Grain regression analysis

by

Wichard Sullwald

Thesis presented in partial fulfilment of the requirements
for the degree of Master of Science in Applied mathematics
at Stellenbosch University

Department of Mathematical Sciences
Applied Mathematics Division
University of Stellenbosch
Private Bag X1, Matieland 7602, South Africa.

Supervisors:

G.J.F Smit

J.H Knoetze

April 2014

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: .....

Abstract

Grain regression analysis forms an essential part of solid rocket motor simulation. In this thesis a numerical grain regression analysis module is developed as an alternative to cumbersome and time consuming analytical methods. The surface regression is performed by the level-set method, a numerical interface advancement scheme. A novel approach to the integration of the surface area and volume of a numerical interface, as defined implicitly in a level-set framework, by means of Monte-Carlo integration is proposed. The grain regression module is directly coupled to a quasi -1D internal ballistics solver in an on-line fashion, in order to take into account the effects of spatially varying burn rate distributions. A multi-timescale approach is proposed for the direct coupling of the two solvers.

Uitreksel

Gryn regressie analise vorm 'n integrale deel van soliede vuurpylmotor simulasie. In hierdie tesis word 'n numeriese gryn regressie analise model, as 'n alternatief tot dikwels omslagtige en tydrowende analitiese metodes, ontwikkel. Die oppervlak regressie word deur die vlak-set metode, 'n numeriese koppelvlak beweging skema uitgevoer. 'n Nuwe benadering tot die integrasie van die buite-oppervlakte en volume van 'n implisiete numeriese koppelvlak in 'n vlak-set raamwerk, deur middel van Monte Carlo-integrasie word voorgestel. Die gryn regressie model word direk en aanlyn aan 'n kwasi-1D interne ballistiek model gekoppel, ten einde die uitwerking van ruimtelik-wisselende brand-koers in ag te neem. 'n Multi-tydskaal benadering word voorgestel vir die direkte koppeling van die twee modelle.

Acknowledgements

This thesis was sponsored by the fluxion ledger grant (GAUF600.11214.084AA.T-18).

Firstly I would like to extend my most sincere gratitude to all involved in the fluxion program for providing me with the opportunity to research an interesting and challenging subject. Special mention, however, needs to be made for a number of individuals.

Adriaan Steenkamp, engineer at Flamengro (Armscor), who originally proposed the project. Thank you for sharing the vast knowledge you poses on a large number of relevant and related subjects during both formal and informal discussions on the project. Your curiosity and creativity has been something to aspire to.

Werner Rossoau, design engineer at Rheinmetal Denel Munitions. Thank you for all the expertise you shared on internal ballistics and SRM design. Also for being so approachable and willing to assist with a number of matters. Your inputs on the SPP conference paper, and general guidance throughout the project, has been a tremendous asset.

Professor Francois Smit, my primary supervisor. A great deal is to be said of the manner in which you coordinated the project and allowed me free reign to explore. Your advice on matters both technical and personal is something I will carry forward with me. Thank you for your patience and wisdom.

Dankie Moeder en Vader. Julle sal altyd my eerste poging om advies bly. Ek waardeur julle belangstelling in my doene en late, insluitend die projek. Julle vrywilligheid met advies en ondersteunning is baie kosbaar vir my.

Sarita, thank you for listening. Il tuo amore e supporto e' senza conditioni e non meritato.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations and acronyms

1-D	1 dimensional
2-D	2 dimensional
3-D	3 dimensional
AIAA	American institute of aeronautics and astronautics
CAD	Computer aided design
CFD	Computation fluid dynamics
CFL	Courant Frederich Levy
CSAR	Center for simulation of advanced rockets
FMM	Fast marching method
FOM	Face off-setting method
IB	Internal ballistics
IT	Ignition transient
LSM	Level set method
MC	Monte-Carlo
PDE	Partial differential equation
Q1-D	Quasi-1 dimensional
SDF	Signed distance function

SLIC	Simple line interface calculation
SPP	Solid performance program
SRM	Solid rocket motor
STL	Stereo lithography file
VOF	Volume of fluid

Roman letters

A_{nz}	Nozzle exit plane area
A_p	Combustion chamber port area
A_γ	Burning surface area/interface γ surface area
A_s	Burning surface area/interface γ_s surface area
A_{tri}	Area of triangular patch tri
c	Saint Robert's/Vielle's burn rate model coefficient
d	Euclidean distance between two points/vectors
dx, dy, dz	Dimensional grid spacing
d_{min}	The minimum grid spacing of a computational grid in any dimension
$D^{+/-}$	Difference operators for numerical differentiation
e	Saint Robert's/Vielle's burn rate model exponent
F	Thrust generated by an SRM
$g()$	Numerical flux function
H	The Hamiltonian of a function
$H()$	The heavy-side function
H_{env}	A modified heavy-side function for thin envelope MC integration

i, j, k	Index labels
$isovalue$	A real number representing a chosen iso-value of an iso-surface
L	Edge of a triangular path
m	Mass
$\dot{m}$	Mass flow rate
$\dot{m}_s$	Mass flow rate from a burning surface γ_s
$\dot{m}_{nz}$	Mass flow rate through a nozzle exit plane
M	Mach number
$\vec{\mathbf{n}}$	A normal vector of a surface or patch
N	A natural number/the number of MC-points
N_{inside}	The number of MC-points inside an interface
P	Pressure
P_0	Static pressure
$P_{0_{nz}}$	Static pressure at a nozzle exit plane
P_a	Atmospheric/ambient pressure
P_{nz}	Pressure at a nozzle exit plane
PL	A plane in 3-D space
$\dot{r}$	Burn rate of a solid propellant
$\dot{r}^{IB}$	The speed function passed to the grain regression module
R_{gas}	Specific gas constant
s	A parameter for the parametric description of an interface
S	A switch function used for a second order numerical spatial scheme
t	Time

tri	A triangular patch
T_f	Flame temperature
U	Velocity of a flow field
U_{nz}	Velocity of a flow field at a nozzle exit plane
V	The speed with which an interface is advanced
V_{ext}	An extension of the speed function V across a higher dimensional domain
vox	A voxel of a 3-D computational grid
$\vec{\mathbf{x}}$	A position vector

Greek Letters

γ	General interface in 2-D/3-D
γ_S	A general or burning surface interface
κ	The curveture of an interface
Λ	A level set of a function
λ	A weight between 0 and 1
Ω	A 1-D/2-D/3-D domain
Ω_g	A discrete computational grid domain
Ω_{inside}	Domain potion inside an interface
$\Omega_{outside}$	Domain potion outside an interface
ϕ	An implicit interface representation or signed distance function
ϕ_γ	An implicit representation of an interface γ
ρ	Density
δ	The Dirac-delta function
Ψ	Volume

Ψ_γ	Volume enclose by the interface γ
Γ_{gas}	Specific heat ratio of a gas

Subscripts

a	Atmospheric pressure
c	A motor casing
ext	Extension velocities for the LSM
env	An envelope around an interface
i, j, k	Indeces in the x- y- and z-directions
nz	Nozzle exit plane
S	Static pressure

Chapter 1

Introduction

Solid Rocket Motors (SRM) are propulsion systems that deliver a thrust capable of propelling a payload over a distance at high speed. According to Gruntman [1], the earliest forms of SRMs date back to the 13th century and were powered by black powder or gunpowder. Since then, rockets have evolved substantially and an extensive theory of SRMs has come about. For a good description of the general theory of SRMs the reader is referred to Sutton [2] and Nakka [3]. Nakka gives a more informal description of the theory, but makes for a good introduction to the field of rocketry.

In recent years, numerical simulation of SRMs has become an area of interest for the developers of new SRM propulsion systems such as space agencies and weapons manufacturers. The ability to simulate the operational phase of a novel SRM design, without any need for manufacturing of parts or the use of a test bed, provides manufactures the opportunity to test more variations of motor designs and ultimately optimize the motors to a greater extent.

The numerical simulation of an SRM can be sub-categorized into internal ballistics, burn rate predictions, and grain regression analyses. In this thesis the problem of grain regression analysis in SRMs is investigated.

1.1 Solid rocket motors

1.1.1 Motor layout

In order to describe the operation of a rocket motor mathematically, a basic understanding of the layout of a motor, as illustrated in Figure 1.1, is necessary.

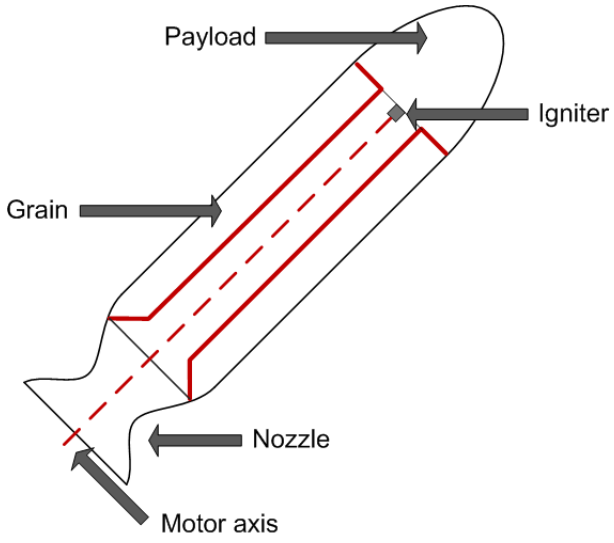


Figure 1.1 – A generic layout of rocket motors

An SRM is, in principle, a simple device. A combustion chamber is loaded with a solid propellant, or grain, which in modern rockets typically comes in the form of an ammonium chloride mixture with an aluminum fuel. The grain is ignited and an exothermic reaction is initiated, during which the propellant burns and generates combustion products or gasses. The generated gases cause a pressure build-up inside the combustion chamber and are expelled from a nozzle, causing a net force to act on the motor.

Graphic illustrations are used to more clearly define what is meant by the grain, motor casing, and burning surface of an SRM. As stated above, the grain of an SRM refers to the solid propellant that is loaded inside the motor. An example of a grain is illustrated in Figure 1.2.

The motor casing, in this case, refers to the casing around the motor combustion chamber. An example of a motor casing is illustrated in Figure 1.3.

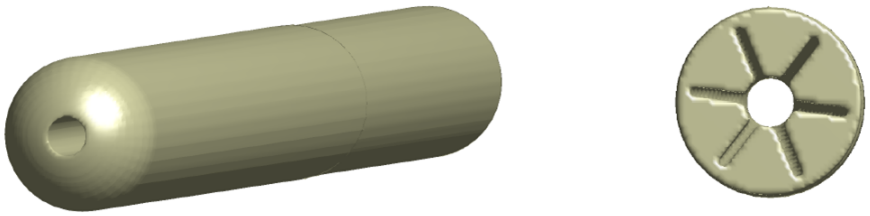


Figure 1.2 – An example of an SRM grain.

