---------" << endl;
5           LV.calcCOM();
6           LV.calcI();
7           LV.calcStructureMass();
8       }
```

The `calcStructureMass()` function is described below and starts with a for loop on line 6 after initialing some temporary variables. This for loop runs in a different direction than most other for loops in the model. This for loop starts at the top of the launch vehicle, i.e. the last stage and move downwards. This is not done without a reason. The main reason for doing this is to reduce the iterations needed in to converge to a solution for the structural mass in the main function.

Compression force acting on a structural object is mostly affected by the drag, the load factor and the objects above it. A more accurate axial compression load lead to calculation of structural thickness closer to the final solution. It is thus better to calculate the top objects first and let the program work its way down.

After initializing the for loop the program check if the current stage is the last stage and if so the program calculates the mass of the fairing and the pay-load frame. This is done by calculating the aerodynamic drag and load factor acting on the fairing and passing these values as variable to the calculation of dry mass function in the fairing object (line 12). After this the payload frame mass is calculated by passing the compression force created by the payload and acceleration to its mass calculation function (line 16).

```
1  void Launchvehicle::calcStructureMass()
2  {
3      double z=0.0;
4      double compForce = 0.0;
5
6      for (int i=numberOfStages-1 ; i >= 0 ;i--)
7      {
8          if (stage[i]->getLastStage())
9          {
10             // CALC FAIRING MASS
11             cout << "calculating fairing mass: ";
12             stage[i]->fairing[0]->calcDryMass(getAerodynamicDrag(),zGforce());
13             cout << "done" << endl;
14             cout << "calculating payload frame mass: ";
15             compForce = stage[i]->payload[0]->getMass() * zGforce();
16             stage[i]->payloadframe[0]->calcMass(compForce);
17             cout << "done" << endl;
18         }
19        else {
20             // CALC INTER STAGE MASS
21             ...
22             ..
23             .
```

The function continuous and is much longer than illustrated here. The function lets every structural subsystem calculate its mass based upon new inputs calculated from the `getAerodynamicDrag()`, `ZGforce()`, `compressiveForce(z)` and `moment(z)` function. At the end of the function an update function; `updateDryMass()` is called to get the newly calculated mass from all objects and update the total launch vehicle mass.

Chapter 7

Simulation Input

Various launch vehicles were used as input for the model. They are described in this chapter along with their input. The program uses a .cpp file called "database" in which all launch vehicles are stored. Propellants and pressurizers are stored in their respective class .cpp file. Material is stored in a header file called materialDatabase.h. Finally a header file which contains generals settings is used, named "settings.h".

This chapter first discusses general settings used to generate a launch vehicle. After this a section about the Atlas-Centaur launch vehicle is discussed and finally a section about a small modern liquid engine launcher is discussed.

I. General Settings

General settings are mostly placed within the settings header file in the program. The settings file used in displayed below, all values which are not dimensionless have SI units attached to them. The choice for integration constants around a value of 1000 to 2000 was to ensure numerical integration without large errors while maintaining a relative short processing time.

A safety factor of 1.25 was chosen for the entire structure. The value of the safety factor was determined rather arbitrary. This has a reason for it being a global safety factor where normally every subsystem would have its own safety factor based on system requirements or else. This safety factor is in line with the general standards for ultimate load, no safety factor could be found for the Atlas-Centaur launch vehicle in literature. The structure is thus designed to withstand the worst case scenario with a factor of 1.25 increase in force/moment/etc..

Stiffener and rings height and width are predetermined in the settings file to values which are representable for normal rings and stiffeners compared to the expected skin thickness of around 1 [mm]. This means skin thickness is the only variable which changes during the program run time.

Buckling wave numbers are taken to be the minimum buckling force or close to this minimum. These were chosen after iterating to find the right values. A solver can be used but this will affect computation, this can be decreased by implementing an optimization algorithm. This might be an addition to the model in the future.

Other general settings include some mathematical or astronomical constant, this can be seen under constants in the code below.

settings.h

```
//SETTINGS files
namespace Settings {

// general settings
static const double safetyfactor = 1.25;
static const int massCalculationIterations =2;
static const int COMCalculationSlices = 2000;
static const int propellantLevelSlices = 2000;
static const int MOICalculationSlices = 2000;
static const int MomentCalculations = 1000;


// stifferner and ring settings [m]
static const int numberOfStiffeners = 15;
static const double stiffenerHeight = 0.02;
static const double stiffenerWidth = 0.0005;
static const int numberOfRings =15;
static const double ringHeight = 0.02;
static const double ringWidth = 0.0005;
static const double Is = stiffenerWidth * pow(stiffenerHeight,3)/12.0;
static const double Ir = ringWidth * pow(ringHeight,3.0)/12.0;

//buckling
static const int numberOfBuckleHalfWavesM = 13;
static const int numberOfBuckleWavesN = 7;