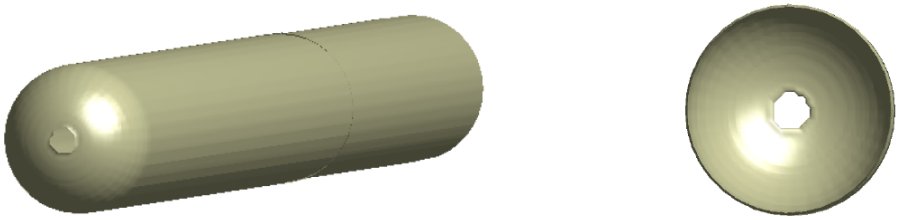


Figure 1.3 – An example of an SRM casing.

The burning surface refers to the surface of the grain exposed to the combustion chamber. The burning surface of the grain example, given in Figure 1.2, is illustrated in Figure 1.4.

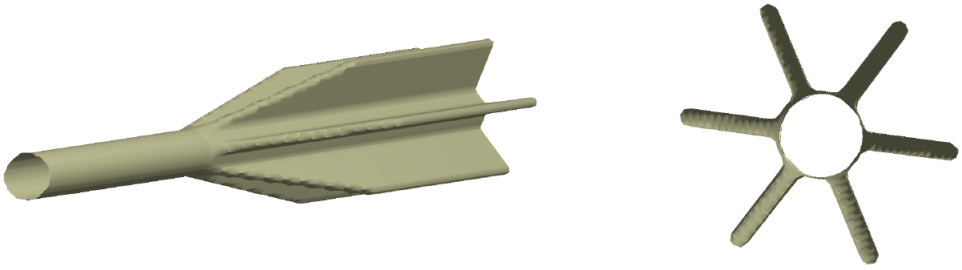


Figure 1.4 – An example of an SRM burning surface.

In order to show the burning surface in relation to the motor casing, Figure 1.5 illustrates the burning surface inside the semi-transparent motor casing.

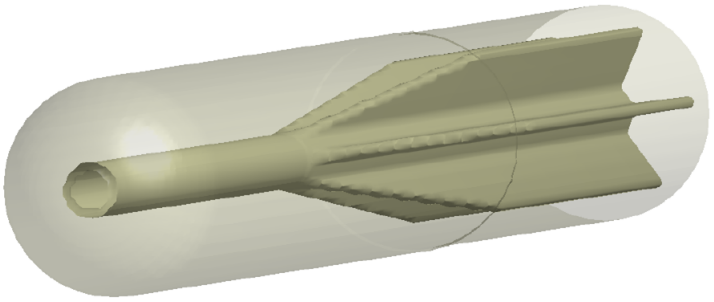


Figure 1.5 – An example of an SRM burning surface and casing.

1.1.2 Motor model

The basic expression for describing the thrust F generated by a solid fuel propulsion system is:

$$F = \dot{m}\, U_{nz} + (P_{nz} - P_a)A_{nz}, \tag{1.1.1}$$

where $\dot{m}$ is the mass flow rate of combustion products out of the nozzle, U_{nz} the exit velocity of the exhaust gasses relative to the motor, P_{nz} the static exit pressure of the nozzle, P_a the atmospheric pressure and A_{nz} the nozzle exit plane area. The simplicity of the equation is deceiving as the thrust is the result of the integral of pressure forces acting on the entire motor, as illustrated in Figure 1.6.

A nozzle design is said to have an optimal expansion ratio if the nozzle exit pressure is equal to the ambient pressures:

$$P_{nz} = P_a.$$

This implies that the second term on the right hand side of equation (1.1.1) will become zero for optimized nozzles and the mass flow rate becomes the only source of thrust.

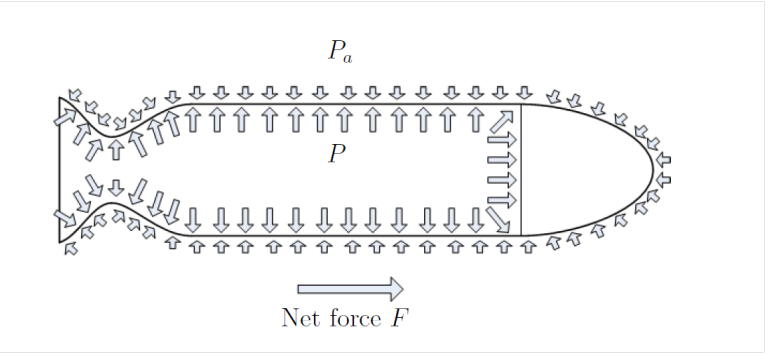


Figure 1.6 – Pressure forces on a SRM during its operational phase.

The mass flow is described by,

$$\dot{m} = \rho A_s \dot{r} - \frac{dP}{dt}. \tag{1.1.2}$$

From equation (1.1.2) it is seen that the mass flow $\dot{m}$ is a function of the burning surface area A_s and the burn rate $\dot{r}$. The solid propellant surface is assumed to regress in a direction normal to the burning surface at a speed called the burn rate. The burn rate of a solid propellant is commonly modelled as a function of the local static pressure P_0 at the burning surface, by:

$$\dot{r} = c(P_0)^e. \tag{1.1.3}$$

This is known as the Saint Robert's or Vielle's law and models the burn rate as a function of pressure. The constants c and e in equation (1.1.3) are obtained empirically for each propellant by performing a number of experimental propellant burns under various pressures and fitting a regression curve through the resulting data. More complicated burn rate models exist, which take phenomena such as erosive burning into account. A more detailed discussion on burn rate models will follow in Section 3.1.2, (the interested reader is referred to Waesche [4], and Geatrix [5] for further reading). A 2-D illustration of the physical process described by equations (1.1.2) and (1.1.3) is shown in Figure 1.7.

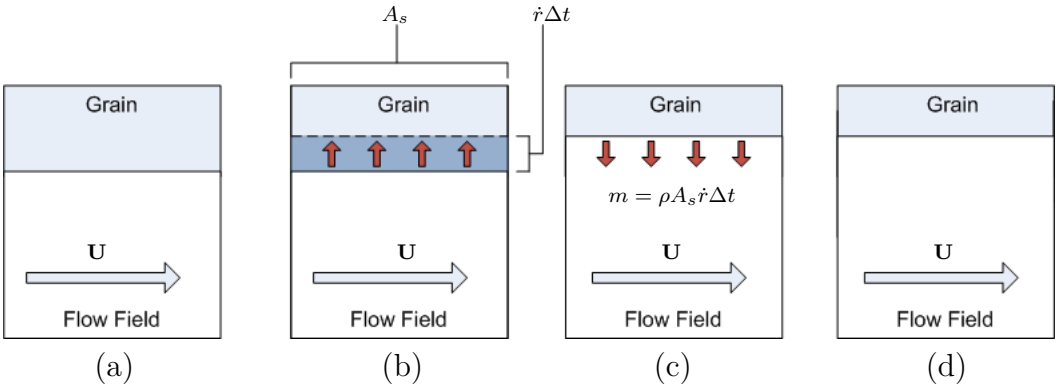


Figure 1.7 – (a) Burn surface and flow field at time t. (b) Regression of burn surface in the normal direction. (c) Mass flow addition to the flow field. (d) Burn surface and flow field at time $t + \Delta t$.

In Figure 1.7 it is shown that during a time step Δt, the burn surface regresses a distance $\dot{r}\Delta t$ and a mass of $m = \rho A_s \dot{r} \Delta t$ is injected into the flow field. This can be seen as the basic process that drives the operation of an SRM and is subject to the following parameters:

Nozzle throat area: The throat area of the nozzle restricts the total mass that can be exhausted from the motor and so affects the pressure inside the combustion chamber, which in turn affects the burn rate of the propellant.

Grain characteristics: The composition of the propellant determines the burn rate constants c and e, of equation (1.1.3), as well as the propellant density ρ of equation (1.1.2).

Burn area profile: The total area of the burning surface and the port areas of the combustion chamber, that evolves as the grain walls regress, are commonly modelled as a function of burnt distance.

These parameters are the main area of focus during the design iterations of a motor. They are manipulated in order to achieve a desired thrust curve which depends on a set of mission requirements.

1.1.3 Thrust curves

A motor's performance can be measured by a thrust-time curve, or simply a thrust curve, which gives the total thrust delivered by a motor as a function of time, as illustrated in Figure 1.8. A thrust curve of an SRM can be divided into three separate phases, namely the ignition transient (IT), quasi-steady state, and burn-out phases, respectively.

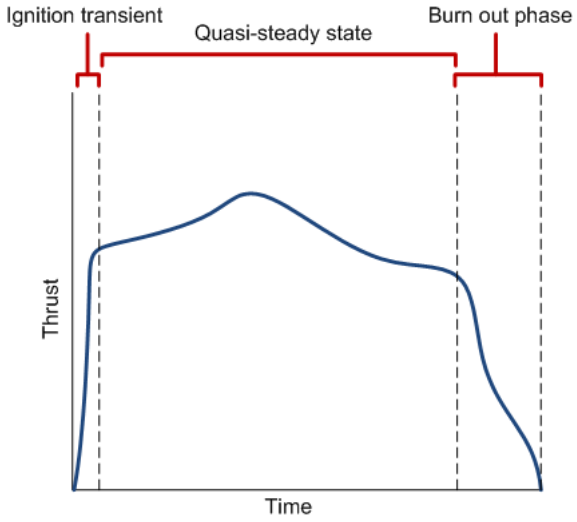


Figure 1.8 – Thrust curve of a SRM.

Ignition transient phase: The IT phase is defined as the time between the ignition signal and the instance at which the SRM reaches a quasi-steady operational state. The IT phase comprises of three sub-stages, referred to as the induction stage, the flame spreading stage, and the chamber filling stage. During the induction phase the local ignition of the grain nearest to the igniter occurs. During the flame spreading phase, a flame front travels along the surface of the grain until the entire exposed surface area of the grain is ignited. The propellant starts to undergo a change of state from solid propellant to hot combustion gasses and the combustion chamber is filled to a point where the pressure inside the motor reaches a quasi-steady level, constituting the chamber filling phase.