// constants
static const double pi = boost::math::constants::pi<double>();
static const double R = 8.3144621; // gas constant
const double gEarth = 9.807; //[m/s2]
}
```

II. Atlas-Centaur

The Atlas-Centaur launch vehicle is a two stage launcher capable of carrying large payloads into space. Data used for input to simulate the launcher is mostly taken from a paper reviewing the flight performance of the Atlas-Centaur written by the Lewis Research Center.[1] This paper review the AC-13, AC-14 and AC-15. These are early version of the launch vehicle and this is thus what the model is simulating. Over the years the Atlas-Centaur LV has evolved significantly thus little resemblance to the earlier version remains, results from this simulation can thus not be used to resemble the current day version of this launcher.

The appendix of this report contains the full description of the code used to simulate the Atlas-Centaur launcher since it was deemed to long to put in the body of the report. Input not mentioned here can be found there.

[1]Lewis' staff, see n. 11.

Table 7.1: *Key variables Atlas-Centaur*

	Atlas	Centaur	
stage diameter	3.05	3.05	$[m]$
LOX tank pressure	3.4	3.2	$[bar]$
fuel tank pressure	6.0	1.9	$[bar]$
LOX temperature	97.0	97.0	$[K]$
fuel temperature	290	18.9	$[K]$
Load carrying LOX tank	true	true	
Load carrying fuel tank	true	true	
Common Bulkhead	true	true	
LOX tank above fuel tank	true	false	
Pressure tank location	engine(s)	engine(s)	
Pressure tank pressure	200	200	$[bar]$

Table 7.2: *Material data used for simulation of the Atlas-Centaur LV.*

Name	301 Full Hard SS	301 Half Hard SS	Alu Honeycomb	Alu 5086
$\rho[kg/m^3]$	7880	7880	32	2657
$\sigma_{yield}[MPa]$	965	758	10	225
$E[GPa]$	193	193	0.72	70.3
μ	0.27	0.27	0.3	0.33

i. Key Variables

The key variables chosen for the Atlas-Centaur are variables directly contributing to its build and are taken from existing literature and can be seen in table 7.1. Figure 7.1 shows the Atlas Centaur plotted by the model. All variables stated in the table are important factors in determining the structural dry mass of the launch vehicle.

ii. Material

Material data used in the simulation is shown in table 7.2. ρ indicates density, σ indicates tensile yield stress, E represents Young's modulus and μ represents Poisson's ratio. All materials used for the Atlas-Centaur LV is assumed to be isotropic. The aluminum honeycomb material is not used to withstand structural loads. This materials is only used as spacer for the glass fiber composite and it is assumed it carries no load. This is why it yield stress is composed in a way that is will not yield under any load.

iii. Engines

Three engines where used for the simulation of the Atlas-Centaur. The Centaur upper stage used two RL10A-3 engines manufactured by Prat & Whitney. The Atlas stage used one sustainer engine and two booster engines both developed and constructed by Rocketdyne. Data about the engines is collected from the report mentioned earlier in this section. The oxidizer to fuel ratios where used to calculated necessary propellant mass for the oxidizer and fuel in the liquid engine rocket stages. Engine data can be observed in table 7.3.

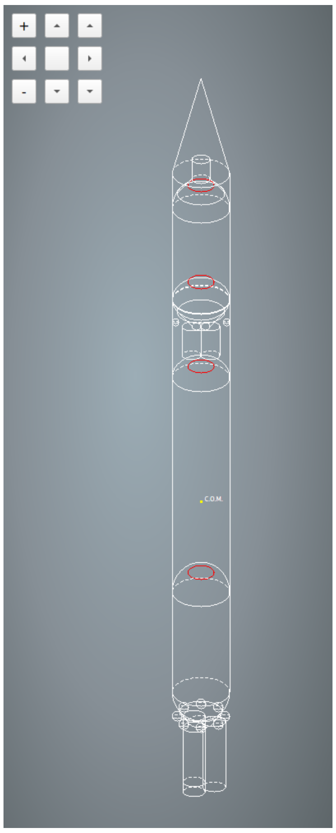


Figure 7.1: *Plot of the simulated Atlas-Centaur launch vehicle.*

Table 7.3: *Engine data used for simulation of the Atlas-Centaur LV.*

Name	RL10A-3	Sustainer Engine	Booster Engine
Manufacturer	Prat & Whitney	Rocketdyne	Rocketdyne
Thrust$[kN]$	66.7	258	747
Reported Mass$[kg]$	228	1300	1452
Calculated mass$[kg]$ ^{2}	142	313	852
Diameter$[m]$	1.0	1.2	1.2
Length$[m]$	1.5	3.43	3.43
$0/F$	4.83	2.23	2.23

2 - Mass is calculated using Zandbergen M-C1 and M-S1 relations

Table 7.4: *Propellant data used for simulation of the Atlas-Centaur LV.*

Name	RP-1	LH2	LOX
$T_{min}[K]$	270	14	54
$T_{max}[K]$	310	23	97
$\rho_{max}[kg/m^3]$	823	77	1305
$\rho_{min}[kg/m^3]$	792	68	1105

iv. Material allocation

Most structural parts in the AC-13, AC-14 and AC-15 were reported to use full hard 301 stainless steel. Exception are:

- **Fairing:** Aluminum honeycomb and aluminum sheet sandwich construction. The actual construction was of a glass fiber honeycomb structure but no reliable data could be found on its properties therefore aluminum 5086 sheet was chosen as replacement.
- **Centaur Oxidizer tank:** Reported to use half-hard 301 stainless steel. This material is very similar to the full-hard equivalent with a slightly lower yield stress.

v. Tank Geometry

Tank cap geometry was simulated to be the same to the actual shape whenever possible. The only exception where the true shape could not be fully simulated was the top cap of the Centaur hydrogen tank. This shape was assumed to be spherical in the model, the real shape deviated from this slightly.

Atlas fuel tank: Conical bottom and spherical top

Atlas oxidizer tank: Inverted spherical bottom and spherical top

Centaur fuel tank: Inverted Ellipsoid bottom ($a/b = \sqrt{2}$) and spherical top

Centaur oxidizer tank: Ellipsoid bottom and Ellipsoid top (both with $a/b = \sqrt{2}$)

The Atlas first stage has the fuel tank below the oxidizer tank and the Centaur has its oxidizer tank below its fuel tank. Both stages have integrated tank configurations as can be seen from the inverted caps in both stages. All tanks are load carrying tanks and thus cylindrical parts of tanks have only a single wall. Trust frames for both stages where determined to be attached to the tanks and not the hull, the same applies to the payload frame.

vi. Propellant and pressurizer

The RL10A-3 engines used by the Centaur use liquid hydrogen and liquid oxygen as their main propellant. The engines in the Atlas stage both use liquid oxygen and kerosene (RP-1) as their main propellants. The pressurizer used by both stages is helium gas stored in spherical tanks. Helium as a gas only needed two parameters to function in the model which is heat capacity ratio γ set at a value of 1.66. The other value was molar mass $\mathscr{M}$ set to a value of $0.004[kg/mol]$. All propellant and pressurizer data was taken from the NIST database and can be seen in table 7.4.

Table 7.5: *Key variables for the small modern launcher configuration.*

Small modern launcher	Stage 1	Stage 2	Stage 3	
stage diameter	1.4	1.4	1.4	$[m]$
LOX tank pressure	4.0	30.0	30.0	$[bar]$
fuel tank pressure	4.0	30.0	30.0	$[bar]$
LOX temperature	97.0	97.0	97	$[K]$
fuel temperature	290	290	290	$[K]$
Load carrying LOX tank	false	false	false	
Load carrying fuel tank	true	false	false	
Common Bulkhead	false	false	false	
LOX tank above fuel tank	true	true	true	
Pressure tank location	between tanks	between tanks	between tanks	
Pressure tank pressure	200	200	200	$[bar]$

III. Small Modern Launcher

A liquid configuration for a small modern launcher was simulated as a three stage launcher using kerosene and liquid oxygen as propellant for all stages. Data from this simulation was obtained internally at the NLR. No further reference to data obtained is thus mentioned, data taken from elsewhere is naturally referenced.

i. Key Variables

The key variables chosen for the Small modern launcher are variables directly contributing to its build and are based upon typical values for a small launcher. These determining variables are described in table 7.5. Figure 8.1 shows the launcher as plotted by the model. Similar to the Atlas Centaur are all variables stated in the table are important factors in determining the structural dry mass of the launch vehicle.

ii. Material

Materials used in the small modern liquid launcher consist of orthotropic as well as isotropic materials. Carbon fiber reinforced plastic is mainly used for the outside structure of the launcher but also many tanks use this material to contain their propellant. An aluminum alloy is also used as a structural material, mainly for the liquid oxygen tanks. Material data can be found in table 7.6.

Since only material data for uni-directional CFRP was available the material has to have a lay-up pattern as input. Lay-up of all CFRP laminates where determined to be $[60/-60/0]^s$. This is a thin symmetric balanced quasi-isotropic lay-up and was deemed appropriate for the structure.

iii. Engines

Engines selected for the project are unitary engines. These RP-1/LOX engines provide relative little thrust but are used for all stages. The third and last stage only uses one of these engines, the second stage uses 6. The first stage uses 36 of these engines assembled in a large manifold to construct an aerospike nozzle. This engine of the first stage was therefore treated as if it were one engine.

Table 7.6: *Material data used for simulation of the small modern liquid LV.*

Name	CFRP Uni-Directional tape	Aluminium 5086 h36	Aluminum Honeycomb
$\rho[kg/m^3]$	1607	2657	32
$\sigma_1[MPa]$	1951	225	10
$\sigma_2[MPa]$	22.6	225	10
$E_1[GPa]$	119	70.3	0.72
$E_2[GPa]$	7.10	70.3	0.72
μ	0.31	0.33	0.3

Table 7.7: *Engine data used for simulation of the small modern liquid LV.*

Name	1^{st} stage engine	2^{nd} stage engine	3^{rd} stage engine
Manufacturer	-	-	-
Thrust$[kN]$	277.8	46.1	7.7
Calculated mass$[kg]$ 3	309.2	217.2	36.2
Diameter$[m]$	1.4	0.35	0.35
Length$[m]$	1.5	1.2	1.2
$0/F$	2.23	2.23	2.23

2 - Mass is calculated using Zandbergen M-S1 relation

table 7.7 shows the data used as input for the engines. Thrust shown is the magnitude of engine thrust in vacuum.

iv. Material Allocation

The outside of the entire rocket is made out of a $[60/-60/0]^s$ lay-up sandwich panel this means that all load-carrying tanks, i.e. the fuel tanks also use this wall as their cylindrical wall. The end-caps of the fuel tanks are made for a non-sandwich construction with CFRP. Thrust-frames and payload-frames were also made with the CFRP sandwich wall.

The non-carrying LOX tanks are all fully made of aluminum. This means the LOX tanks have a double wall at the cylindrical part where the inside in the aluminum tanks wall and the outside is the earlier mentioned sandwich panel.

v. Tank Geometry

Tank geometry implemented was quite simple, predetermined propellant masses resulted in the following tank shapes.

Stage 1 fuel tank: cylindrical tank with spherical caps.

Stage 1 oxidizer tank: cylindrical tank with spherical caps.

Stage 2 fuel tank: spherical tank.

Stage 2 oxidizer tank: spherical tank.

Stage 3 fuel tank: spherical tank.

Stage 3 oxidizer tank: spherical tank.

vi. Propellant and Pressurizer

Propellant for all three stages is RP-1 with liquid oxygen. The data for these propellants can be found in table 7.4. To pressurize the fuel and oxidizer tanks gaseous Helium was chosen to be fitted in small spherical pressure tanks between the two tanks. The volume of these pressure tanks was calculated with the volume of the to be pressurized tanks and an initial tank pressure of $200[bar]$. The number of pressure tanks for each stage was determined by increasing the number from an initial two tanks until they fitted between the fuel and oxidizer tanks.

Chapter 8

Verification & Validation

The report divides verification of the model up into two parts; internal and external verification. Internal verification will focus on verifying functions within the model, external verification will focus on verification of results.

I. Buckling Functions

Various buckling function were used in order to calculate the correct thickness required to withstand each proposed load case. Verifying that these functions within the model, output correct values is essential for the calculation of structure mass. Several cases where verified using a simple FEM analysis with similar input parameters. The FEM analysis where executed using ABAQUS software.

Buckling load criterion was calculated for a simple isotropic cylinder and compared to FEM data. This comparison can be seen in table 8.1. The table shows good comparison between theoretical buckling values ($\gamma = 1$) and obtained FEM data with values within 0.1% of each other. The other values show a significant decrease in strength when imperfections are accounted for ($\gamma \neq 1$). This value becomes significantly lower when radius over thickness ratio increases, i.e. thin walled cylinder. It can also be observed from the table that the function used for orthotropic materials is conservative compared its isotropic counterpart with a value that is 17% lower.

The correlation factor; γ, calculated for the isotropic and orthotropic cylinder have values of 0.32 and 0.27 respectively. This means

Table 8.1: *Comparison of obtained critical buckling loads for an isotropic aluminum cylinder under axial compression.*

Input $[m]$		Results $[kN]$	
t	0.002	Isotropic($\gamma = 1$)	172.0
r	1	Isotropic	55.3
l	2	Orthotropic	45.9
		FEM	171.8

Table 8.2: *Comparison of obtained critical buckling loads for an carbon fiber laminate with (0/60/-60) symmetrical lay-up, cylinder under axial compression.*

Input [m]		Results [kN]	
t	0.002	Orthotropic	49.1
r	1	FEM Lay-up	70.0
l	2		

II. Build Process

Since the model "builds" a launch vehicle form the ground up the build process needed to be verified. The model builds a launch vehicle by creating various coordinates for every part inside a stage. These various parts are than given a place within the rocket stage, finally this stage is put in its correct place in the launch vehicle. To ensure that each component has its correct position a plotting tool was developed. This plotting tool plots the rocket in 3 dimensions where a quick assessment can be made to determine if the launch vehicle is build correctly. An example of a plot can be seen in figure 8.1.

Figure 8.1 shows a wire-frame plot of a entire launch vehicle with fuel, oxidizer and pressurization tanks drawn to scale. engines are represented as simple cylinders with a diameter and height. inter-stage structures are plotted between stages and it can be verified from the plot that these lengths are correct and that no object collides with another. Payload is represented as a cylinder inside the fairing. The model supports more fairing designs but only plots conical shapes. Other objects plotted are thrust frames, payload frames.

Other than object location more things can be checked using this plotting tool. An option was implemented to draw the location of the center of mass, this can also be seen in figure 8.1. It can't be directly verified that it is in the absolute exact location but an engineer can use this method to access if its calculation is in the right ballpark.

A similar plotting function was developed to check propellant level at a certain percentage of maximum volume. Figure 8.1 shows full propellant levels for the third and second stage and approximately half full tanks for the first stage, the levels are indicated with a red circle. Propellant levels suffer from the same drawback as the center of mass plot, when it comes to verification through plotting. This means the calculation cannot be verified directly through the plotting tool. However, one can be certain propellant levels do not exceed the limits of the tank and one can determine if calculated propellant levels are approximately in the correct position since corresponding propellant volume fraction is known.

III. Mass distribution

Mass distribution plots can show mass varies in the launch vehicle and if this looks correctly according to input given. This function is verified using the small modern launcher as input and using the `MassPerMeter(z)` function plotted versus z coordinates. Figure 8.2 shows the plot for three different propellant levels to verify its workings. The figure shows a clear mass distribution where the first stage distribution differs from full to empty. The half full tank clearly displays mass at the bottom of the tank and no mass at the top half. This is of course expected behavior since the propellant is drained from top to bottom.

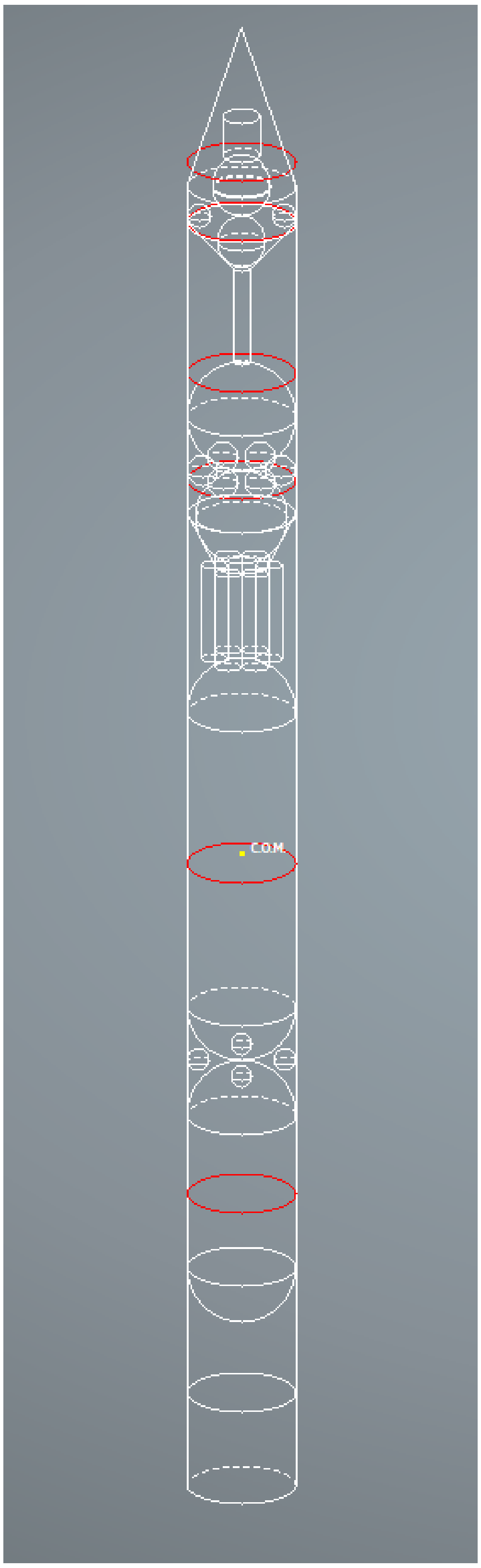


Figure 8.1: *Example of the small modern launcher vehicle plot.*

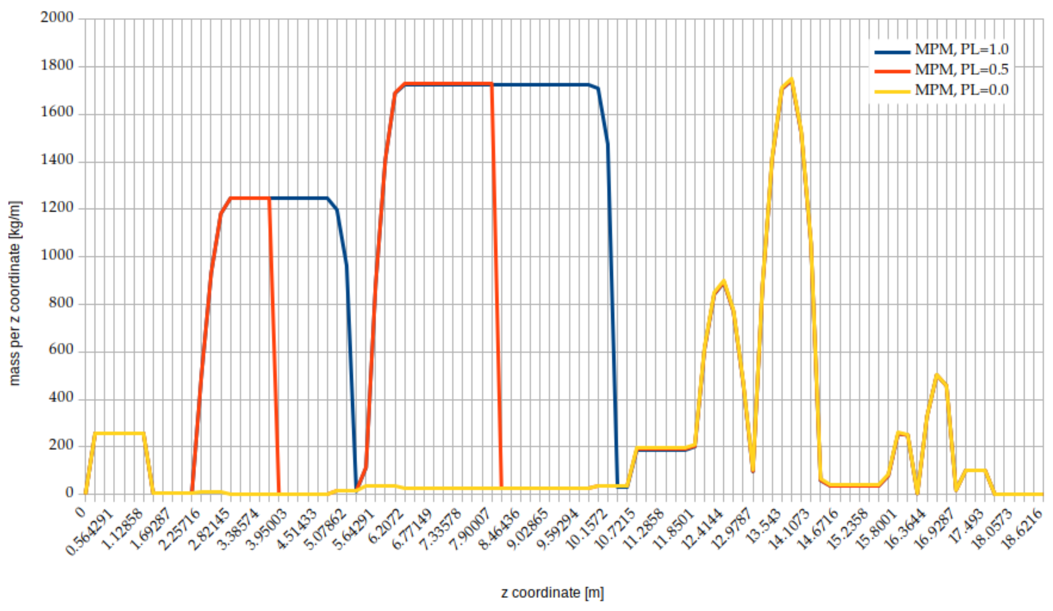


Figure 8.2: *Mass distribution of the small modern launch vehicle.*

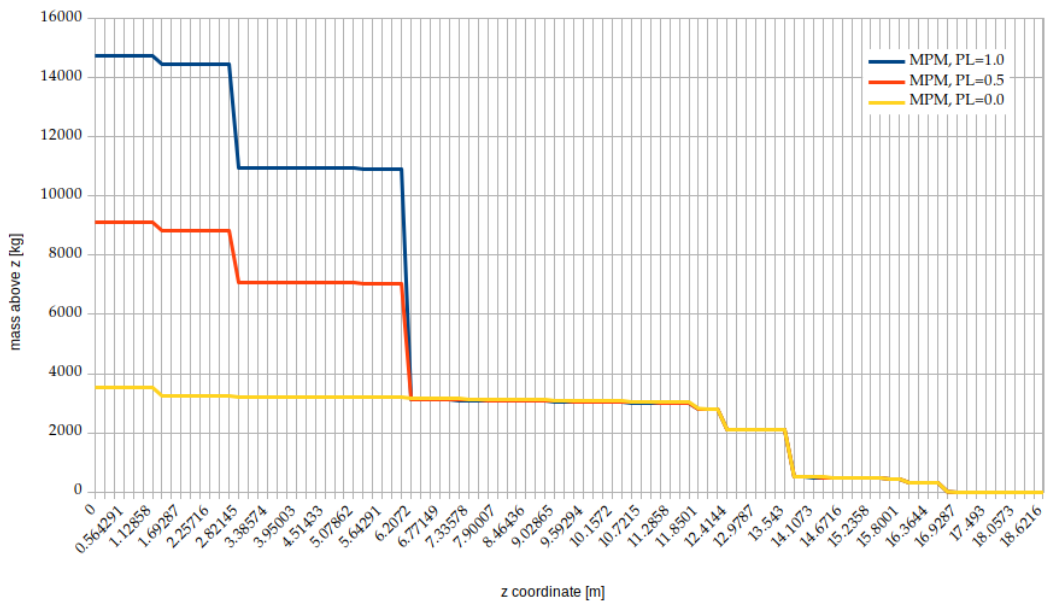


Figure 8.3: *"Mass above z" of the small modern launch vehicle.*

IV. Axial Load

Verification of the working of the axial load function is necessary eventhough it is very similar to the mass distribution. It is modeled differently however. The mass distribution used for computing the axial load only looks at mass above the structure an mass distribution of the structure itself. To clarify what is meant by this an example is given.

The axial load acting on a z coordinate on a cylindrical wall of a propellant tank is proportional to the mass of the subsystems above it and the section of dry mass of the tank above the z coordinate only. This means that propellant inside of the aforementioned tank is assumed to not affect the axial load on the cylindrical wall. The weight of the propellant was modeled to be carried by the bottom of the tank and not the cylinder.

This thus required a different function which is plotted is figure 8.3. Another method used to verify the working of this function was to compare the vale of the total mass calculated by the function at $z = 0$ to the total mass calculated by the model. These values matched with slight error due to numerical integration by the axial load function.

V. Moment

Moment calculation is verified by inducing a load-case with a thrust vector and analyzing the moment throughout the structure this is plotted in figure 8.4. This figure shows a smooth solution to the moment experienced by the structure. Numerical integration was done from either side starting with an moment of zero at the ends. The lateral thrust vector is induced at $z = 0$ and shows a rapid increase of moment as it approached the center of mass after which moment decreases due to mass being accelerated.

Numerical integration from both ends met at the center of mass. It can be observed that both integrations have the same value and slope at the center of mass which indicates that the moment is correctly computed.

VI. External Verification

External verification is done by comparing calculated values of the Atlas-Centaur rocket to reported values for which data is present. Results of this verification can be seen in table 8.4 and table 8.3. Similarities can be observed through with calculated thicknesses approximating reported ones. This results in fairly low mass error in the dry mass calculations for the Atlas stage but a large error in computation of the Centaur dry mass.

Structural mass of the Centaur stage is fairly well approximated compared to dry mass. This magnitude of this gap can be attributed to the large difference in fairing mass and the Centaur having a rather large mass dedication to avionics ($\sim 650[kg]$) and a relative heavy mass value for its insulation panels ($\sim 555[kg]$).

The method of calculating fairing thickness in the model was not observed in literature by the author. This might be a cause for the large reported error. The reliability of this method is thus undetermined until it can be validated through CFD and FEM models of fairings. This is however outside the scope of this project.

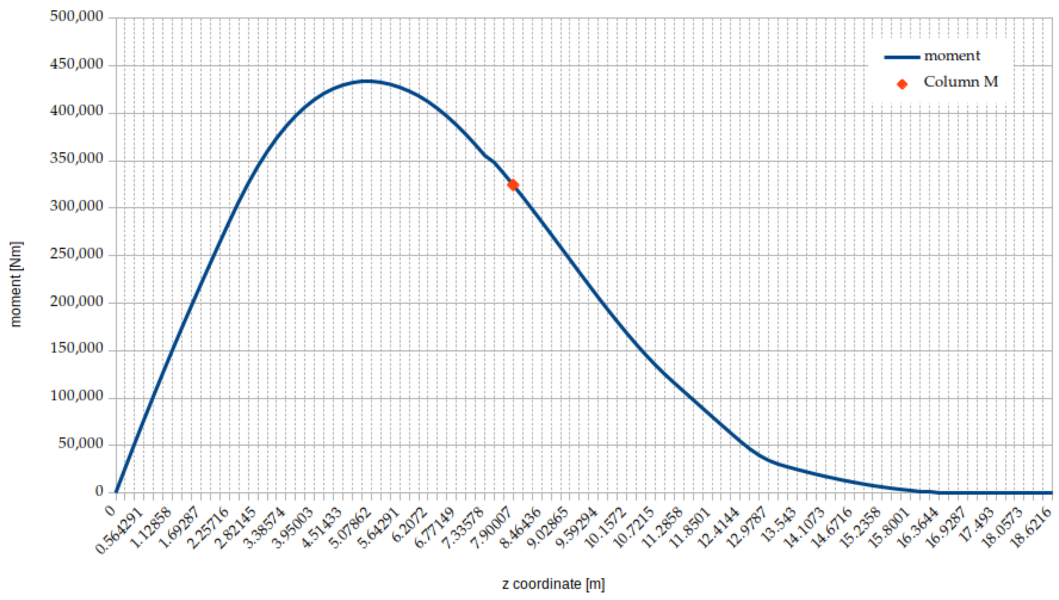


Figure 8.4: *moment experienced by structure of the small modern launch vehicle at* 2[*deg*] *thrust vectoring.*

VII. Internal Verification

Internal verification is this report was defined as checking if the model has met it's requirements mentioned in the first chapter. The clarify the requirements are again listed.

1. Computation time shall be tune-able versus computation accuracy.
2. TUDAT compatibility
3. TUDAT independent
4. structural mass shall be calculated within $\pm 10\%$ of actual values.

The first three requirements are easily verified the first requirement can be verified by the general settings file which can be seen in chapter 7. A decision can be made to increase or decrease the precision of the numerical integration functions of the model. These functions are the most computation intensive functions in the model and were thus deemed to be able to make the model well tune-able for the user.

The second and third requirement are easily verified since the model is completely build within TUDAT but can be run stand alone. Since the model does not depend on external TUDAT functions.

The last requirement is difficult to verify, since limited data is available. The dry mass of the simulated Atlas stage was within 8.63% of the real life value which complies not with the requirement since it applies to structure mass and not dry mass. It gives however a good indication of accuracy.

Table 8.3: *Comparison of real and calculated values - Centaur*

Parameter	Literature	Calculated Value	Error [%]
Total dry mass [kg]	3150	1361.5	56.8
Total structure mass [kg]	427	500.1	17.1
Total Height [m]	14.63	13.48	7.86
Fuel tank shell thickness[mm]			
Top	0.25 - 0.41	0.15	[-]
Cylinder	0.36	0.30	[-]
Bottom	0.33 - 0.66	0.68	[-]
Oxidizer tank shell thickness[mm]			
Top	0.33 - 0.66	0.68	[-]
Bottom	0.46 - 0.51	0.68	[-]
Fairing mass[kg]	940	311.7	66.8

Table 8.4: *Comparison of real and calculated values - Atlas*

Parameter	Literature	Calculated Value	Error [%]
Total dry mass [kg]	6109	5581	8.63
Total structure mass [kg]	-	1378	
Total Height [m]	21.0	23.0	8.69
Fuel tank shell thickness[mm]			
Top	0.61	0.47	[-]
Cylinder	0.97	0.95	[-]
Bottom	1.04	1.16	[-]
Oxidizer tank shell thickness[mm]			
Top	0.41	0.63	[-]
Cylinder	0.71	0.54	[-]
Bottom	0.61	0.47	[-]
Inter-stage mass[kg]	468	110.37	76.4

The centaur simulation give a structural mass error of 17.1%. This is outside the set requirement formulated at the start of the project. This means this requirement is not met since the error is to large. A hypothesis was made that the error can attributed to simplistic modeling of the thrust and payload-frame. especially the thrust-frame was calculated to be relatively heavy. This cannot be verified since this data is not available. A different configuration of the thrust-frame with rings or a sandwich panel could greatly reduce its thickness and thus mass.

VIII. Validation

Validation of the model happens by comparing the results to ZandBergens equations for mass modeling of stages mentioned in chapter 1. Table 8.5 shows a comparison between the reports model and zandbergens model. It can be observed that the results from both models is quite different. There is no situation where both models approach the same value.

For the small modern launcher (SML in the table) this difference can be explained to a certain extend. Zandbergen's model seems to not favor small and light launchers since it

Table 8.5: *Comparison to Zandbergens model on stage mass*

	Construction mass		Dry Mass		
	Model	Zandbergen	Model	Zandbergen	Reported
Atlas	1378	4807	5581	9748	6109
Centaur	500	2511	1361	2863	3150
AC-LV	1878	7318	6942	12612	9259
SML S1	161	2337	478	2251	
SML S2	106	2138	318	1646	
SML S3	21	2098	58	1523	
SML-LV	288	6574	854	5421	

makes use of a constant in its equations. This means that even the smallest launcher start with a preset value. This value disappears becomes less significant when stage propellant mass increases. This means small stages inadvertently have construction and dry masses above $\sim 1000[kg]$.

When comparing results from the Atlas-Centaur simulation the difference in the models can be better analyzed. Since reported values of its mass are available and the simulated LV is larger the comparison becomes more clear. An interesting observation can be made where the model predicts the mass of the Atlas stage quite well compared to Zandbergen's model. This is vice-versa for the Centaur stage, where Zandbergen's model provides more accurate predictions.

Error in the simulation of the Centaur might be caused by the way fairing thickness is calculated by the model. The fairing is reported to be fairly heavy compared to its approximated mass, 940 vs 311 [kg]. It is difficult to determine how large of a portion of fairing mass cannot be attributed to structural parts, e.g. separation systems. No data was found on the actual thickness of the fairing which make analysis difficult. The Centaur stage also had a unique insulation system for the liquid hydrogen which was quite heavy and unconventional. Whichever is the case the fact remains that fairing mass calculation in the model is an approximation which has not been verified.

Chapter 9

Results

I. Atlas-Centaur

Atlas-Centaur results are described in this section. The mass and required thickness of structural parts is shown and a simulation of the first stage's flight.

i. Mass & thickness

Mass and thickness results for the Atlas-Centaur simulation are displayed in table 9.1. Mass for the propellant tank part is combined since it is considered one subsystem, this is also the case for the fairing. Thickness of the structural parts is determined from the worst-case scenario during flight.