Quasi-steady state: Once the quasi-steady operational pressure is reached, the pressure inside the combustion chamber stabilizes and the rate at which combustion products are generated by the burning grain and the mass flow rate exiting the nozzle, are comparably similar. This means that the rate of change of mass inside the combustion chamber becomes negligibly small. For the quasi-steady state, the second term on the right hand side of equation (1.1.2), is assumed to be zero since its contribution is assumed to be negligible, i.e.:

$$\dot{m} = \rho A_s \dot{r}. \tag{1.1.4}$$

Burn-out phase: The burn-out phase of the grain is the time from the first instance at which an area of the motor casing is exposed to the combustion chamber, to the time at which the grain is completely burnt out and there is no propellant in its original solid form remaining inside the combustion chamber. Depending on the geometry of the grain, the burn-out phase might comprise of a significant fraction of the total motor operation and can be seen as a continued quasi-steady state operation. Equation (1.1.4) remains applicable to the motor simulation during this phase.

Thrust curves give information about the characteristics of a motor which can be defined according to the gradient of the thrust curve. The three most basic examples of characteristic thrust curves are progressive, regressive, and neutral burning thrust curves, as illustrated in Figure 1.9. For example a progressive thrust curve has a positive gradient and implies the thrust delivered by the motor increases during the operation of the motor.

These characteristic motor types may be combined to achieve desired effects. For instance, consider a payload that needs to be propelled at a relatively high speed through a low drag environment for an extended period, for example in an air-to-air missile application. A progressive curve followed by a neutral curve, as illustrated in Figure 1.10, would be beneficial since the payload could be accelerated rapidly up to the desired speed, after which the speed could be sustained for a duration with a smaller amount of thrust. This is referred to as a boost-sustain curve.

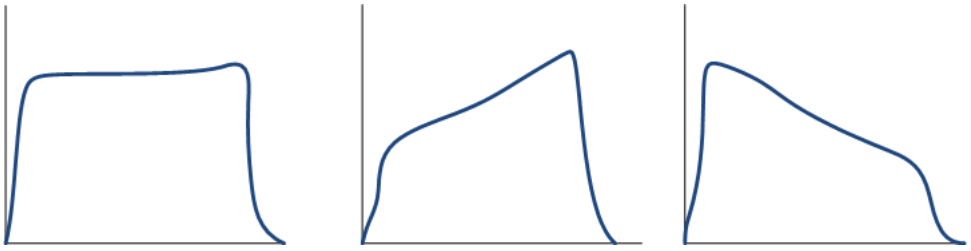


Figure 1.9 – (a) A neutral burning thrust curve. (b) A progressive thrust curve. (c) A regressive thrust curve.

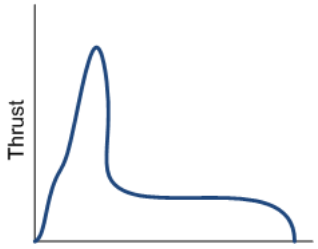


Figure 1.10 – A boost sustain thrust curve.

1.1.4 SRM grain design

The performance of a motor depends on a number of design features, notebly the external shape of the motor casing, the location and size of the igniter, the shape of the nozzle, and the type and shape of the loaded grain. Typically, parameters such as payload mass, flight time and distance to target will be specified in a set of mission requirements. The motor's external shape and aspect ratio can be decided upon. The aerodynamic drag coefficients can be determined either by wind tunnel, or Computational Fluid Dynamics (CFD) simulations.

A required thrust curve can be determined by external ballistic calculations, which the designer can then use as a reference during the rest of the design process. From equations (1.1.1), (1.1.2), and (1.1.3), it can be seen that $\dot{m}, \dot{r}, P$, and A_s, are all co-dependent and form a positive feedback loop, since an increase in the burn area will cause an increase in the mass-flow rate which will

increase the internal pressure inside the motor. This in turn increases the burn rate and in the case of a progressive area profile, will cause a greater rate of increase in the burn area. This makes the motor thrust curve sensitive to the burn area profile and gives merit to the statement that typically, most of a designer's time is spent on refining the burn area profile of a motor design in order to achieve a desired thrust curve.

Apart from the manipulation of the grain design to achieve a desired area profile, the structural integrity of grain should also be taken into account. The forces exerted on the grain during operation can be severe and a structural fracture could cause a motor to fail. Additionally, SRM applications often require the motors to be exposed to varying conditions. For example, a motor used to propel an air-to-air missile will be fixed to the wing of a plane. During its lifespan it might encounter temperatures varying from -50°C to 40°C. The grain structure should be able to withstand the contraction and expansion of both the motor casing and the grain itself as a result of the high temperature variations.

1.2 Grain regression analysis

Grain regression analysis refers to that part of SRM simulation that calculates the evolving burning-surface area and combustion chamber volume and so describing the burn area profiles. For clarity, the definition of a burn area profile is the total burning surface as a function of the distance that the propellant walls have burnt or regressed. Figure 1.11 gives an example of a burn area profile along with illustrations of the grain burning surface at different times during the grain regression.

1.2.1 Previous work

Traditionally, burn areas have been modeled as analytic functions of distance burnt. A good example is the work of Umbel [6]. This can in some instances also be achieved with the help of Computer Aided Design (CAD) software packages, by parametrizing surface models. The Solid Propulsion Program

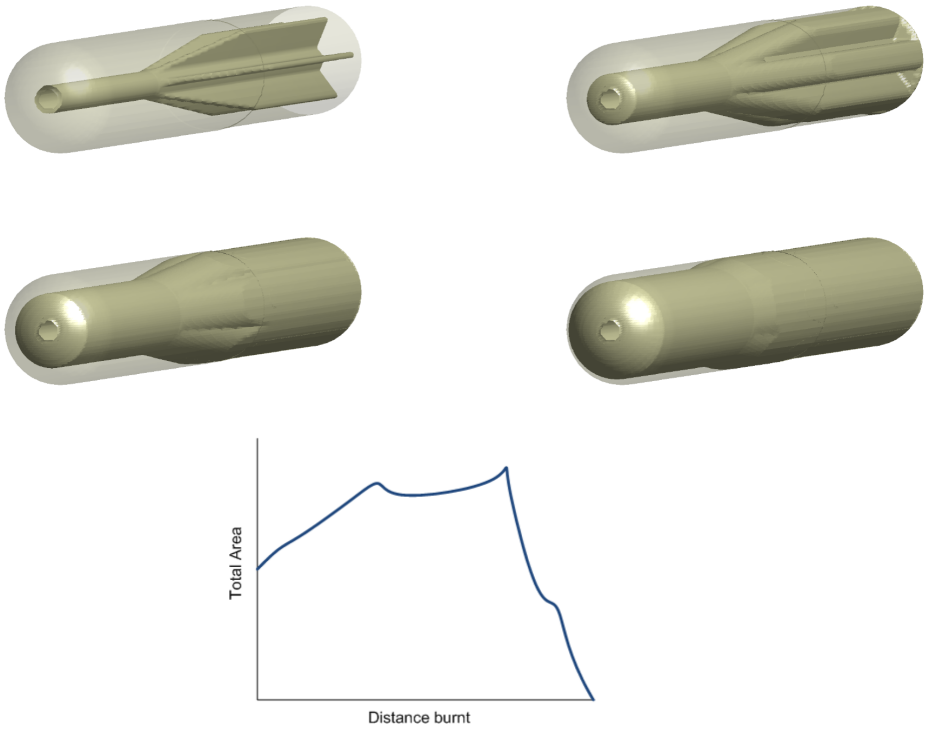


Figure 1.11 – An evolving burning surface and a corresponding area profile of an SRM grain design.

(SPP) [7] uses primitives such as planes, spheres, and the hyperbolic and parabolic tori, to build an analytical model of the surface and so creates a function for calculating the burn area. The analytical or geometric modeling of grain regression provides accurate area descriptions, however, the complexity of some grain designs make analytical modeling extremely difficult and time consuming.

Numerical interface tracking methods have become a viable alternative and some examples of their implementation for the purposes of grain regression analysis have been successful. The Center for Simulation of Advanced Rockets (CSAR) at the university of Illinois has developed a code called ROCGRAIN, a sub-module of their SRM simulation code ROCSTAR, which utilizes a minimum distance function to calculate the burn surface area [8]. The SNPE propulsion software called PIBAL utilizes a constant velocity surface advancement algorithm that treats the burn surface as an advancing interface and calculates the explicit surface area at each time step [9].

A recurring theme in the above-mentioned approaches is the use of constant burn rates to perform the regression of the burning surface. This is an uncoupled approach to the grain regression analysis, since the area profiles are first calculated before the internal combustion simulation is done. An assumption about the spatial distribution of burn rates is implicitly introduced, and no information about the spatial variations of burn rates can be incorporated into the grain regression model. This assumption can potentially introduce significant errors into the simulation since varying burn rates could cause the topological characteristics of the grain surface to vary from the predicted values. In order to circumvent the introduction of significant errors, the common practice is to divide the grain into smaller segments and model each one's area profile separately, choosing the location of the divisions such that the possibility of having a topographical feature spread from one segment to another is minimal.

There has been a number of attempts to utilize interface advancement schemes with spatially and temporally varying burn rates, in order to couple the grain regression analysis and the internal ballistics solvers and accurately describe the surface evolution. Recently CSAR has added the module ROCPROP for the 3-D grain regression that employs a method called the Face Offsetting Method (FOM), developed by Joia [10]. Level set methods introduced by Sethian and Osher [11] has also been employed, most notably by Cavalini [12], however, the LSM was only coupled to a 0-D internal ballistic solver. To our knowledge, the only example of a 3-D grain regression module, coupled to quasi-1-Dimensional (Q1-D) internal ballistics (IB) with spatially varying burn rates is the ROCSTAR code of CSAR.

1.3 Motivation of research

The production and testing of experimental SRMs are expensive and time consuming endeavors. The use of numerical simulation tools in order to reduce the cost and design time during the development of a novel motor has become necessary for SRM manufacturers competing in the market place.

Due to the fact that analytical modelling of complex grain designs is a time consuming task, and difficult to incorporate into internal ballistics simulations, the need for an automated grain regression module became apparent. The effects of the assumption of spatially constant burn rates also warranted further investigation.

The aim of the work presented in this thesis was to establish a working grain regression module capable of handling arbitrary grain designs and couple to a Q1-D internal ballistics (IB) solver with spatially varying burn rates.

1.4 Thesis layout

Chapter 2: Numerical interface advancement

First a general description of the interface tracking problem is presented, after which some popular numerical techniques for solving the equations of motion of a general interface are discussed. The requirements of an interface tracking technique employed for grain regression analysis are discussed and a detailed description of the chosen technique, namely the Level Set Method (LSM), is given.