The first 135 seconds of flight of the launch vehicle is simulated and required thickness was determined. In order to determine load case the propellant left at a certain time in flight has to be determined. This was done by determining the propellant mass flow of the first stage propulsion system. Thrust and specific impulse is used to determine the mass flow rate. This is turn calculates how much mass is lost during flight time t, which is set at 153 seconds, this is the time where the LV experiences a maximum load case of 5.7.

- **Booster thrust:** $747[kN]$
- **Sustainer thrust:** $258[kN]$

- **Booster specific impulse:** $282[s]$
- **Sustainer specific impulse:** $248[s]$

$$\dot{m} = \frac{F_T}{I_{sp} g_0} \tag{9.1}$$

mass flow then becomes:

- **Booster mass flow:** $270[kg/s]$
- **Sustainer mass flow:** $106[kg/s]$
- **Total mass flow:** $646[kg/s]$

- **Booster mass flow:** $307[kg/s]$
- **Sustainer mass flow:** $93[kg/s]$
- **Total mass flow:** $646[kg/s]$

Table 9.1: *Mass and thickness of structural parts of the Atlas-Centaur simulation*

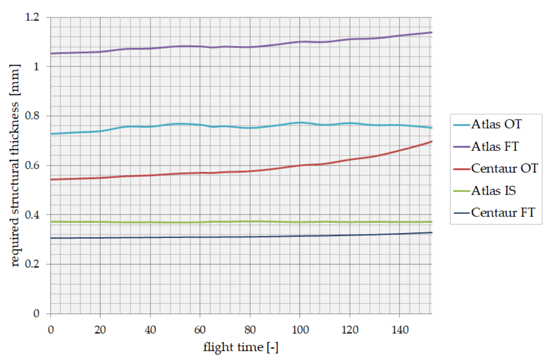
	stage 1		stage 2	
	mass[kg]	thickness[mm]	mass[kg]	thickness[mm]
Fuel tank top		0.474093		0.15013
Fuel tank cylinder	487.448	0.948187	160.657	0.300259
Fuel tank below		1.16129		0.405905
Oxidizer tank top		0.634981		0.68363
Oxidizer tank cylinder	541.14	0.537306	128.691	0.505699
Oxidizer tank below		0.268653		0.68363
Thrust frame	102.513	3.62756	179.61	3.92148
Payload frame			11.0355	0.240942
Fairing cylinder			311.668	4.30375
Fairing tip				2.48408
Pressure tank	17.0528		9.89508	
Interstage	110.37	0.313654		
Total structure	1258.5238		801.55658	

The findings of this simulation can be found in figure 9.1a. The figure plots required thickness of various structural parts versus the flight time. Thickness was chosen to be plotted instead of stress since it is more intuitive for illustration. This has to do with the fact that the critical mode of failure can switch between tensile and buckling during flight. The abbreviations in the plot, OT and FT stand for oxidizer and fuel tank respectively.

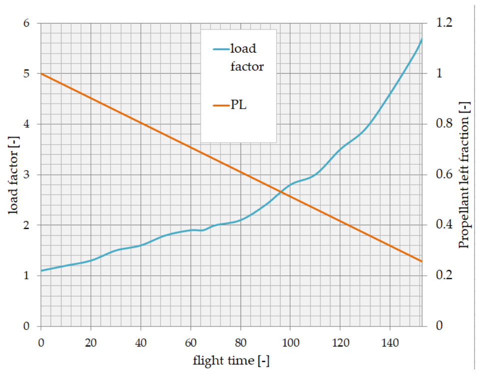
The figure shows rather constant lines of required thickness. Change in pressure due to hydro-static effects increases with increasing with load-factor. It decreases with decreasing propellant level. During the first 153 seconds of flight the propellant level in the second stage, the Centaur does not decrease. This can be observed from the plot since required thickness increases due to an increase in acceleration (load factor). This effect is especially visible in the Centaur's oxidizer tank, since the LOX is much heavier than the liquid hydrogen in the fuel tank.

ii. Center of mass & moment of inertia

Center of mass and moment of inertia were determined for a first stage with an empty tank to a first stage with a full tank. The results of this simulation can be seen in figure 9.2a and 9.2b. If constant mass flow is assumed than it can be observed from figure 9.2a that center of mass does not shift linearly during flight. During the first 70 percent of flight time the center of mass stays nearly at a constant position where a large shift happens during the last 30 percent of flight. Mass moment of inertia changes rather linearly with a value of 40 percent of its initial value during depletion. This translates to a factor of approximately 2.5 increase in angular acceleration with a similar moment applied between lift-off and burn-out.

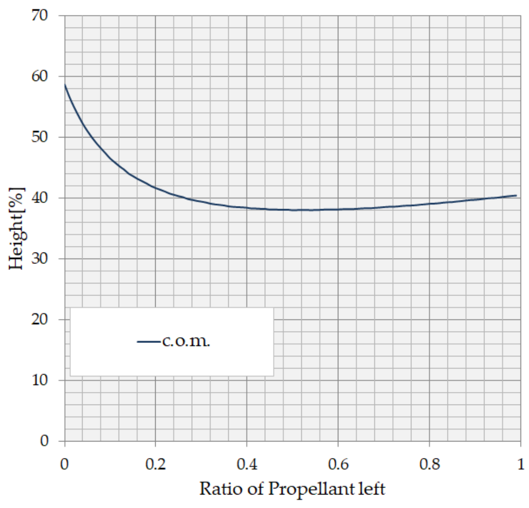


(a) *required thickness of structure*

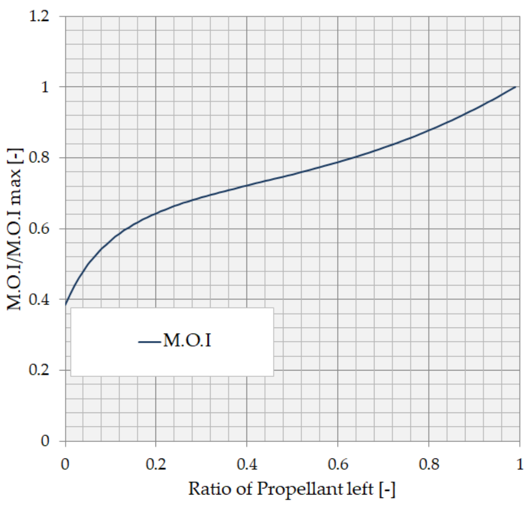


(b) *load factor and propellant left*

Figure 9.1: *Atlas Centaur simulation during the first 153 seconds of flight*



(a) *Center of mass drift during first stage burn, expressed in percentage of total height.*



(b) *Mass moment of inertia change during first stage burn, expressed as fraction of maximum value calculated.*

Figure 9.2: *Atlas Centaur simulation of center of mass and mass moment of inertia.*

II. Small Modern Launcher

The small modern launcher was simulated and results for this simulation are shown in this section. Similar to the Atlas-Centaur simulation the mass and thicknesses are shown as well as a simulation during flight. The first stage of the launcher is assumed to have a constant mass flow of:

$$\dot{m} = 89.35[kg/s] \tag{9.2}$$

The flight time t_{flight} is assumed to be 121 seconds.

This means that total propellant burned is:

$$M_{p,burned} = \dot{m}t_{flight} = 10.8 \cdot 10^3[kg] \tag{9.3}$$

Total propellant for this stage was determined to be:

$$M_p = 11.6 \cdot 10^3[kg] \tag{9.4}$$

Propellant left at the and of the flight is thus.

$$(PL) = 1 - \frac{10.8}{11.6} = 6.9 \cdot 10^{-2} \tag{9.5}$$

i. Mass & thickness

Required thickness for structural subsystems during the first 121 seconds of flight can be observed in figure 9.3a to 9.3d. This shows that fairing thickness structural thickness is dependent on the maximum dynamic pressure. Most not pressurized structural parts such as inter-stages and hulls are affected by load factor more than drag and thus sees it critical load at the end of the simulation.

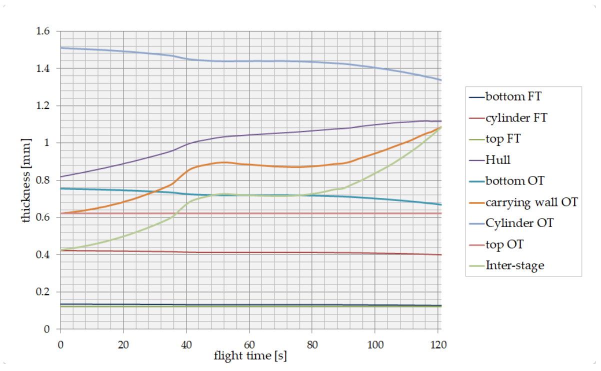
The simulation results are summarized in table 9.2, the table shows the required thickness of every subsystem to withstand every load-case during the trajectory. The corresponding mass of each subsystem is also displayed. A total construction mass for the small modern launcher was determined to be 287.72$[kg]$.

ii. Center of Mass and Moment of inertia

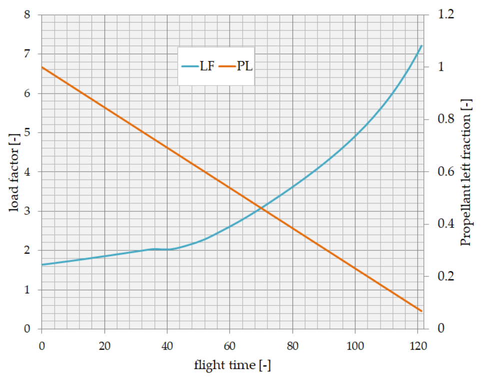
Moment of inertia and center of mass plots can be seen in figure 9.4. This plot shows the properties plotted versus propellant left in the first stage tank. It can be observed that center of mass shifts upwards during flight in a non-linear way. Assuming constant engine mass flow the center of mass stays somewhat constant during the firs half of flight time. Moment of inertia seems to decrease at an increasing pace during flight, this makes somewhat sense since weight is removed further away from the center of mass during the end of the flight, i.e. propellant is at the bottom of the tank.

A hypothesis on why center of mass is approximately at a constant distance during the first half of flight was formulated. It is thought this behavior stems from the fact that the center of mass initially is located somewhere between the propellant level of the two propellant tanks in the first stage. Removing propellant above and below the center of mass and thus retaining a somewhat similar center of mass is thought of as the reason of this behavior.

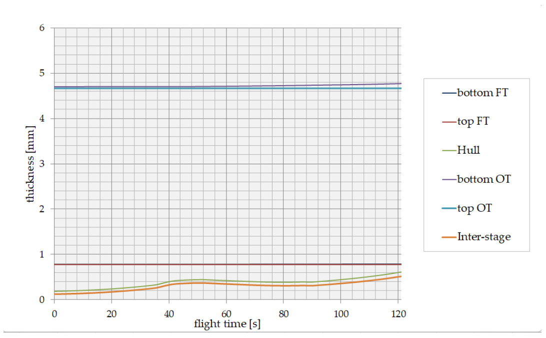
(a) *stage 1, required thickness of structure*



(b) *load factor and propellant left*



(c) *stage 2,required thickness of structure*



(d) *stage 3,required thickness of structure*

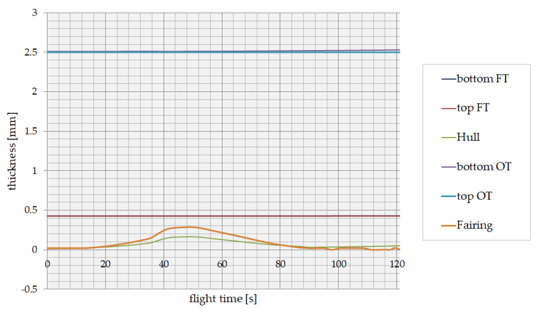


Figure 9.3: *Small modern launcher simulation during the first 121 seconds of flight*

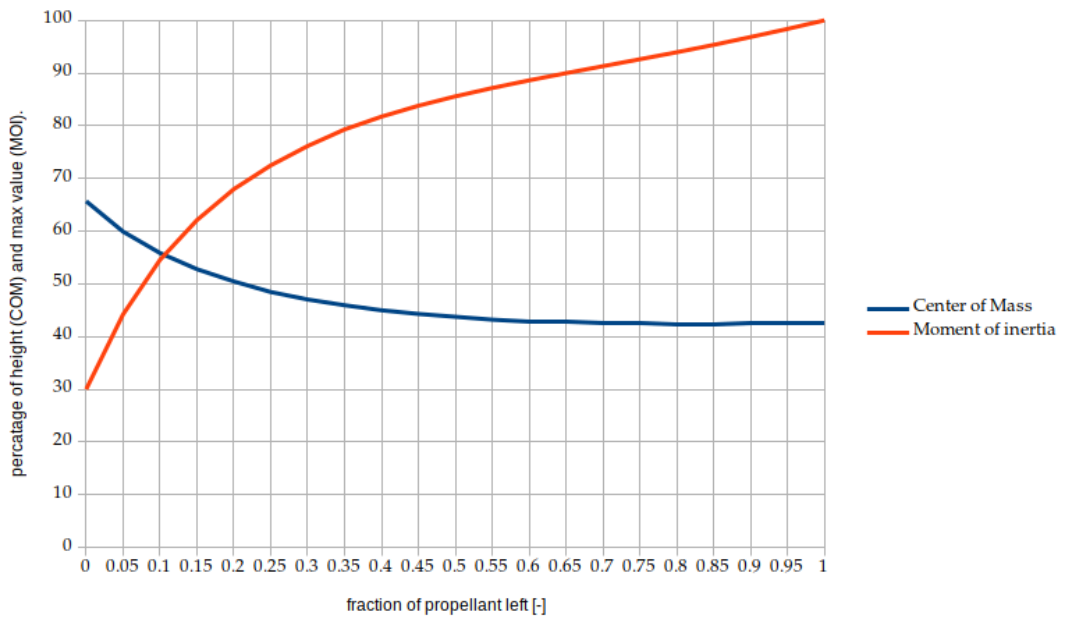


Figure 9.4: *Moment of inertia and center of mass plotted versus propellant left for the small modern launch vehicle.*

Table 9.2: *Mass and thickness of structural parts of the small modern launcher simulation*

	stage 1		stage 2		stage 3	
	mass[kg]	thickness[kg]	mass[kg]	thickness[kg]	mass[kg[	thickness
Fuel tank top		0.122699		0.775635		0.426599
Fuel tank cylinder	8.46388	0.422502	5.5746		0.917432	
Fuel tank below		0.134974		0.786445		0.429849
Oxidizer tank top		0.622222		4.66667		2.50001
Oxidizer tank cylinder	108.734	1.51059	75.7261		12.8057	
Oxidizer tank carrying wall		1.08656				
Oxidizer tank below		0.755296		4.77083		2.53103
Thrust frame	9.06641	0.60259	1.52842	0.154329	1.00409	0.0958104
Payload frame					0.722545	0.236727
Fairing					3.63879	0.286797
Pressure tank						
Hull	12.7538	1.11769	9.13478	0.613921	1.82161	0.165398
Interstage	22.1719	1.08565	13.8023	0.51699		
Total structure	161.05		105.76		20.91	

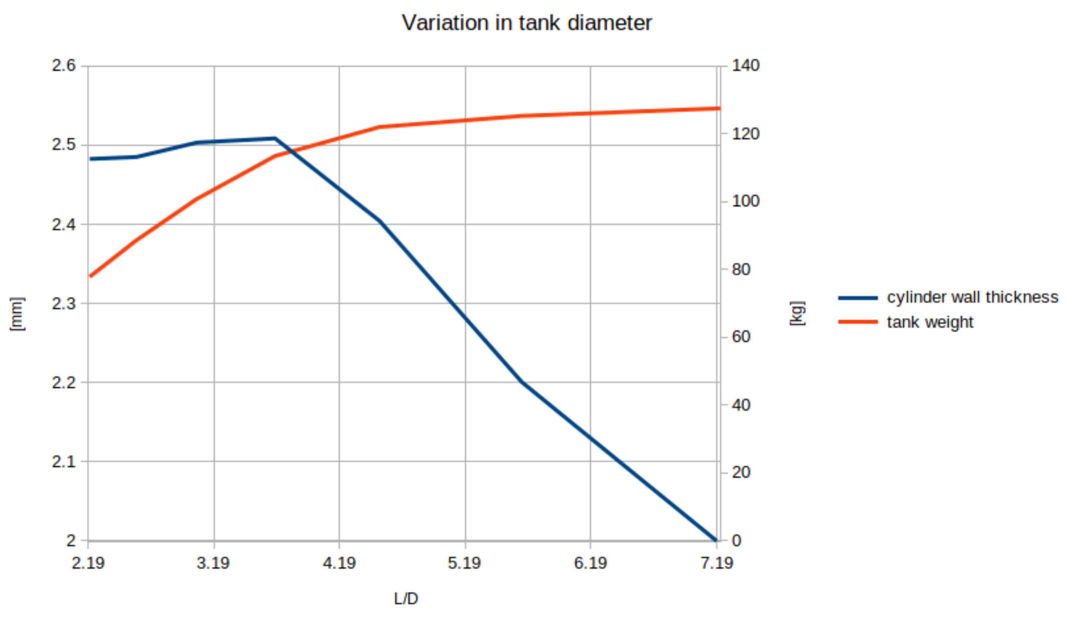


Figure 9.5: *Change of tank diameter vs tank weight and cylinder wall thickness of a low pressure tank with buckling failure.*

III. Changing variables

This section shows different kinds of graphs where the effect of changing variables is shown and analyzed.

i. Tank diameter variation of a low pressure tank

Variation of the diameter of a cylindrical tank with spherical caps is shown in figure 9.5. The horizontal axis shows the length over diameter value of the tank. The left side of the graph thus indicates a tank with a large diameter while the right side represent thinner and longer tanks. The left vertical axis shows the thickness of the cylindrical wall of the tank and the right vertical axis shows the weight of the tank in $[kg]$.

The tank in figure 9.5 is calculated through with a relative low pressure. This means that the thickness of the material is based upon buckling failure instead of failure due to high internal pressure. The tank experiences a flight load similar to the tanks of the Atlas-Centaur and the small modern launcher is the previous sections.

The mass of the tank decreases by enlarging the diameter of the propellant tank. This is expected since the tank become more "spherical" because is length also decreases. The tank can thus encompass the same volume with a smaller surface area.

The cylindrical wall thickness of the tank seems to increase with an increasing diameter. This might seem counter-intuitive since the length of the tank gets smaller but due to an increase in circumference the required thickness due to buckling increases.

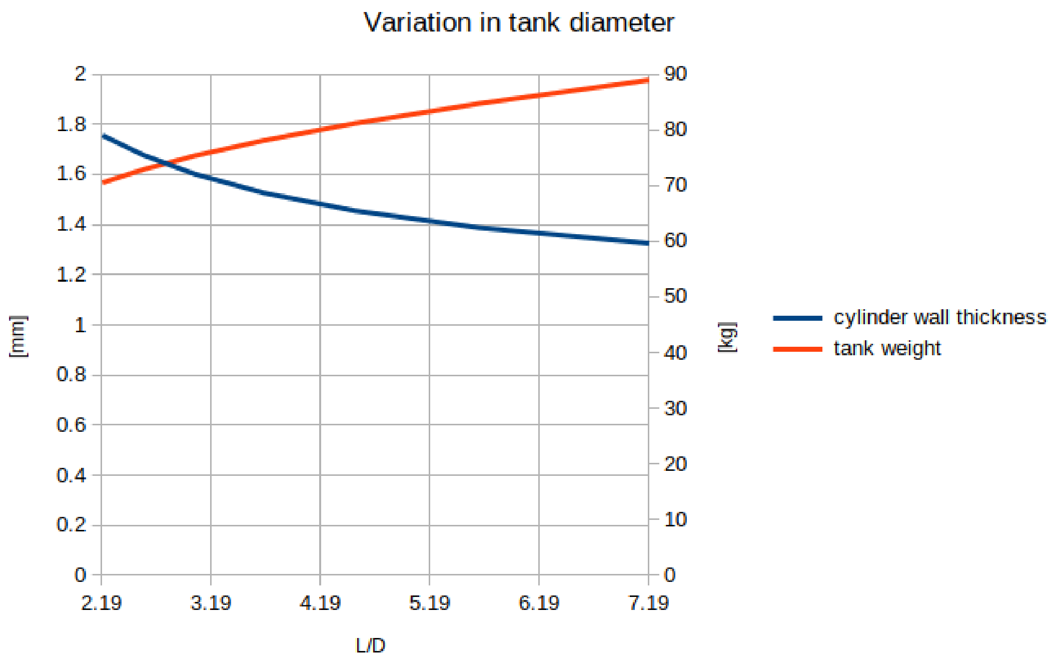


Figure 9.6: *Change of tank diameter vs tank weight and cylinder wall thickness of a high pressure tank with tensile failure.*

ii. Tank diameter variation of a high pressure tank

The same calculations were done on a high pressure tank with tensile failure due to the high internal pressure inside. The results of this calculation can be seen in figure 9.6. The graph is similar to the one in figure 9.5 with the same variable on each axis.

Similar to the low pressure tank a decrease in weight can be observed when the diameter increases. In this case the decrease is much more linear and less steep with the plotted length over diameter parameter. An increase in cylindrical wall thickness is expected with an increase in diameter. Since required thickness of a cylinder under pressure scales linearly with diameter (see appendix).

The interesting observation that can be made from both figures is the lower mass and thickness of a tank under high pressure compared to a tank under low pressure. This is caused by the internal pressure of the tank counteracting the compression force that causes buckling failure. This results in a high pressure tank with a smaller required wall thickness than a tank has a relatively low internal pressure.

iii. Varying tank pressure

Figure 9.7 shows a tank with a constant diameter but changing tank pressure. This is done to illustrate the transition between buckling and internal pressure failing. Previous figures illustrated the effect of changing diameter on low and high pressure tanks. Where low pressure tanks needed a thicker wall. An increase in pressure however also requires a thicker wall due to greater tensile strain on the material. This indicates that there is a minimum required thickness for a certain tank pressure. This is what is shown in figure 9.7.

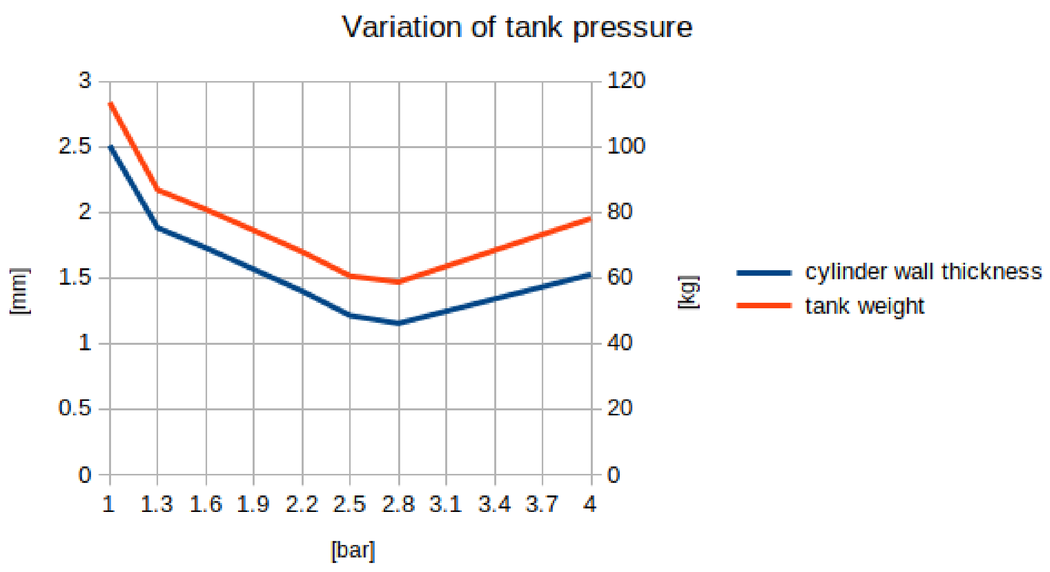


Figure 9.7: *Variation of tank pressure plotted versus tank weight and cylindrical wall thickness.*

At roughly an internal tank pressure of $2.8[bar]$ the wall thickness is at its minimum. This is obviously not a general rule and this point will vary for different launch vehicle configurations. After this point internal pressure will be a deciding factor in determining the wall thickness of the propellant tank. Before this point buckling is the deciding factor in determining the cylindrical wall thickness. It is important to conclude that simple equations that relate cylindrical wall thickness to internal pressure only work for relatively high pressures. At lower tank pressures one cannot depend on these equations to give a representative wall thickness.

iv. Variation of tank a/b parameter

The variable a stands for the radius of the tank or the semi-major axis of an ellipsoid. The variable b stands for the semi-minor axis of the ellipsoid or the "height" of the tank cap. It is thus possible to change both parameters to change the shape of a propellant tank. A value for $a/b = 1.0$ indicates a spherical tank. A higher value indicates a slightly squeezed tank. Although spherical caps have an ideal shape to minimize thickness of the cap wall, it can be desirable to "squeeze" the tank flat to shorten the structure. The effect of changing a/b is illustrated in figure 9.8.

The figure has the variable a/b plotted on the horizontal axis with a value starting at one(spherical) to a more flattened tank cap. The vertical axis on the left indicates the percentage lost or gained compared to a spherical configuration. For this tank configuration only a small loss in tank length is achieved by increasing a/b. The sacrifice for this is a rather large increase in total tank mass. This seems to be the general behavior for a change in this parameter. Tank cap wall thickness also needs to be increased significantly when increasing the a/b parameter. As a general rule it was found not to let this parameter exceed the value of $a/b = sqrt(2)$.

v. Propellant storage temperature

Figure 9.9 shows the effect of propellant storage temperature on the structural weight and length of a liquid oxygen tank. Colder propellant has a higher density which leads to the

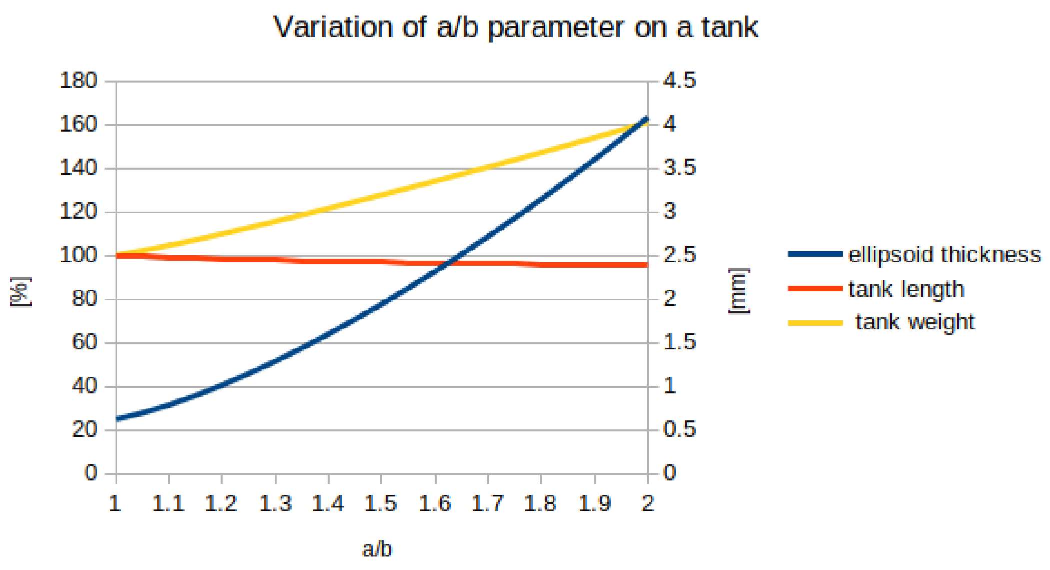


Figure 9.8: *Variation of tank a/b plotted versus tank weight and cylindrical wall thickness and tank length.*

possibility of constructing smaller propellant tanks. For the tank calculated a difference of 16.0% in mass and 13.7% in height can be achieved. This is quite a significant gain or loss in structural mass. As discussed earlier in this report the model does not take into account possible increase in insulation thickness. This effect is difficult to analyze since colder propellant would need thicker insulation but a smaller tank also encompasses less surface area. The magnitude of the effect of insulation is thus unknown.

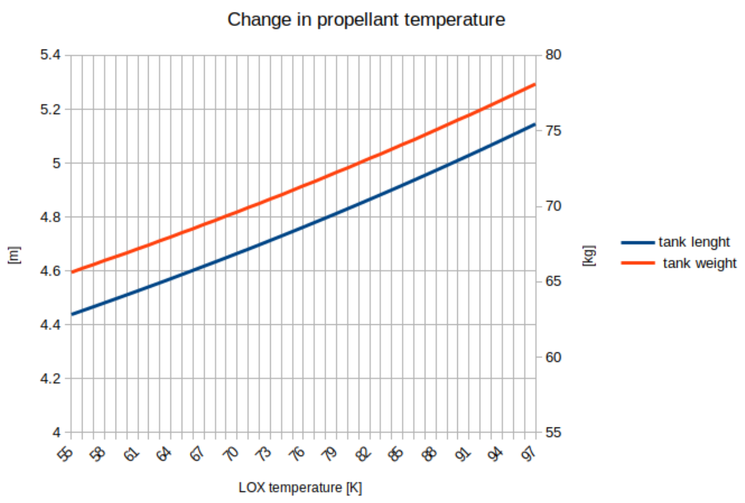


Figure 9.9: *Change in propellant storage temperature plotted versus tank weight and height.*

Chapter 10

Conclusions & Recommended Continuation

The final chapter of the body of the report describes conclusions and recommended continuation of the research. First a section reporting conclusions to the research. This section will describe lessons learned and interpretation of the results of the simulation. A second section will describes recommendations regarding continuation of the research.

I. Conclusions

Conclusions regarding the created model start with difficulties encountered before interpretation of the results is presented.

Fairing modeling One difficulty encountered was modeling of the fairing. As mentioned previously in the report the modeling of the fairing was not found in literature and is thus without validation. This means there was not a way to determine is validity without a relative large set of data about fairing thickness. The reason for modeling the fairing are obvious for a structural mass model but the error encountered from one validation (Centaur fairing)check was large.

The method used in the model assumes an equal pressure distribution over the surface of the fairing based on a drag force. Pressure is however not equally distributed over the fairings surface, but can vary greatly. Heat-flux into the fairing is also not taken into account which could be an important factor in structural design. These parameters and effects are outside the scope of the project. The conclusion thus remains that modeling of the fairing is difficult to validate.

Under valued dry mass Observing the results clearly indicates that the model under values the actual dry and construction mass of a rocket stage. These calculated values differ in error but the general trend is a lower calculated mass than the one observed from literature. This is interesting since calculated thickness values seems to comply rather well with ones found in literature. This indicates that the "missing" mass comes from structural parts not currently modeled.

The model assumes all structural mass as a "perfect" monocoque structure, this might result in lower expected values for structural mass. Perfect monocoque structure in this instance means a structure which parts flow over in each other fluently and is completely constructed from monocoque shells. This means connection structures, attachment points, etc. are not modeled. It is thus concluded that these "subsystems" likely compromise a large portion of the structural mass. Future research could perhaps model these by assuming a simple ring with bolted connection between parts or welds in metallic structures. This however creates an abundances of extra variables which need to be determined. Thus, The feasibility and usefulness of modeling these parts to determine structural mass more accurately will have to be accessed first.

Another reason for under approximating dry mass is not modeling insulation of the propellant tanks. Most notably, this resulted is a large error in the dry mass of the Centaur simulation which uses a relative heavy insulation panel system which is jettisoned during flight. This stage is also fully cryogenic using liquid hydrogen as fuel which could also result the large insulation mass due to large temperature differences and tank surface.

Critical load-cases Different load-cases for one LV were analyzed, since multiple load-case support was implemented in the model. Since even flight trajectories were analyzed the critical load-cases during flight could be analyzed. The analysis of the critical load case for each structural part was analyzed and a few conclusion can be drawn from it.

- Structural subsystems and parts yielding on buckling almost always have a critical load-case when the load factor (i.e. acceleration) is maximized.
- As exception; fairing thickness is determined by max dynamic pressure q,likely not the maximum load factor.
- Stress difference due to Hydro-static pressure it rather small. Higher load-factors mean the hydro-static pressure increases but higher load factors also appear with decreasing propellant levels which in turn decrease hydro-static pressure. They often almost cancel each other out.
- Max hydro-static pressure often occurs when propellant tanks are completely filled.

This results in a general sense in three recommended load cases for structural analysis of a launch vehicle:

1. Max dynamic pressure, q
2. Maximum load factor (often when propellant tanks are close to empty)
3. Full tanks.

This obviously does not paint the whole picture so to say for structural analysis. But this could be a good start if computation time is limited.

Center of Mass Both launch vehicles simulated showed a similar shift in center of mass when their tanks were drained. Its location stays somewhat constant at the beginning of the burn and shifts rather rapidly upwards after a significant portion of the propellant has been burned off. This means a linear shift in center of mass location in relation to propellant mass cannot be assumed.

Moment Determination of moment acting on the structure was not a trivial analysis. The resulting plots showed that when only thrust vectoring is applied as force the structure experiences a maximum moment between the bottom of the LV and its center of mass. This was somewhat expected since the mass of the LV starts to counter act the induced acceleration and at a certain point decreases its moment until it is zero at the top of the LV. Magnitude of moment was observed to vary greatly with location and information of this relation is necessary for proper analysis of the structural thickness.

Verification & Validation The research was somewhat stumped by lack of verification and validation data. This means that the validation of the model could have been more thoroughly if more data was available. Time constraint is also a factor in lack of this analysis. The model has grown quite large within the developed period, with a larger number of functions which needed to be tested and verified. This broad range of verification of functions and features was traded off against the depth of each analysis.

Usefulness of the research This research has given insight into structural design of a conventional launch vehicle. The research has analyzed critical factors for structural mass. From limited data the usefulness of mass determination could not be well validated. The model does seem to accurately predict necessary wall thickness under tension and buckling which is definitely a step in the right direction for a complete dry mass prediction model. Insight into distribution of axial load and moment acting on the structure was achieved through use of functions within the model. The moment acting on the LV was in particular non-trivial. For example, the small modern launcher simulation has a maximum moment applied to it between its engines and center of mass. The magnitude of this moment is crucial for determining required structural thickness and knowledge of its distribution is thus essential for a proper structural design.

A more practical sense of usefulness of the model is determination of center of mass, moment of inertia, axial load and moment throughout the structure. Solutions for these parameters were not trivial but have been achieved with success. For example, it was observed from detailed center of mass shifts that this shift is not linear as one might aspect between full and empty tanks. The relation between propellant left and center of mass location can take non-trivial shapes as can be seen in previous chapter. Another practical use of the model is the approximation of moment of inertia, this can be used to determine maximum angular acceleration of a hypothetical launch vehicle. This can be used in trajectory analysis as an upper bound boundary condition.

Concluding The research goal to develop an accurate and quick model for structural mass might be somewhat in the future still. Thickness of walls can be accurately predicted but structural mass seems to be still relative inaccurate. Further research seems necessary to implement structural parts which do not directly carry load but contribute in a different way, e.g attachment methods. Hopefully this research will have contributed to the research of approximating dry stage and launch vehicle mass.

II. Recommended Continuation

Further extension of the model might be seen in the form of adding various other subsystems and parts to the mix of subsystems. This might start by adding insulation if required thickness of said insulation can be determined. If so than implementation into the model would be without trouble since surface area of propellant tanks is already known. Further research into other structural parts a mentioned above might see improvement into approximation of dry mass.

Further validation of the model if data from other launch vehicles becomes accessible would be greatly increase the value of the model and the research which was conducted. It is thus recommended that this model is further validated in future research if possible.

A larger more complicated continuation of the model might be the implementation of the dynamic beam equation. which is shown below.[1]

$$\frac{\delta^2}{\delta z^2}\left(EI\frac{\delta^2 w}{\delta z^2}\right)^2 = -\mu\frac{\delta^2 w}{\delta t^2} + q(z) \tag{10.1}$$

The reason it is interesting to implement this equation into the model is that solving the equation for one can be applied for more accurate determination of moments due to bending during flight. The second reason is the determination of natural frequencies of the LV. The reason this equation could be implemented in this particular model is the existence of the function μ in the model which stand for mass per unit length. This already exist within the model as well as functions determining Young modulus E and moment of inertia I.

[1]Seon M Han, Haym Benaroya, and Timothy Wei. "Dynamics of transversely vibrating beams using four engineering theories". In: *Journal of sound and vibration* 225.5 (1999), pp. 935–988.

Bibliography

[1] John David Anderson Jr. *Fundamentals of aerodynamics*. Tata McGraw-Hill Education, 2010.