Some remarks on the properties of implicit surface representation is followed by a discussion on the numerical techniques used to solve the level set equation. A detailed description of the chosen method for signed distance function generation is also presented.

Finally the evaluation of interface properties such as the total surface area (burn area) is discussed and the use of Monte-Carlo integration for this purpose is motivated and explained.

Chapter 3: Internal ballistics coupling

A short description of the internal ballistics solver that will be used to couple the grain regression analysis module is presented. The domain discretization for both the LSM and IB solvers is discussed and the method of coupling the two grids is explained. The use of multiple time-scales for the two solvers is discussed. Finally the layout of the fully coupled algorithm for motor simulation is given.

Chapter 4: Results and validation

The grid dependency of the implicit surface representation by means of signed distances as evaluated. The MC integration methods, including a thin envelope approximation to a surface, are validated through a number of 2-D and 3-D analytical cases and the geometric evaluation of existing SRM desings. The LSM interface advnacement scheme is verified by three analytical cases that pose typical known problem areas for numerical interface advancemt namely corners, cusps, and a changing topololy. Finally a fully coupled simulation of a novel motor design was run and the results compared to the actual static test experimental data.

Chapter 5: Conclusions and recommendations

Some discussions and conclusions ons on the validity of the preposed grain regression module. The signed distance genertion, MC-integration, and level set numerical advancement is each discussed seperately. A number of areas for future reseach are identified, including some initial ideas and strategies.

1.5 Published work

Part of the work presented has been published in the AIAA proceedings of the 2013 Joint Propulsion Conference. Sullwald et al [64] describe the preposed grain regression module, including the MC-integration techniques. Rousseau et al [65] describe a rapid SRM design tool that is to be extended by the grain regression module of Sullwald et al [64]. The papers are included in Appendix C.

Chapter 2

Numerical interface advancement

This chapter is devoted to the numerical model developed for grain regression analysis of SRMs. The model is based on level set methods and Monte-Carlo integration. First the requirements of a technique to be applied to grain regression analyses are set. The general problem of interface tracking is formulated and a number of popular interface tracking techniques are discussed, after which the choice of LSM is motivated. The techniques for setting the initial conditions, a signed distance function, and integration of the surface area and volume of a 3D implicit interface, are developed. Finally the error sources introduced by the interface tracking and parameter integration techniques are discussed in their entirety.

2.1 Requirements of a numerical interface advancement method for grain regression analysis

The objective is to create a grain regression module capable of treating arbitrarily complex grain surfaces and motor casing geometries that can be coupled

to a Q-1D internal ballistics solver. The requirements for the interface tracking procedure include:

- The interface advancement techniques in their entirety should be applicable to 3D interfaces.
- The initial conditions of the interface tracking procedure are to be generated from a CAD output file.
- Arbitrarily complex geometries, including sharp discontinuities in surface gradient and changes in topology, must be dealt with effectively.
- The interface must be evolved or advanced in its normal direction at non-uniform, spatially and temporally varying speeds. This also implies that a method for determining the local normal direction of the interface should exist within the framework of the interface tracking procedure.
- Accurate calculation of geometric properties of the interface, such as surface area, that are relevant to the internal ballistics simulations, should be possible.
- The geometries of motor casings should be incorporated into the calculations of geometric properties to effectively describe the burn out phase of a motor.
- A direct coupling to Q1-D internal ballistics should be possible.

It should be noted that for the purpose of the current work the coupling to 3-D internal ballistics solvers will not be attempted, due to the relatively naive knowledge that currently exists regarding the 3-D flow characteristics of an SRM. It is however a goal of the author to facilitate the possibilities of coupling a grain regression module to a fully 3-D internal ballistics modules in the future, as discussed further in Chapter 5. In order to achieve this, the regression module is subject to number of added requirements, the most substantial of which is the need to facilitate the generation of a 3-D computational grid for each time step of the fully 3-D motor simulation.

2.2 The problem of advancing interfaces

The problem of tracking an interface that moves with a speed V in the normal direction $\vec{\mathbf{n}}$ is common in many physics and engineering applications and a substantial amount of research exists that is focused on interface tracking. A good overview of techniques can be found in the book by Sethian [13].

A closed interface γ on a domain Ω, can be seen as a boundary between two regions of the domain, say Ω_{inside} and $\Omega_{outside}$ as illustrated in Figure 2.1

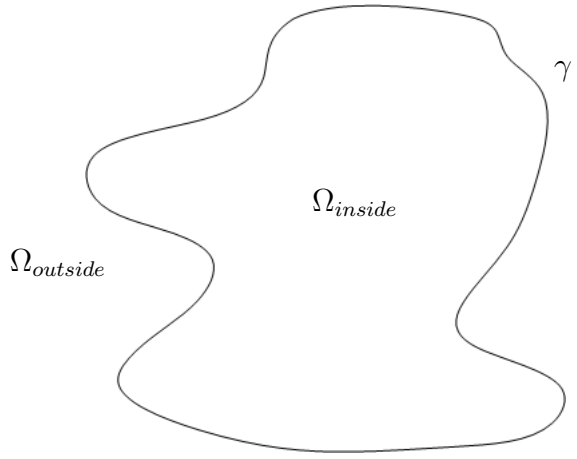


Figure 2.1 – A 2-D interface γ, separating Ω_{inside} from $\Omega_{outside}$.

One of the main challenges is the formulation of the speed function V, which may depend on local properties such as curvature, global properties such as the size or area of the interface, or properties independent of the size or shape of the interface. The focus of this chapter will be on tracking an interface with a known speed function and calculating global parameters such as interface length in 2-D or surface area in 3-D. For the purpose of the current work, we will only consider propagation in the normal direction and no external advection velocity is considered, since the physics of SRM grain regression analysis adheres to this restriction.

In Figure 2.2, the parametric representation of γ , a simple closed curve in $\mathbb{R}^2$, as well as its normal direction (or the direction in which it is advanced) is illustrated. Let $\gamma(t)$ be a family of curves generated by moving in its normal direction with a speed $V(t)$. The curve is parametrized so that the position of

γ at time t is given by, $\vec{\mathbf{x}}(s,t)$ for $0 > s > s_{max}$, where s is the spatial parameter used to move along the length of the interface and s_{max} is the length of the interface.

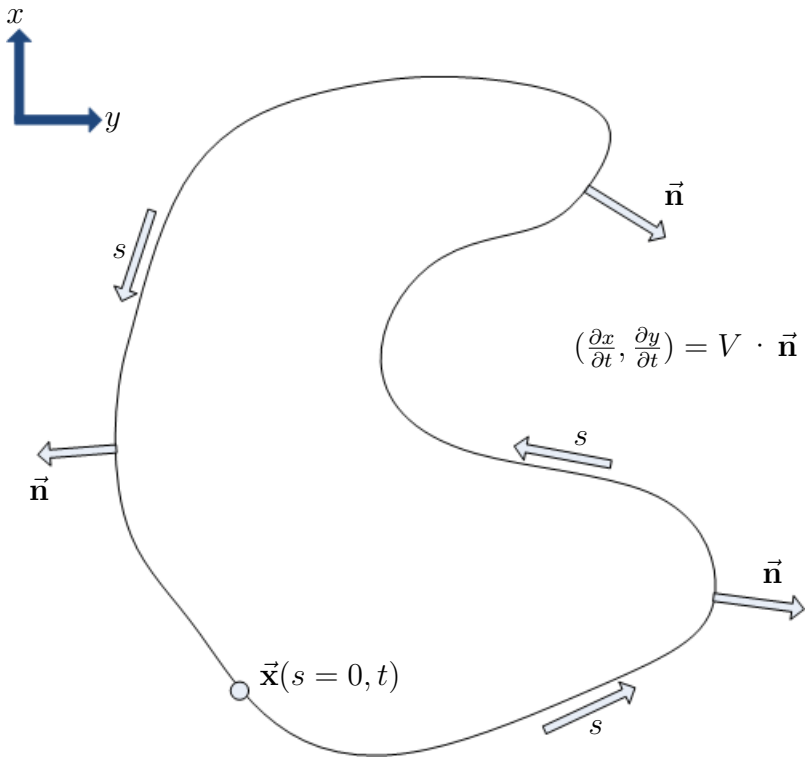


Figure 2.2 – parametric view of an interface propagating in its normal direction.

The normal vector for the chosen parametrization is given by,

$$\vec{\mathbf{n}} = \frac{\partial y/\partial s}{((\partial x/\partial s)^2 + (\partial y/\partial s)^2)^{1/2}}, \frac{-\partial x/\partial s}{((\partial x/\partial s)^2 + (\partial y/\partial s)^2)^{1/2}}. \tag{2.2.1}$$

The equations of motion for the interface can now be written in terms of individual components $\partial\vec{\mathbf{x}} = (\partial x/\partial t, \partial y/\partial t)$ where,

$$\partial x/\partial t = \left(V \frac{\partial y/\partial s}{((\partial x/\partial s)^2 + (\partial y/\partial s)^2)^{1/2}} \right); \tag{2.2.2}$$

$$\partial y/\partial t = \left(-V \frac{\partial x/\partial s}{((\partial x/\partial s)^2 + (\partial y/\partial s)^2)^{1/2}} \right). \tag{2.2.3}$$

2.3 Lagrangian approach

A standard approach to modeling moving interfaces comes from discretizing equations (2.2.2) and (2.2.3). Essentially, the parametrization is discretized into a set of marker particles whose position at any time t are computed and used to reconstruct the front. This is often referred to as the marker and string technique, so called since the linear movement of each of the marker particles in the normal direction with distance $\mathbf{V} \times \Delta t$ can be seen as the string connecting the particle positions at time t with their positions at time $t + \Delta t$, as illustrated in Figure 2.3.

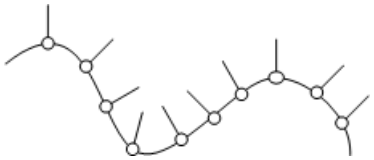


Figure 2.3 – The marker-string method.

These techniques present problems under shock formation and dissipation of the marker particles in cases of high negative or positive curvature. A second challenge is that in some cases, a decision on whether a marker remains physically relevant is required. This is done by what is known as an entropy condition.

2.3.1 Shocks and dissipation

At areas of negative curvature the propagation will cause the markers to bunch together and the arc length between them will decrease, which essentially forms a shock in the solution. Consequently, in order to maintain numerical stability, the time step will need to be decreased and can become small to the point of being impractical. At areas of positive curvature the markers will dissipate, making accurate reconstruction of the front difficult as information is essentially lost in the dissipation of the markers. Both of the above mentioned occurrences are illustrated in Figure 2.4. Note how the markers group together at a concave section of the interface, and dissipate at a convex section.