[2] Stuart W Churchill and Humbert HS Chu. "Correlating equations for laminar and turbulent free convection from a vertical plate". In: *International journal of heat and mass transfer* 18.11 (1975), pp. 1323–1329.

[3] Gary A Crowell Sr. "The descriptive geometry of nose cones". In: *URL: http://www. myweb. cableone. net/cjcrowell/NCEQN2. doc* (1996).

[4] Stephen R Davis. *C++ for Dummies*. John Wiley & Sons, 2009.

[5] Seon M Han, Haym Benaroya, and Timothy Wei. "Dynamics of transversely vibrating beams using four engineering theories". In: *Journal of sound and vibration* 225.5 (1999), pp. 935–988.

[6] A Hedayat, KC Knight, and RH Champion Jr. "Transient Analysis of X-34 Pressurization System". In: (1998).

[7] M Daniel Isaac and Ori Ishai. "Engineering mechanics of composite materials". In: *New York and Oxford* (1994).

[8] M. van Kesteren. "Air launch versus ground launch". MA thesis. TU Delft, 2013.

[9] Dietrich E Koelle. *Handbook of Cost Engineering for Space Transportation Systems (revision 1) with Transcost 7.1: Statistical-analytical Model for Cost Estimation and Economical Optimization of Launch Vehicles*. TransCostSystems, 2003.

[10] Lewis' staff. *Flight performance of Atlas-Centaur AC-13, AC-14, AC-15 in support of the surveyor lunar landing program*. Tech. rep. NASA, Lewis Research Center, 1974.

[11] J.H. Lienhard IV and J.H. Lienhard V. *A Heat Transfer Textbook*. 4th. Version 2.11. Cambridge, MA: Phlogiston Press, 2017. URL: `http://ahtt.mit.edu`.

[12] F. Miranda, B.T.C. Zandbergen, and D. Dirkx. "Design Optimization of Ground and Air-Launched Hybrid Rockets:" 2015. URL: `Item%20Resolution%20URL%20http://resolver.tudelft.nl/uuid:832aa36d-041b-4896-97a7-837f91cdc5d6`.

[13] Isaac Newton. "Principia mathematica". In: *Newton's principia* 634 (1934).

[14] Viatcheslav V Osipov et al. "Dynamical model of rocket propellant loading with liquid hydrogen". In: *Journal of Spacecraft and Rockets* 48.6 (2011), pp. 987–998.

[15] JP Peterson, P Seide, and VI Weingarten. "Buckling of thin-walled circular cylinders". In: (1968).

[16] AJP Van Kleef et al. "Innovative Small Launcher". In: (2015).

[17] P.J.M. Verschuren et al. *Designing a research project*. The Hague : Eleven International Publishing, 2010.

[18] George Z Voyiadjis and Peter I Kattan. *Mechanics of composite materials with MATLAB*. Springer Science & Business Media, 2005.

[19] VI Weingarten and P Seide. “Buckling of thin-walled truncated cones”. In: *NASA Space Vehicle Criteria (Structures), NASA SP-8019, Washington DC* (1968).

[20] Eric W. Weisstein. *Circle Packing*. 2017. URL: `http://mathworld.wolfram.com/CirclePacking.html`.

[21] Eric W. Weisstein. *Cone*. 2017. URL: `mathworld.wolfram.com/Cone.html`.

[22] Eric W. Weisstein. *Conical Frustum*. 2017. URL: `mathworld.wolfram.com/ConicalFrustum.html`.

[23] Eric W. Weisstein. *Geometric Centroid*. 2017. URL: `http://mathworld.wolfram.com/GeometricCentroid.html`.

[24] Eric W. Weisstein. *Surface of Revolution*. 2017. URL: `mathworld.wolfram.com/SurfaceofRevolution.html`.

[25] Roy Wildvank. “Master thesis proposal - Launch Vehicle Structure Mass Model”. In: (2017).

[26] W.A.R. Wildvank. “Reusing Rocket Stages - A Literature Survey”. MA thesis. TU Delft, 2017.

[27] Warren Clarence Young and Richard Gordon Budynas. *Roark’s formulas for stress and strain*. Vol. 7. McGraw-Hill New York, 2002.

[28] B.T.C. Zandbergen. *AE1222-II: Aerospace Design and Systems Engineering Elements I*. 2015.

[29] BTC Zandbergen. “Aerospace Design and Systems Engineering Elements I, Part Launcher Design and Sizing”. In: (2011).

[30] BTC Zandbergen. “Simple mass and size estimation relationships of pump fed rocket engines for launch vehicle conceptual design”. In: (2015).

Appendix A

Stresses in thin shell tanks

I. Internal pressure

i. Cylinder

$$without\ end\ caps : \sigma_z = 0 \tag{A.1}$$

$$with\ end\ caps : \sigma_z = \frac{p_T R}{2t} \tag{A.2}$$

$$\sigma_\phi = \frac{PR}{t} \tag{A.3}$$

ii. Sphere

$$\sigma_\varphi = \sigma_\phi = \frac{PR}{2t} \tag{A.4}$$

iii. Ellipsoid

$$\sigma_\varphi = \frac{p_T R_1}{2t} \tag{A.5}$$

$$\sigma_\phi = \frac{p_T R_1}{t}\left(1 - \frac{R_1}{2R_2}\right) \tag{A.6}$$

$$R_1 = \frac{\sqrt{a^4 z^2 + b^4 x^2}}{b^2} \tag{A.7}$$

$$R_2 = \frac{(a^4 z^2 + b^4 x^2)^{3/2}}{a^4 b^4} \tag{A.8}$$

at equator:

$$\sigma_\varphi = \frac{p_T a}{2t} \tag{A.9}$$

$$\sigma_\phi = \frac{p_T a}{t}\left(1 - \frac{a^2}{2b^2}\right) \tag{A.10}$$

at the top:

$$\sigma_\varphi = \sigma_\phi = \frac{p_T a^2}{2bt} \tag{A.11}$$

II. Hydrostatic Pressure

i. Cylinder

$$\sigma_z = 0 \tag{A.12}$$

$$\sigma_\phi = \frac{\rho_p gzR}{t} \tag{A.13}$$

ii. Sphere

below the liquid's surface d:

$$\sigma_\varphi = \frac{\rho_p gR^2}{6t}\Big(\frac{3d}{R} + \frac{2cos^2\varphi}{1+cos\varphi} - 1\Big) \tag{A.14}$$

$$\sigma_\phi = \frac{\rho gR^2}{6t}\Big(\frac{3d}{R} + \frac{(3+2cos\varphi)2cos\varphi}{1+cos\varphi} - 5\Big) \tag{A.15}$$

above the liquid's surface d, with W_l as the total liquid weight is Newton:

$$\sigma_\varphi = \frac{W_l}{2\pi R_2 tsin^2\varphi} \tag{A.16}$$

$$\sigma_\phi = -\sigma_\varphi \tag{A.17}$$

$$W_l = \rho g\pi d^2\Big(R - \frac{d}{3}\Big) \tag{A.18}$$

iii. Ellipsoid

below the liquid's surface d with W_l the weight of the liquid below z:

$$\sigma_\varphi = \frac{W_l}{2\pi R_2 tsin^2\varphi} + \frac{\rho gR_2(d-z)}{2t} \tag{A.19}$$

$$\sigma_\phi = \frac{-W_l}{2\pi R_1 tsin^2\varphi} + \frac{\rho gR_2(d-z)}{2t}\Big(2 - \frac{R_2}{R_1}\Big) \tag{A.20}$$

$$W_l = \frac{\pi a^2 z^2}{3b^2}(3b - z) \tag{A.21}$$

above the liquid's surface d with W_l the weight of the liquid:

$$\sigma_\varphi = \frac{W_l}{2\pi R_2 tsin^2\varphi} \tag{A.22}$$

$$\sigma_\phi = \frac{-W_l}{2\pi R_1 tsin^2\varphi} \tag{A.23}$$

$$W_l = \frac{\pi a^2 d^2}{3b^2}(3b - d) \tag{A.24}$$

III. Stress due to own weight

i. Cylinder

$$\sigma_z = \rho g z \tag{A.25}$$

$$\sigma_\phi = 0 \tag{A.26}$$

ii. Sphere

$$\sigma_\varphi = \frac{\rho g R}{1 + cos\varphi} \tag{A.27}$$

$$\sigma_\phi = -\rho g R\Big(\frac{1}{1 + cos\varphi} - cos\varphi\Big) \tag{A.28}$$

iii. Ellipsoid

Appendix B

Ellipsoid shell thickness due to internal pressure with Von Mises criterion

Von Mises criterion for plane stress without shear:

$$\sigma_v = \sqrt{\sigma_\varphi^2 - \sigma_\varphi \sigma_\phi + \sigma_\phi^2} \tag{B.1}$$

Stress in Ellipsoid:

$$\sigma_\varphi = \frac{p_T R_1}{2t} \tag{B.2}$$

$$\sigma_\phi = \frac{p_T R_1}{t}\Big(1 - \frac{R_1}{2R_2}\Big) \tag{B.3}$$

With:

$$R_1 = \frac{\sqrt{a^4 z^2 + b^4 x^2}}{b^2} \tag{B.4}$$

$$R_2 = \frac{(a^4 z^2 + b^4 x^2)^{3/2}}{a^4 b^4} \tag{B.5}$$

Make two terms A and B independent of thickness and internal pressure:

$$A = \frac{R_1}{2} \tag{B.6}$$

$$B = R_1\Big(1 - \frac{R_1}{2R_2}\Big) \tag{B.7}$$

$$\sigma_\varphi = p_T t^{-1} A \tag{B.8}$$

$$\sigma_\phi = p_T t^{-1} B \tag{B.9}$$

Fill in Von Mises criterion:

$$\sigma_v = \sqrt{p_T^2 t^{-2} A^2 - p_T^2 t^{-2} AB + p_T^2 t^{-2} B^2} \tag{B.10}$$

After rearranging by getting t to one side and replacing Von Mises stress with material yield stress we get:

$$t = \frac{p_T}{\sigma_{yield}} \sqrt{A^2 - AB + B^2} \tag{B.11}$$

Appendix C

Modelling of force, shear and moment lines

This chapter shows plots taken from the model which indicate the method used to determine shear and moment at a certain location z. Figure C.1 shows the three main variants used in the model which are indicated as point, even and sloped load. The force line indicated in what way the force is acting on the beam. The beam can be assumed to be anchored around coordinate 16 on the horizontal axis, the moment and shear line values at that point than become the reaction forces necessary to counteract the force being applied. A note on the point load force line is it's apparent value of zero over the entire beam. This is not true since at the point where the moment line meets the horizontal axis on the left most side a point load is applied. The point load shear and moment lines are quite trivial since the shear line is constant and equal to the force applied. The moment line is simply the force times the distance from the force. It is also important to note that the moment is defined to be negative for positive value of force and vice versa. The even and sloped load forces are defined as force per unit distance. This creates a situation where the force line needs to be integrated to create the shear line. In the areas where no force is applied the shear remains constant, similar to the point load case. The moment line value for the even load is determined by the value of the shear line at the same coordinate and a reference distance. The shear line value is the total load applied up until that point but cannot be multiplied by the same coordinate like the point load case. A reference distance needs to be applied. This reference distance is equal to: $z' = z + 0.5(z_{end} - z)$. where z is the desired coordinate and z_{end} is the location where the even loads ends. The reason for this is that the moment is determined by converting the even load up until that point into a point load with the same total force and moment. This point load is located where the total surface under the force line(i.e. the shear line), is half that of the surface area under the force line of the desired coordinate. For the even load case this is trivial in the middle and hence we get this simple equation.

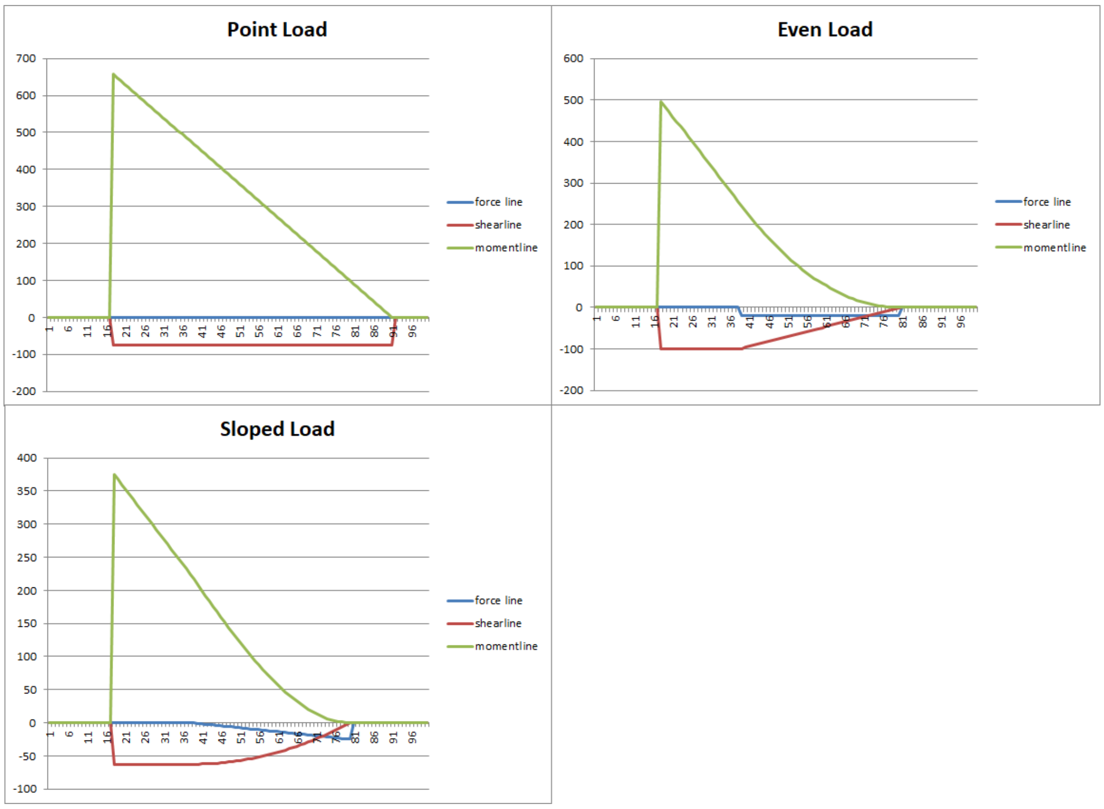
The sloped load moment line works similar but is a little bit more complicated in determining the reference location. Figure C.2 shows a triangle which can be assumed to be the force line of a sloped load. This means that its surface area (i.e. shear line) is equal to:

$$S_{total} = S_1 + S2 = \frac{a \cdot b}{2} \tag{C.1}$$

and per definition:

$$S_1 = S_2 \tag{C.2}$$

Figure C.1: *Force, shear and moment lines of the three main load methods.*



d needs to be determined since it's the location of the reference location, since this is where there is an equal surface on each of its side.

$$a \cdot b = \frac{d \cdot e}{2} \tag{C.3}$$

Since the triangles are similar:

$$\frac{a}{b} = \frac{d}{e} \tag{C.4}$$

thus:

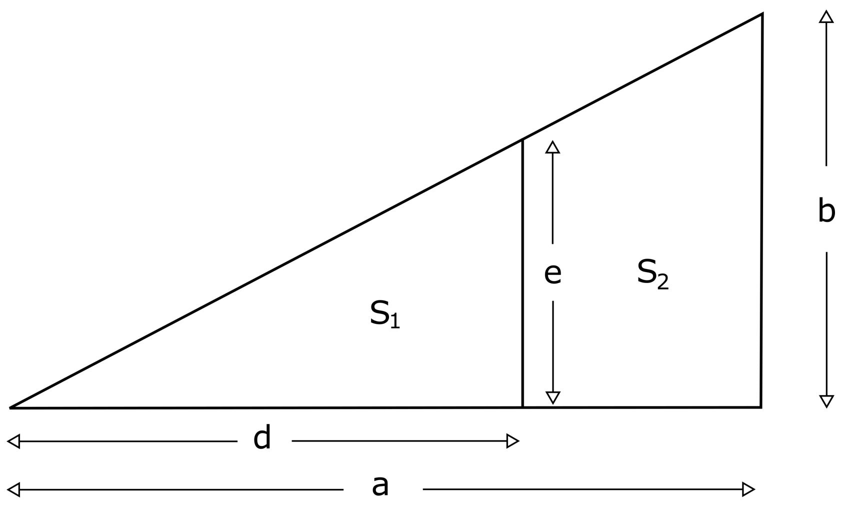
$$e = \frac{a \cdot b}{2d} \tag{C.5}$$

$$\frac{a}{b} = \frac{2d^2}{a \cdot b} \tag{C.6}$$

finally:

$$d = \sqrt{\frac{a^2}{2}} \tag{C.7}$$

Figure C.2: *Triangle showing surface under a sloped load.*



Appendix D

Centroid of geometric shapes

Centroid for geometric shapes are given here, they are used to determine the center of gravity of objects. The location of the center of gravity for a body is given in equation D.1. where M is the total mass and dm a mass increment.

$$\bar{x} = \frac{\int_0^M xdm}{\int_0^M dm} \tag{D.1}$$

This equation can be rewritten.

$$\bar{x} = \frac{\int_0^L x\frac{dm}{dx}dx}{M} \tag{D.2}$$

The centroid of different shapes is described below. Cylinder (s = cross-section surface, r = radius, h = height):

$$\frac{dm}{dx} = \rho * S = \rho\pi r^2 \tag{D.3}$$

$$r = constant \tag{D.4}$$

$$\int_0^L x\frac{dm}{dx}dx = \frac{\pi\rho r^2}{2}h^2 \tag{D.5}$$

$$\bar{x} = \frac{h}{2} \tag{D.6}$$

Cone (R = radius of the base):

$$r = \frac{R}{h}x \tag{D.7}$$

$$\int_0^L x\frac{dm}{dx}dx = \frac{\rho\pi R^2}{4h^2}h^4 \tag{D.8}$$

$$M = \frac{\pi\rho}{3}R^2h \tag{D.9}$$

$\bar{x}$ from the tip of the cone:

$$\bar{x} = \frac{3}{4}h \tag{D.10}$$

Thin hollow Cone:

$$S = 2\pi r t \tag{D.11}$$

$$\frac{dm}{dx} = 2\rho\pi r t \tag{D.12}$$

$$r = \frac{R}{h} x \tag{D.13}$$

$$\int_0^h x \frac{dm}{dx} dx = \frac{2\rho\pi R t}{3h} h^3 \tag{D.14}$$

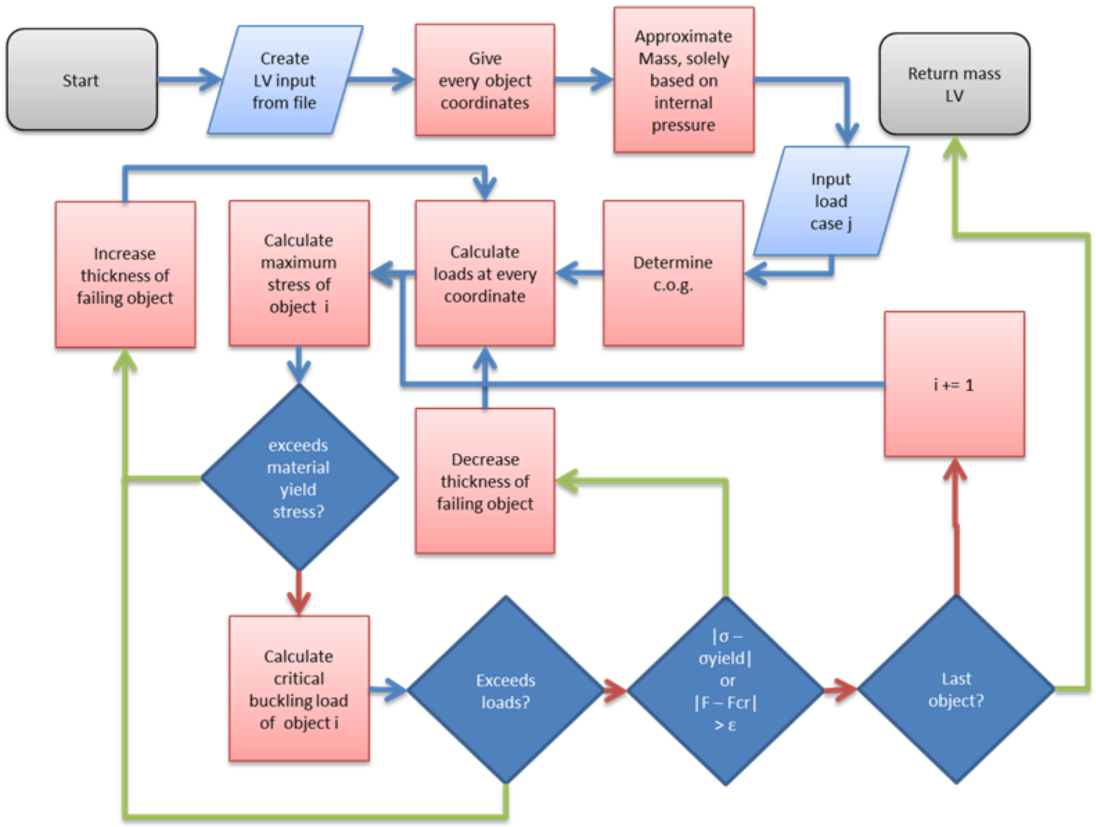
$$M = t\rho\pi R \sqrt{R^2 + h^2} \tag{D.15}$$

$$\bar{x} = \frac{2}{3} h \tag{D.16}$$

Appendix E

Model Flow Chart

Figure E.1: *Flow chart of current model (version 1)*


Start
Create LV input from file
Give every object coordinates
Approximate Mass, solely based on internal pressure
Return mass LV
Input load case j
Increase thickness of failing object
Calculate maximum stress of object i
Calculate loads at every coordinate
Determine c.o.g.
i += 1
exceeds material yield stress?
Decrease thickness of failing object
Calculate critical buckling load of object i
Exceeds loads?
|σ – σyield| or |F – Fcr| > ε
Last object?

Appendix F

Atlas Centaur Simulation Input

Full code used for input for the Atlas-Centaur launch vehicle.

Atlas-Centaur

```
Launchvehicle Database::atlasCentaur()
{
    Launchvehicle AtlasCentaur("Atlas-Centaur");
    AtlasCentaur.createStages(2);

    double massFuel,massOxidizer,diameter,internalPressure, height, aOverBTop,
     aOverBBottom, ullageVolume, propellantTemperature, propellantDensity;
    Materials material;
    Propellants propellant;


     ///////////////////////////////
    // ATLAS
    ///////////////////////////////
    AtlasCentaur.stage[0] = new Stage(123500,true,false,3.05,false, "engine");

    // ENGINES
    //
    AtlasCentaur.stage[0]->createEngines(3);
    AtlasCentaur.stage[0]->engine[0] = new Engine("Sustainer Engine"
     ,"Rocketdyne",257.98  * pow(10,3),1300.0,3.43,1.2,2.231546);
    AtlasCentaur.stage[0]->engine[1] = new Engine("Booster Engine  "
     ,"Rocketdyne",74.7327 * pow(10,4),1452.0,3.43,1.2,2.231546);
    AtlasCentaur.stage[0]->engine[2] = new Engine("Booster Engine  "
     ,"Rocketdyne",74.7327 * pow(10,4),1452.0,3.43,1.2,2.231546);

    // THRUST FRAME
    //
    AtlasCentaur.stage[0]->createThrustframe();
    AtlasCentaur.stage[0]->thrustframe[0] = new ThrustFrames();
    AtlasCentaur.stage[0]->thrustframe[0]
     ->wall.addLayer(&MaterialDatabase::FullHard301SS);