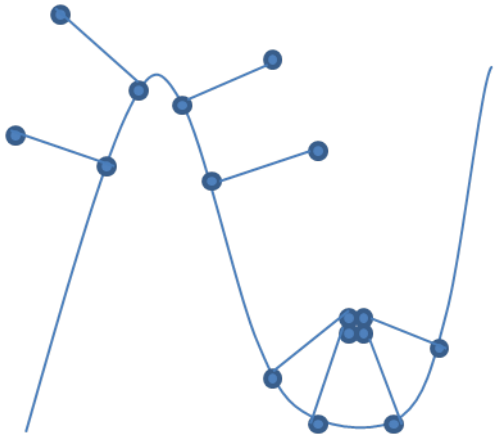


Figure 2.4 – A curve in 2-D advancing with speed in the outward normal direction. Note the shock forming at the local minimum and the dissipation of markers at the local maximum.

A possible route around the problem is to reinitialize the markers when arc lengths between neighboring markers decrease to below a given minimum distance. This would negate the numerical instability but sacrifices some accuracy since new errors are introduced by re-initializing the markers onto an approximation of the interface.

2.3.2 Changes in topology

In the case of a cusp or when a change in topology occurs, the markers can move through a part of the interface and become irrelevant to the physical problem, as illustrated in Figure 2.5. Two circles are advancing outward and the markers that move through the interface are highlighted. An entropy condition is necessary to find the physically relevant solution and discard the non-physical sections of the interface.

The notion of entropy and its connection to physically relevant solutions of propagating interfaces is formally derived and discussed in depth by Sethian [13].

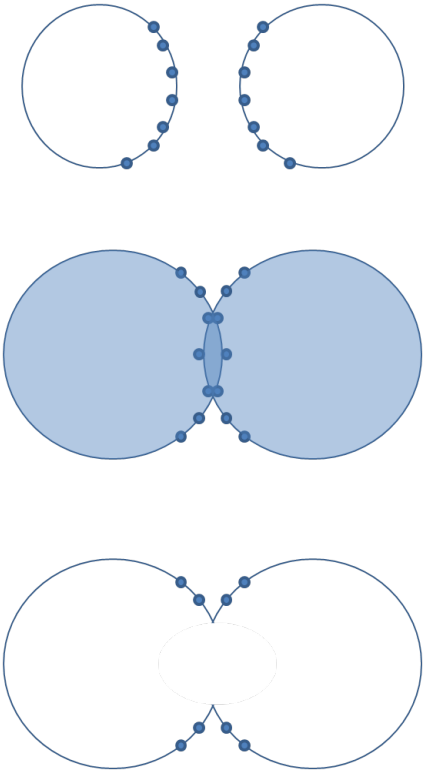


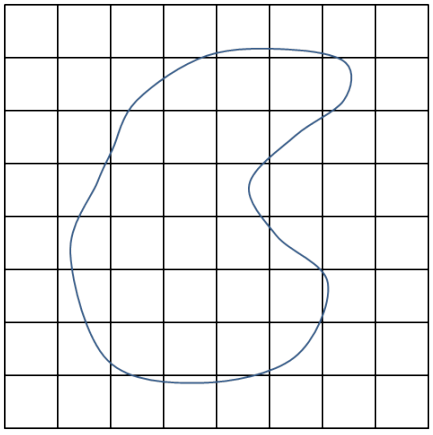
Figure 2.5 – Two circles advancing outward and merging after some time. An entropy condition is needed to establish which markers are kept alive and which are deleted.

2.4 Eulerian approach

An Eulerian approach to the front tracking problem is representing the front on a fixed grid and locating the front at each time step, as it moves through the grid domain. These are often referred to as 'front capturing' techniques, and have been preferred to the Lagrangian 'front tracking' approach in a majority of the literature. The fixed grid approach allows for the problems of shock formation, dissipation, and the difficulty of handling topological changes to be treated more effectively, since they are implicitly resolved and no marker particles need to be deleted. The most popular approaches are the Volume Of Fluid (VOF) and the Level Set Method, or LSM.

2.4.1 Volume of fluid (VOF) method

Introduced by Noh and Woodward [14] in the form of simple line interface calculations, VOF was originally designed for transport under an advection velocity, which depended solely on the position of the front and not on the local shape or orientation. This was later extended by Chorin [15] such that interfaces advancing or propagating in the normal direction could be described. The VOF method is based on a fixed grid setup where the cell value is equal to the fraction of the cell 'inside' the interface. If the cell is on the outside, it is awarded a value of 0, and if the cell is inside, it is awarded a value 1. The cells that are cut by the interface are given a fraction between 0 and 1, so as to represent the portion of the cell that lies within the interface. An example of an interface as well as its representation in the VOF framework is illustrated in Figure 2.6.



0.0	0.0	0.0	0.1	0.2	0.2	0.1	0.0
0.0	0.0	0.3	0.8	1	1	0.4	0.0
0.0	0.1	0.8	1	0.9	0.5	0.1	0.0
0.0	0.3	1	1	0.7	0.0	0.0	0.0
0.0	0.7	1	1	0.9	0.3	0.0	0.0
0.0	0.6	1	1	1	0.9	0.1	0.0
0.0	0.3	0.9	1	0.9	0.7	0.0	0.0
0.0	0.0	0.1	0.2	0.1	0.0	0.0	0.0

Figure 2.6 – The VOF representation of an interface

The cell fractions are updated and the front propagates by performing systematic sweeps along each coordinate direction. In Figure 2.7 an example of a single dimensional sweep is illustrated.

The original method of front reconstruction was by means of straight lines. Since then many elaborate reconstruction techniques, including slanted lines and curves have been introduced, see Lafauri [16] and Hirt [17].

These methods handle changes in topology much better than the Lagrangian methods because of their Eulerian nature, however there are still a number

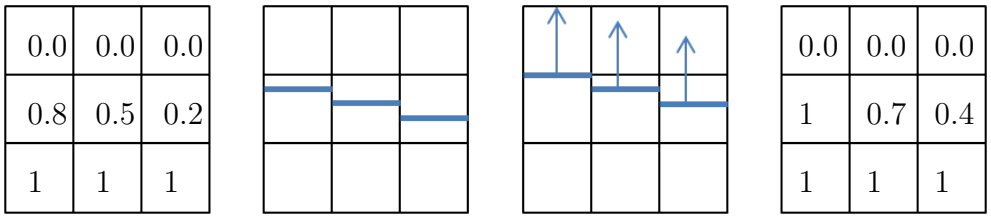


Figure 2.7 – VOF advancement scheme - y direction sweep.

of drawbacks. In order to accurately capture detail of the front, the grid resolution needs to be refined substantially as a result of the simple, crude method of representing the front. There is also a dependency on the grid orientation and for intricate geometries with complex speed functions, this can become problematic. An alternative technique was proposed during the late 1980s that refined the Eulerian framework for interface tracking. The essential idea was to not only classify a cell as inside or outside an interface, but by its distance from the interface. This made for a more detailed representation of the front.

2.4.2 Level set methods

The LSM, introduced by Osher and Sethian [11], is used to advance an interface γ in its normal direction at a speed V, where γ is an interface in 2-D, 3-D or higher dimensions. Both spatially and temporally varying speed functions can be employed and treated by the LSM.

The first step in the development of these techniques started with the analysis of corners and singularities in propagating interfaces. The role of curvature as a regularizing or smoothing term was investigated and the connection to the notion of entropy conditions and shocks in hyperbolic conservation laws in gas dynamics were shown. This led to the realization that schemes from computational fluid mechanics, specifically designed for approximating solutions to hyperbolic conservation laws, can be used to solve the equations of front propagation. For a good description and overview of these techniques refer to Sethian [13].

The method relies on representing γ as the zero level set of a higher dimensional function, say ϕ, defined on a domain Ω that spans γ, so that γ essentially divides Ω into two sub-domains, Ω_{inside} and $\Omega_{outside}$.

The level set $\Lambda_{isovalue}$, for a given $isovalue \in \mathbb{R}$, of a function f is defined as,

$$\Lambda_{isovalue} = \{(\vec{\mathbf{x}}_1, ..., \vec{\mathbf{x}}_n) | f(\vec{\mathbf{x}}_1, ..., \vec{\mathbf{x}}_n) = isovalue\},$$

and thus, the following holds true if γ is implicitly represented as the zero level set of ϕ_γ:

$$\gamma = \{\vec{\mathbf{x}} \mid \phi_\gamma(\vec{\mathbf{x}}) = 0\}, \tag{2.4.1}$$

$$\phi_\gamma(\vec{\mathbf{x}}) > 0 \ \forall \ \vec{\mathbf{x}} \in \Omega_{outside}, \tag{2.4.2}$$

$$\phi_\gamma(\vec{\mathbf{x}}) < 0 \ \forall \ \vec{\mathbf{x}} \in \Omega_{inside}, \tag{2.4.3}$$

where $\vec{\mathbf{x}}$ here refers to an arbitrary point of the domain Ω on which ϕ_γ is defined. A visual illustration of the implicit representation of a 2-D circular interface is given in Figure 2.8.

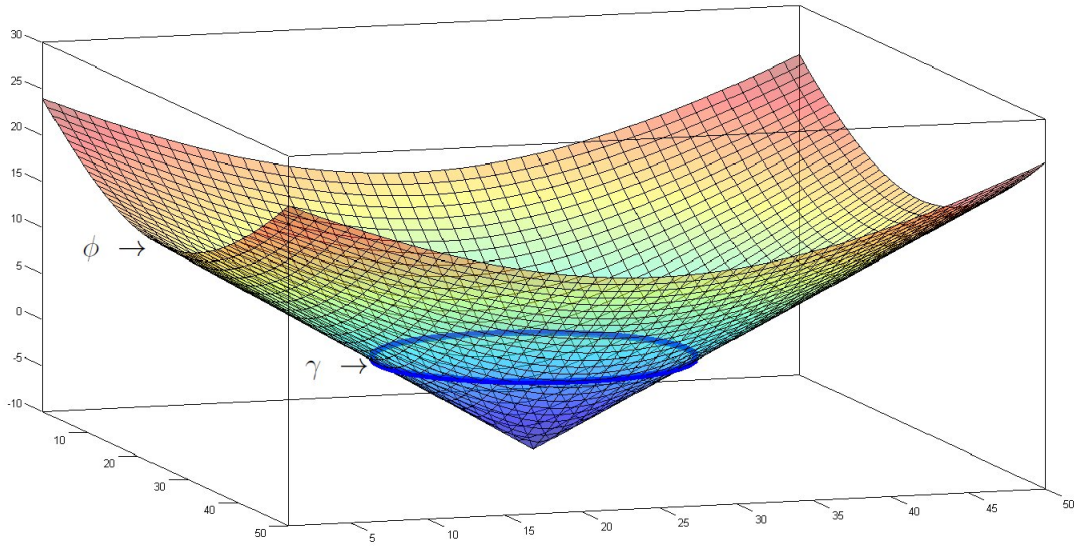


Figure 2.8 – Visual representation of an implicit circular interface.

Once an implicit front representation is initialized, the LSM advances an interface implicitly by solving the level set equation on the implicit function, and so moving forward in time. This equation can be derived by considering the zero level set as the evolving interface γ.

The requirement is that the zero level-set of ϕ_γ remains concurrent with γ^t, the evolving interface, at any given time t. Let $\vec{\mathbf{x}}^t$ be an arbitrary point on the interface. Now the requirement can be written as

$$\phi(\vec{\mathbf{x}}^t, t) = 0. \tag{2.4.4}$$

Finding the material derivative derivative of equation (2.4.4) with respect to t yields,

$$\frac{\partial \phi}{\partial t} + \nabla(\phi(\vec{\mathbf{x}}^t, t)) \cdot \frac{\partial \vec{\mathbf{x}}^t}{\partial t} = 0. \tag{2.4.5}$$

Now let V be the speed at which γ propagates in its normal direction $\vec{\mathbf{n}}$, then $V = \frac{\partial \vec{\mathbf{x}}^t}{\partial t} \cdot \vec{\mathbf{n}}$. Note also that the unit normal of in terms of the implicit representation ϕ can be found by $\vec{\mathbf{n}} = \frac{\nabla \phi}{|\nabla \phi|}$. By substitution equation (2.4.5) reduces to

$$\frac{\partial \phi}{\partial t} + V|\nabla \phi| = 0, \tag{2.4.6}$$

the level set equation, as first proposed by Osher and Sethian, [11].

After the implicit function evolution, the interface at the new time step needs to be extracted from the evolved implicit function. A conceptual illustration of a 2-D example of the procedure is given in Figure 2.9.

2.4.3 Extending the speed function for the LSM

A point perhaps not immediately clear from the above description of the LSM, is the need for extension velocities in order to solve equation (2.4.6) on the entire domain Ω on which ϕ is defined. Depending on the application, the speed with which the interface propagates might be a function of local, or global properties of the interface, such as curvature or enclosed volume, or independent factors such as external physics of the particular problem being simulated.

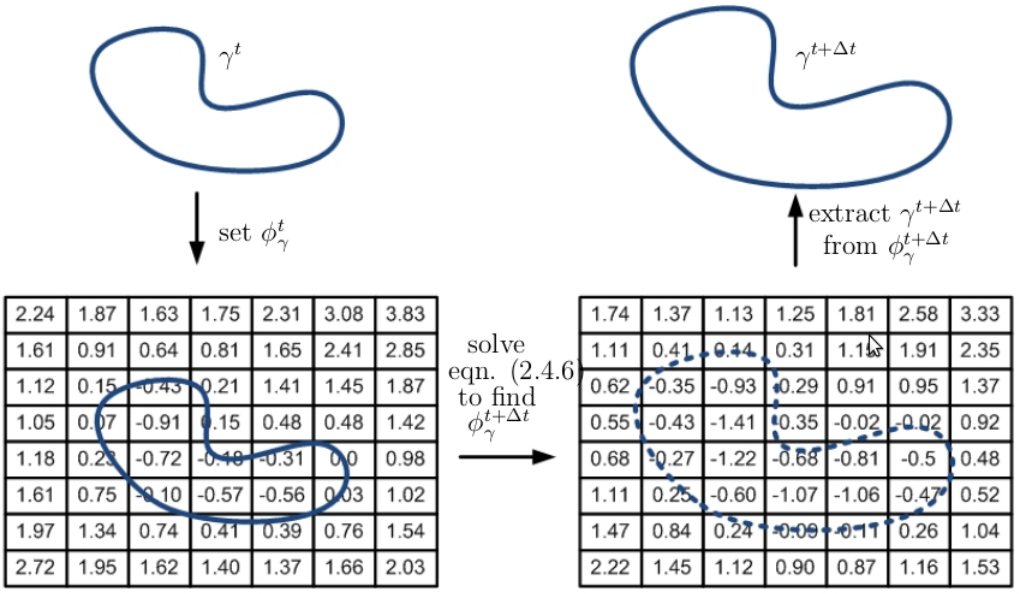


Figure 2.9 – A 2-D illustration of the LSM procedure.

For most applications, such as the problem of grain regression in SRM simulation, the speed function is only defined on the interface itself and the speed with which the interface advances at locations away from the interface, is ill defined and needs to be inferred from the speed function as defined on the interface itself.

Let the speed with which the interface propagates be given by V and the speed function across the higher dimensional domain Ω by V_{ext}. The only restriction on V_{ext} is that it needs to concur with the speed function V at location approaching the interface itself, i.e.

$$\lim_{\vec{x}\to\vec{x}^*} V(\vec{x}) = V(\vec{x}^*), \tag{2.4.7}$$

for any point $\vec{x}^*$ on the interface itself. The choice of V_{ext} can be made such that a particular requirement of the application is fulfilled. For example Adelstein and Sethian [18] described a method for extending a speed function such that the level sets of the implicit function remain equidistant and do not bunch up or spread out. Another way of interpreting this is that the gradient of the implicit ϕ function does not become steep or shallow. This method, relying on the fast marching method, Sethian [19], has become one of the most popular techniques for building extensions to speed functions in LSM application. Some examples

of other techniques are for instance found in Malladi [20], where the speed at a point not on the interface was set equal to the speed at the closest point located on the interface and in Sussman [21], where an integral expression for the speed on the interface was evaluated both on and off the interface in order to build V_{ext}.

For the current application the speed function will only vary in a single dimension along a 3-D interface, since Q1-D internal ballistics simulation will be utilized. The speed at any position of a plane perpendicular to the motor axis, at a given distance along the motor axis, will be set equal to the speed defined for the interface at the given distance.

2.5 Numerical integration of the level-set equation

The level set equation (2.4.6) for advancing an interface may be written in conservative Hamilton-Jacobi form as,

$$\phi_t + H(\phi) = 0, \tag{2.5.1}$$

with $H(\phi)$ given by

$$H(\phi) = V|\nabla\phi| = 0. \tag{2.5.2}$$

The equation can be solved utilizing techniques borrowed directly out of the large number of schemes developed for CFD. Osher and Sethian [11] develop both first and higher order spatial schemes and showed that these schemes converge to the physically relevant entropy satisfying solution of the level set equation. This was achieved by formulating the level set equation in conservative hyperbolic form as $u_t + [G(u)]_x = 0$, and finding the limit of the associated viscous form $u_t + [G(u)]_x = \epsilon u_{xx}$ with $\epsilon \to 0$ [11]. As noted by Sethian [13], the first and second order upwind schemes are sufficient for most applications. More detail about the derivation of the 'entropy satisfying' schemes is given in Appendix B.

The schemes mentioned above make provision for a direction changing speed function, i.e. the speed with which the interface propagates can change sign, meaning that the interface can move in both positive and negative directions. For the application considered in this thesis, the velocity function will remain positive, and therefore simplified versions of the general schemes are presented.

In order to build a numerical scheme, rewrite equation (2.5.1) as

$$\phi_t = -H(\phi). \tag{2.5.3}$$

Assume for the moment a speed function $V = 1$ so that $H(\phi) = |\nabla\phi|$. Furthermore let the function ϕ be defined on a discrete 1-D grid with a uniform grid spacing of Δx. Equation (2.5.3) can then be approximated by

$$\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} = -g(\nabla(\phi)_{i+1/2}^n, \nabla(\phi)_{i-1/2}^n), \tag{2.5.4}$$

where ϕ_i^n refers to the value of ϕ at time $t = n$ at the grid point $x = i$. The gradients $\nabla(\phi)_{i+1/2}^n$ and $\nabla(\phi)_{i-1/2}^n$ can be approximated by central difference operators $\frac{\phi_{i+1}^n - \phi_i^n}{\Delta x}$ and $\frac{\phi_i^n - \phi_{i-1}^n}{\Delta x}$, respectively.

The Hamiltonian $H(\phi) = |\nabla\phi|$ is approximated by the numerical flux $g(\nabla(\phi)_{i-1/2}^n, \nabla(\phi)_{i+1/2}^n)$, where $g(u_1, u_2)$, for two scalars u_1 and u_2, is defined as

$$g(u_1, u_2) = [\max(u_1, 0)^2 + \min(u_2, 0)^2]^{1/2}. \tag{2.5.5}$$

A basic scheme for 1-D can now be written as

$$\phi_i^{n+1} = \phi_i^n + \Delta t(\max(\frac{\phi_i^n - \phi_{i-1}^n}{\Delta x}, 0)^2 + \min(\frac{\phi_{i+1}^n - \phi_i^n}{\Delta x}, 0)^2)^{1/2}. \tag{2.5.6}$$

The scheme is illustrated in Figure 2.10.

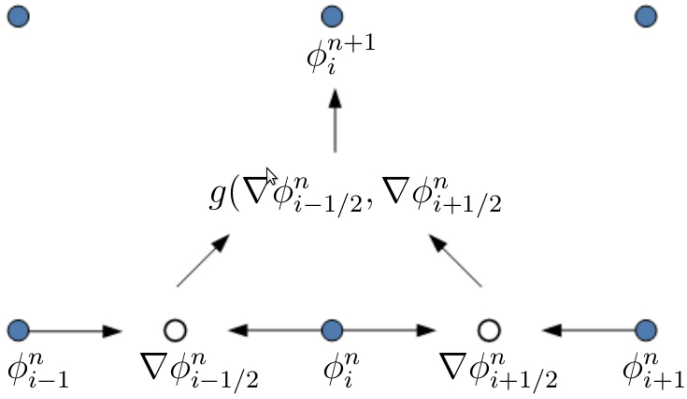


Figure 2.10 – The basic 1-D entropy satisfying scheme for solving the level set equation.