    // FUEL TANK
    //
    massFuel =  AtlasCentaur.stage[0]->getTotalPropellantMass()
     / (1.0 + AtlasCentaur.stage[0]->engine[0]->getOxidizerFuelRatio());
    AtlasCentaur.stage[0]->createFuelTanks(1);
    diameter = AtlasCentaur.stage[0]->getStageDiameter();
    internalPressure = 6.0 * pow(10,5);
    material = MaterialDatabase::FullHard301SS;
    aOverBTop = 1.0;
    aOverBBottom = 1.0;
    propellant = Propellants::RP1();
    propellantTemperature = 290.0;
    propellantDensity = propellant.getDensity(propellantTemperature);
    ullageVolume = 0.025;
    bool loadCarrying = true;
    AtlasCentaur.stage[0]->fuelTank[0] = new
     Tank(massFuel,diameter,internalPressure,propellantTemperature
      ,propellantDensity,"Ellipse","Balloon","Cone",aOverBTop,aOverBBottom
      ,ullageVolume,loadCarrying);
    AtlasCentaur.stage[0]->fuelTank[0]
     ->cylinderWall.addLayer(&MaterialDatabase::FullHard301SS);
    AtlasCentaur.stage[0]->fuelTank[0]
     ->topWall.addLayer(&MaterialDatabase::FullHard301SS);
    AtlasCentaur.stage[0]->fuelTank[0]
     ->bottomWall.addLayer(&MaterialDatabase::FullHard301SS);

    // OXIDIZER TANK
    //
    massOxidizer =  AtlasCentaur.stage[0]->getTotalPropellantMass()
     / (1.0 + (1.0/AtlasCentaur.stage[0]->engine[0]->getOxidizerFuelRatio()));
    AtlasCentaur.stage[0]->createOxidizerTanks(1);
    internalPressure = 3.4 * pow(10,5);
    material =  MaterialDatabase::FullHard301SS;
    aOverBTop = 1.36;
    aOverBBottom = -1.0;
    propellant = Propellants::LiquidOxygen();
    propellantTemperature = 97.0;
    propellantDensity = propellant.getDensity(propellantTemperature);
    ullageVolume = 0.025;
    loadCarrying = true;
    AtlasCentaur.stage[0]->oxidizerTank[0] = new Tank(massOxidizer
     ,diameter,internalPressure,propellantTemperature,propellantDensity
     ,"Ellipse","Balloon","Ellipse",aOverBTop,aOverBBottom,ullageVolume
     ,loadCarrying);
    AtlasCentaur.stage[0]->oxidizerTank[0]
     ->cylinderWall.addLayer(&MaterialDatabase::FullHard301SS);
    AtlasCentaur.stage[0]->oxidizerTank[0]
     ->topWall.addLayer(&MaterialDatabase::FullHard301SS);
    AtlasCentaur.stage[0]->oxidizerTank[0]
     ->bottomWall.addLayer(&MaterialDatabase::FullHard301SS);




    // PRESSURANT TANKS
    Pressurizers pressurant = Pressurizers::Helium();
    int numberOfPressureTanks = 8;
    AtlasCentaur.stage[0]->createPressurantTanks(numberOfPressureTanks);
    internalPressure = 200.0 * pow(10,5);
    double pressurantTemperature = 273.0;
    material =  MaterialDatabase::FullHard301SS;
    aOverBTop = 1.0;
    aOverBBottom = 1.0;
    ullageVolume = 0.10;
    for (int i=0; i < numberOfPressureTanks ; i++)
    {
        AtlasCentaur.stage[0]->pressurantTank[i] = new
         PressurantTanks(material,pressurant,ullageVolume
         ,pressurantTemperature,internalPressure);
    }

    // INTERSTAGE
    //MADE OF 301 HALF HARD STAINLESS STEEL
    AtlasCentaur.stage[0]->createInterstages(1);
    AtlasCentaur.stage[0]->interstage[0]
     ->wall.addLayer(&MaterialDatabase::FullHard301SS);
    AtlasCentaur.stage[0]->interstage[0]
     ->wall.addStiffener(&MaterialDatabase::FullHard301SS);
    AtlasCentaur.stage[0]->interstage[0]
     ->wall.addRing(&MaterialDatabase::FullHard301SS);

    ///////////////////////////////
    //  CENTAUR
    //////////////////////////////
    AtlasCentaur.stage[1] = new
     Stage(13948.0,true,true,3.05,true, "engine"); //max propellant mass
      , integrated tanks, fuel on top, stage diameter

    // PAYLOAD
    //
    AtlasCentaur.stage[1]->createPayload();
    AtlasCentaur.stage[1]->payload[0] = new
     Payload("surveyor 6",995.0,1.0,1.0);

    // ENGINES
    //
    AtlasCentaur.stage[1]->createEngines(2);
    AtlasCentaur.stage[1]->engine[0] = new Engine("RL10A-3","Prat & Whitney"
     ,66700.0,227.5,1.5,1.0,4.83); // name, manufacturer, thrust , mass ,
    // length , diameter, O/F
    AtlasCentaur.stage[1]->engine[1] = new Engine("RL10A-3","Prat & Whitney"
     ,66700.0,227.5,1.5,1.0,4.83);

    // THRUST FRAME
    //
    AtlasCentaur.stage[1]->createThrustframe();
    AtlasCentaur.stage[1]->thrustframe[0] = new ThrustFrames();
    AtlasCentaur.stage[1]->thrustframe[0]
     ->wall.addLayer(&MaterialDatabase::FullHard301SS);

    // FUEL TANK
    // made of 301 full hard stainless steel, a over b values are estimated
    // , filled with liquid hydrogen, propellant density based on pressure
    // and temperature via NIST database

    massFuel =  AtlasCentaur.stage[1]->getTotalPropellantMass()
     / (1.0 + AtlasCentaur.stage[1]->engine[0]->getOxidizerFuelRatio());
    AtlasCentaur.stage[1]->createFuelTanks(1);
    diameter = AtlasCentaur.stage[1]->getStageDiameter();
    internalPressure = 1.9 * pow(10,5);
    material = MaterialDatabase::FullHard301SS;
    aOverBTop = 1.0; //spherical top cap
    aOverBBottom = -1.425; //negative since cap is inverted
    ullageVolume = 0.015;
    loadCarrying = true;
    AtlasCentaur.stage[1]->fuelTank[0] = new
     Tank(massFuel,diameter,internalPressure, 18.9,67.0,"Ellipse"
      ,"Balloon","Ellipse",aOverBTop,aOverBBottom,ullageVolume,loadCarrying);
    AtlasCentaur.stage[1]->fuelTank[0]
     ->cylinderWall.addLayer(&MaterialDatabase::FullHard301SS);
    AtlasCentaur.stage[1]->fuelTank[0]
     ->topWall.addLayer(&MaterialDatabase::FullHard301SS);
    AtlasCentaur.stage[1]->fuelTank[0]
     ->bottomWall.addLayer(&MaterialDatabase::FullHard301SS);


    // OXIDIZER TANK
    // made of 301 half hard stainless steel, a over b values are estimated
    // , filled with liquid oxygen, propellant density based on pressure and
    //  temperature via NIST database
    massOxidizer =  AtlasCentaur.stage[1]->getTotalPropellantMass()
     / (1.0 + (1.0/AtlasCentaur.stage[1]->engine[0]->getOxidizerFuelRatio()));
    AtlasCentaur.stage[1]->createOxidizerTanks(1);
    diameter = AtlasCentaur.stage[0]->getStageDiameter();
    internalPressure = 3.2 * pow(10,5);
    material = MaterialDatabase::HalfHard301SS;
    aOverBTop = 1.425;
    aOverBBottom = 1.425;
    ullageVolume = 0.025;
    loadCarrying = true;
    AtlasCentaur.stage[1]->oxidizerTank[0] = new
     Tank(massOxidizer,diameter,internalPressure,97.0,1107.0,"Ellipse"
     ,"Balloon","Ellipse",aOverBTop,aOverBBottom,ullageVolume,loadCarrying);
    AtlasCentaur.stage[1]->oxidizerTank[0]
```

```
     ->cylinderWall.addLayer(&MaterialDatabase::FullHard301SS);
    AtlasCentaur.stage[1]->oxidizerTank[0]
     ->topWall.addLayer(&MaterialDatabase::FullHard301SS);
    AtlasCentaur.stage[1]->oxidizerTank[0]
     ->bottomWall.addLayer(&MaterialDatabase::FullHard301SS);

    // PRESSURANT TANKS
    pressurant = Pressurizers::Helium();
    AtlasCentaur.stage[1]->createPressurantTanks(2);
    internalPressure = 200.0 * pow(10,5);
    pressurantTemperature = 273.0;
    material =  MaterialDatabase::FullHard301SS;
    aOverBTop = 1.0;
    aOverBBottom = 1.0;
    ullageVolume = 0.10;
    AtlasCentaur.stage[1]->pressurantTank[0] = new
     PressurantTanks(material,pressurant,ullageVolume,pressurantTemperature
      ,internalPressure);
    AtlasCentaur.stage[1]->pressurantTank[1] = new
     PressurantTanks(material,pressurant,ullageVolume,pressurantTemperature
      ,internalPressure);

    // FAIRING
    //
    double heightNoseCone;
    AtlasCentaur.stage[1]->createFairings(1);
    diameter = AtlasCentaur.stage[1]->getStageDiameter();
    height = 6.7;
    heightNoseCone = 4.87;
    AtlasCentaur.stage[1]->fairing[0] = new
     Fairing(diameter,height,heightNoseCone,"blunted cone");
    AtlasCentaur.stage[1]->fairing[0]
     ->wallTip.addLayer(&MaterialDatabase::Aluminium5086H36);
    AtlasCentaur.stage[1]->fairing[0]
     ->wallTip.addCore(&MaterialDatabase::AluHoneycomb);
    AtlasCentaur.stage[1]->fairing[0]
     ->wallTip.addLayer(&MaterialDatabase::Aluminium5086H36);
    AtlasCentaur.stage[1]->fairing[0]->wallTip.settc(0.02);
    AtlasCentaur.stage[1]->fairing[0]
     ->wallCylinder.addLayer(&MaterialDatabase::Aluminium5086H36);
    AtlasCentaur.stage[1]->fairing[0]
     ->wallCylinder.addCore(&MaterialDatabase::AluHoneycomb);
    AtlasCentaur.stage[1]->fairing[0]
     ->wallCylinder.addLayer(&MaterialDatabase::Aluminium5086H36);
    AtlasCentaur.stage[1]->fairing[0]
     ->wallCylinder.settc(0.02);

    // PAYLOAD FRAME
    //
    AtlasCentaur.stage[1]->createPayloadFrame();
    AtlasCentaur.stage[1]->payloadframe[0]->setTankAttached(true);
    AtlasCentaur.stage[1]->payloadframe[0]
     ->wall.addLayer(&MaterialDatabase::FullHard301SS);

    return AtlasCentaur;
}
```

Atlas-Centaur