2.5.1 First order scheme

Extending the 1-D scheme and assuming the speed function V to remain positive, a first order 3-D scheme is developed as follows:

$$\phi^{n+1} = \phi^n - (\Delta t)V\nabla^+, \tag{2.5.7}$$

where ∇^+ approximates $|\nabla\phi|$ and is given by

$$\begin{aligned}\nabla^+ &= [\max(D^{-x}_{i,j,k},0)^2 + \min(D^{+x}_{i,j,k},0)^2 + \\ &\quad \max(D^{-y}_{i,j,k},0)^2 + \min(D^{+y}_{i,j,k},0)^2 + \\ &\quad \max(D^{-z}_{i,j,k},0)^2 + \min(D^{+z}_{i,j,k},0)^2]^{1/2}. \end{aligned} \tag{2.5.8}$$

The following notation conventions are applied

$$D^- = \frac{\phi_i - \phi_{i-1}}{dx}, \tag{2.5.9}$$

$$D^+ = \frac{\phi_{i+1} - \phi_i}{dx}. \tag{2.5.10}$$

2.5.2 Second order scheme

For the second order scheme, we define

$$D_i^{--} = \frac{2u_i - 3u_{i-1} + u_{i-2}}{dx}, \tag{2.5.11}$$

$$D_i^{+-} = \frac{u_{i+1} - 2u_i + u_{i-1}}{dx}, \tag{2.5.12}$$

$$D_i^{++} = \frac{2u_{i+2} - 3u_{i+1} + u_i}{dx}. \tag{2.5.13}$$

and modify the spatial difference operators of equation (2.5.8) to:

$$D_i^- = \frac{u_i - u_{i-1}}{dx} + 0.5 * S(D^{--}, D^{+-}), \tag{2.5.14}$$

$$D_i^+ = \frac{u_{i+1} - u_i}{dx} + 0.5 * S(D^{++}, D^{+-}), \tag{2.5.15}$$

where the switch function $S(x, y)$ is defined as

$$S(x,y) = \begin{cases} x & \text{if } |x| \le |y|,\ xy > 0, \\ y & \text{if } |y| < |x|,\ xy > 0, \\ 0 & \text{if } xy \le 0. \end{cases} \tag{2.5.16}$$

By making the above substitution in the first order scheme developed previously, equation (2.5.7), now describes a second order accurate scheme which remains first order accurate for regions located around shocks or high curvature of the interface.

2.5.3 Time step and time integration

Both the first and second order schemes above are based on explicit Euler time integration. The spatial gradient approximation can be used to build higher order Runge-Kutta type implicit time integration. For the current work on SRMs only an explicit time integration will be employed since the benefit of implicit time integration is somewhat negated by the fact that the speed function is dependent on an iterative coupling with an internal ballistics solver. This means that the implicit time integration at different intervals would not be accurate since no prior knowledge of the speed function or burn rate exists.

Another consideration is the fact that the computational cost of the overall algorithm is more notably affected by the computational cost of evaluating the geometric properties of the interface and so improvements on the computational efficiency of the actual solving of the LSM equation is not the primary concern. As far as stability is concerned, the necessary Courant-Frederich-Levy (CFL) condition on the explicit solver requires a time step Δt to adhere to:

$$\Delta t < V * d_{min}, \tag{2.5.17}$$

where d_{min} is the minimum grid spacing in the domain Ω.

2.6 Properties of implicit interfaces

The implicit interface framework of the LSM has some useful properties that allow the intersection, union and averaging of interfaces by simple operations.

2.6.1 Union and intersection of implicit interfaces

Finding the union or intersection of two interfaces by means of geometric arguments might be challenging to employ. With the implicit framework, i.e. representation by means of SDFs or similar functions, intersecting and finding the union of two interfaces reduces to finding the maximum and minimum local values of the superimposed implicit functions.

Let γ_1 and γ_2 be represented by ϕ_1 and ϕ_2, both defined on a common domain Ω. The relations given in Table 2.1 hold true.

Table 2.1 – The implicit analogue for the intersection and union of interfaces.

$\gamma_1 \cup \gamma_2 \leftrightarrow \min(\phi_1, \phi_2)$
$\gamma_1 \cap \gamma_2 \leftrightarrow \max(\phi_1, \phi_2)$

An illustration of what is meant by the union and intersection of two surfaces is given in Figure 2.11.

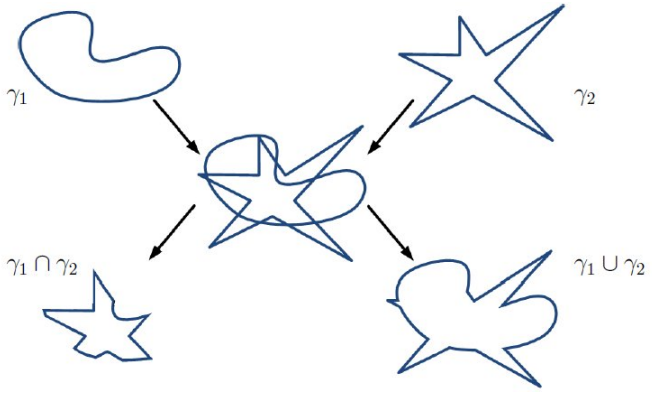


Figure 2.11 – The intersection and union of two interfaces.

2.6.2 Averaging interfaces

Implicit interfaces also allow the averaging or weighted averaging of two interfaces by averaging the implicit surface representations. As before, let γ_1 and γ_2 be represented by ϕ_1 and ϕ_2, both defined on a common domain Ω. By adding ϕ_1 and ϕ_2 and extracting the resulting interface, an average of γ_1 and γ_2 is found. The results for a simple 2-D circle and square are illustrated in Figure 2.12.

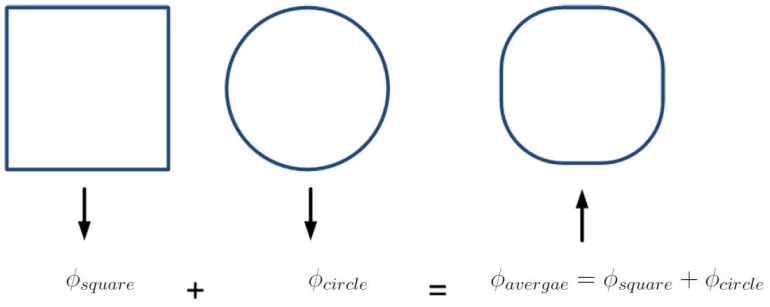


Figure 2.12 – The average of two interfaces found by adding their implicit representations.

It is also possible to perform waited averaging, and thereby finding a smooth merge between two geometries. Define $\phi_1\lambda_2$ as,

$$\phi_1\lambda_2 = \lambda\phi_1 + (1-\lambda)\phi_2 \tag{2.6.1}$$

Figure 2.13 illustrates a smooth transition between a circle and a square by varying λ and employing equation (2.6.1).

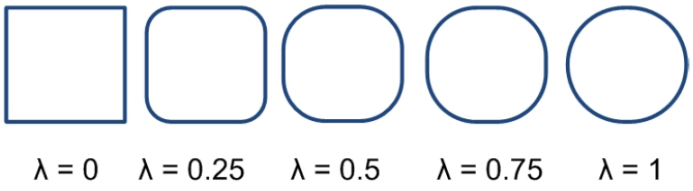
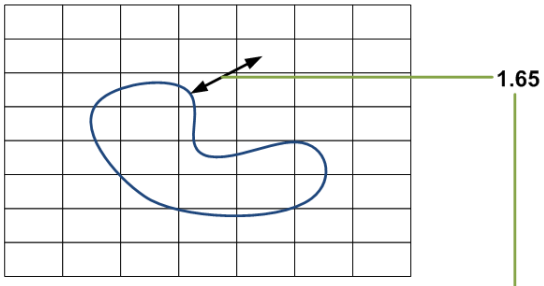


Figure 2.13 – A number of steps in a smooth transition between circle and a sphere

2.7 Signed distance functions of interfaces

2.7.1 Introduction

As discussed before, in order to advance an interface using the LSM, an implicit representation of the interface is required where the interface is embedded as the zero level set of a higher dimensional function. The LSM relies on solving a PDE on the said function and it would therefore be beneficial, numerically, to choose it such as to be relatively smooth. A signed distance function (SDF) possesses both these attributes and is the most commonly used initialization for the LSM. A 2-D example is given in Figure 2.14, where the distance from the interface to the center of each cell is shown.



2.24	1.87	1.63	1.75	2.31	3.08	3.83
1.61	0.91	0.64	0.81	1.65	2.41	2.85
1.12	0.15	-0.43	0.21	1.41	1.45	1.87
1.05	0.07	-0.91	0.15	0.48	0.48	1.42
1.18	0.23	-0.72	-0.48	-0.31	0.0	0.98
1.61	0.75	-0.10	-0.57	-0.56	0.03	1.02
1.97	1.34	0.74	0.41	0.39	0.76	1.54
2.72	1.95	1.62	1.40	1.37	1.66	2.03

Figure 2.14 – A 2-D example of a discrete SDF of an interface on a rectangular grid.

Let γ be a closed interface, defined on a domain Ω, where, as before, closed implies that there is no continuous path in Ω from any point inside the interface to any point outside the interface that does not intersect the interface. Given a coordinate position in Ω, an SDF of γ returns the shortest possible distance to S and signs the distance negative or positive depending on whether the

coordinates are located inside or outside the surface, respectively. If γ were not closed on Ω, there would be a path from a point inside γ to a point outside γ that does not intersect γ, and determining the position along this path at which the region inside γ and the region outside γ meet is therefore arbitrary and the SDF becomes ill-defined.

For a discrete SDF, as is the case in Figure 2.14, the function is defined on a grid Ω_g that is super-imposed over the interface and each grid point is given the value returned by the SDF for the grid point coordinates.

For the purpose of the application considered, it is asumed the interface is available in the stereo lithography (STL) format.

2.7.2 STL representation of surfaces

Since most CAD packages provide the option of exporting surfaces in the form of triangular planer surface patches by means of the STL format, an automated signed distance generator, which utilizes the triangulated format in order to generate 3-D discrete SDFs, was developed. A triangular patch in STL format also contains information about the surface outward normal. The right hand rule convention, as described in appendix A, is applied to the patches. Thus, for a patch given by the vertices $\vec{\mathbf{x}}_1$, $\vec{\mathbf{x}}_2$ and $\vec{\mathbf{x}}_3$, the positive surface normal is found by $\vec{\mathbf{n}} = (\vec{\mathbf{x}}_2 - \vec{\mathbf{x}}_1) \times (\vec{\mathbf{x}}_3 - \vec{\mathbf{x}}_2)$. An extract from an STL file is given in Table 2.2 and its resulting surface patch is shown in Figure 2.15.

Table 2.2 – A single patch in STL format.

```
solid ascii
 facet normal 1.119645e-001 0.000000e+000 9.937122e-001
    outer loop
       vertex 3.586e+000 2.145e+000 1.462e+000
       vertex 1.814e+000 1.632e+000 1.823e+000
       vertex 2.466e+000 2.741e+000 -1.300e+000
    endloop
 endfacet
```

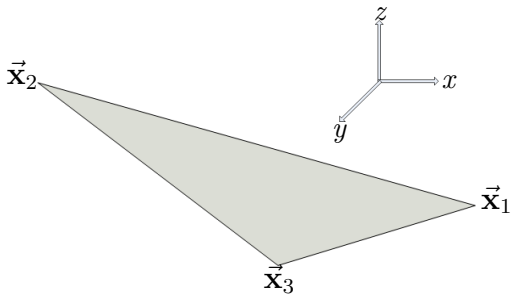


Figure 2.15 – Visual representation of the STL patch given in Table 2.2.

2.7.3 Previous work

Since Sethian and Osher [11, 13] introduced the fast marching method (FMM), there have been a number of variations and improved implementations of the FMM for solving the SDF of surfaces. These methods are based on a boundary value formulation of a simplified Eikonal equation, sometimes referred to as the signed distance equation,

$$|\nabla d| = 1, \tag{2.7.1}$$

where d is the distance from the surface. Equation (2.7.1) is then solved on the grid to determine the distance d. The FMM solves equation (2.7.1) by solving the grid points sequentially. The order is such that the grid points are solved from the lowest distance value to the highest. The reason being that the information only propagates outward to higher values of the SDF and any specific grid point can only be influenced by values lower than itself. This allows the FMM to solve the SDF in $O(n \log n)$ time. By employing untidy priority queues, Yatziv et al. [22] managed an $O(n)$ implementation of the FMM. Swartz and Colella [23] employ a global marching method, first introduced by Kim [24], along with a second order spatially accurate discretization of equation (2.7.1). The global marching procedure makes this method a good candidate for parallel processing. Fast sweeping methods introduced by Zhao [25], is another variation of the FMM for the Eikonol equation that has grown in popularity. Oberhuber [26] performed a fairly recent survey on some numerical methods for generating and recovering the SDF of a given interface

and provides a good source of references for more of the variations of these approaches.

All of these methods, however, require an initialization through some form of analytical computation before computing a complete SDF to a given interface, which makes their advantages secondary to a robust analytical approach.

The naive analytical approach is to perform a brute force calculation visiting each grid point and finding the distance to each triangle surface patch, saving the minimum distance. Finding the distance to a triangle patch is in itself not a simple task. Eberly [27] describes an algorithm for computing the distance from a point to a triangle in 3-D. Finding the scalar dot product of the distance vector and the positive normal (see Section 2.7.2) of the triangle, determines whether the point lies on the inside or the outside of the patch.

Payne and Toga [28], proposed a number of optimizations such as computing only squared distances at first, and taking the square root and calculating the sign only after the minimum squared distance is found. They also proposed storing the triangles in a tree of bounding boxes so that once a distance in, say, box A is found to be significantly less than a distance in box B, all the triangles from box B can be discarded. This procedure is followed hierarchally up the tree and the dimensions of the boxes are decreased incrementally as the grid is refined. An improvement on the method of calculating the sign of each grid point was made by Baerentzen and Aanaes [29], where they define angle weighted pseudo normals to determine the inside or outside location of each grid point. The Meshsweeper algorithm introduced, by Gueziec [30], employs a hierarchical bounding box method, but differs from the work done by Payne and Toga [28] in that the bounding boxes are used to classify the grid points rather than the triangles that make up the surface.

In 2001, Mauch [31] introduced a novel approach where, instead of visiting each point, each triangular patch is observed. Its face, each of its three edges, and each of its three vertices are treated independently, and grid points that might be influenced by each of these are found by defining polyhedra and performing, what is known as, scan conversion on the grid domain. This method gives an implicit sign calculation, since each polyhedron lies either inside or outside the

interface. The method of SDF generation described in Section 2.7.5 is, for the most part, based upon Mauch's methods.

2.7.4 SDF generation

Let γ be a closed surface in STL format, defined on a domain Ω, and let ϕ be the SDF of γ, defined on a grid Ω_g, such that $\gamma \subset \Omega \subset \Omega_g$.

It is important to note that the closest point on a triangular surface to an arbitrary point $\vec{\mathbf{x}}$ in $\Re^3$ could lie on either the face, an edge, or a vertex of a triangular surface. Each of the faces edges and vertices are treated independently, by defining polyhedra that contain the points that will necessarily be closer to the face, edge, or vertex in question, than any other point on the triangular surface, and performing scan conversions to determine which points of the grid lie within the respective polyhedra. These points will be referred to as "possibly closest" to the face, edge, or vertex.

Case 1, face:

Let the vertices $\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2$ and $\vec{\mathbf{x}}_3$ define a triangular patch of γ with normal vector $\vec{\mathbf{n}} = (\vec{\mathbf{x}}_2 - \vec{\mathbf{x}}_1) \times (\vec{\mathbf{x}}_3 - \vec{\mathbf{x}}_2)$, as illustrated in Figure 2.16.

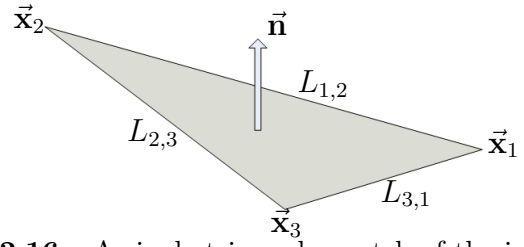


Figure 2.16 – A single triangular patch of the interface γ.

The grid points $\vec{\mathbf{x}} \in \Omega_g$, possibly closest to the triangular face are enclosed by three planes, PL_1, PL_2, and PL_3, perpendicular to the face plane and running through each of the three edges $L_{1,2}$, $L_{2,3}$, and $L_{3,1}$, as illustrated in Figure 2.17. The edge $L_{i,j}$ refers to the edge connecting $\vec{\mathbf{x}}_i$ and $\vec{\mathbf{x}}_j$.

In order to sufficiently describe the plane through $L_{1,2}$, one only needs its normal vector and a point on the plane. Since the edge points are given and known to lie on the planes, there is only the normal vector left to calculate.

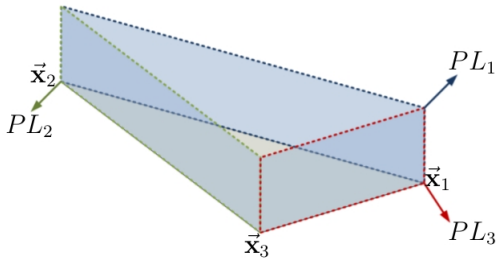


Figure 2.17 – Three planes inclosing points possibly closest to the triangular face.

This can be achieved by finding three consecutive vertices that apply to the right-hand rule convention.

Define $\vec{\mathbf{x}}_*$ as,

$$\vec{\mathbf{x}}_* = \vec{\mathbf{x}}_1 + \vec{\mathbf{n}},$$

as illustrated in Figure 2.18. The vertices $\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_*$, and, $\vec{\mathbf{x}}_2$ now satisfy the right-hand rule convention on the plane PL_1, and the desired normal can be found by

$$\vec{\mathbf{n}}_1 = (\vec{\mathbf{x}}_* - \vec{\mathbf{x}}_1) \times (\vec{\mathbf{x}}_2 - \vec{\mathbf{x}}_*).$$

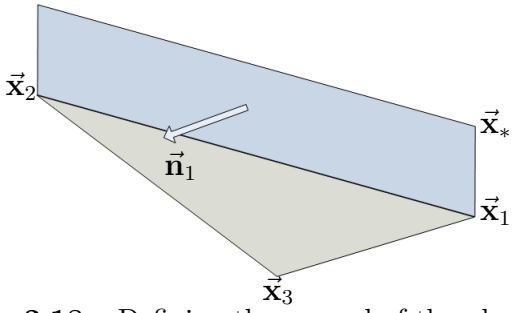


Figure 2.18 – Defining the normal of the plane PL_1.

The normals $\vec{\mathbf{n}}_2$ and $\vec{\mathbf{n}}_3$ can be found in a similar fashion. This then leads to the scan conversion which defines the domain Ω_{x_1,x_2,x_3}, a subset of the domain Ω_g, in which the points possibly closet to the face $x_1x_2x_3$ reside,

$$\Omega_{x_1x_2x_3} = \{\vec{\mathbf{x}} \in \Omega_g \,|\, (\vec{\mathbf{x}}-\vec{\mathbf{x}}_1) \cdot (\vec{\mathbf{n}}_{1,2}) \geq 0, (\vec{\mathbf{x}}-\vec{\mathbf{x}}_2) \cdot (\vec{\mathbf{n}}_{2,3}) \geq 0, (\vec{\mathbf{x}}-\vec{\mathbf{x}}_3) \cdot (\vec{\mathbf{n}}_{3,1}) \geq 0\}.$$

For any point/vertex $\vec{\mathbf{x}} \in \Omega_{x_1x_2x_3}$ the signed distance d from $\vec{\mathbf{x}}$ to the face can be found by,

$$d = (\vec{\mathbf{x}} - \vec{\mathbf{x}}_1) \cdot (\vec{\mathbf{n}}).$$

Case 2, edge:

Let $\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2, \vec{\mathbf{x}}_3$ and $\vec{\mathbf{x}}_4$ define two triangular patches of γ which share an edge $L_{1,2}$ and have normal vectors $\vec{\mathbf{n}}_{1,2,3}$ and $\vec{\mathbf{n}}_{1,4,2}$, as illustrated in Figure 2.19.

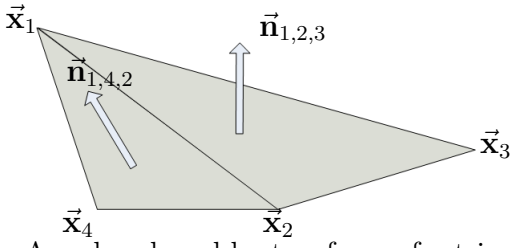


Figure 2.19 – An edge shared by two faces of a triagulated surface.

The grid points possibly closest to the edge $L_{1,2}$ are enclosed by four planes, PL_1, PL_2, PL_3 and PL_4, as illustrated in Figure 2.20.

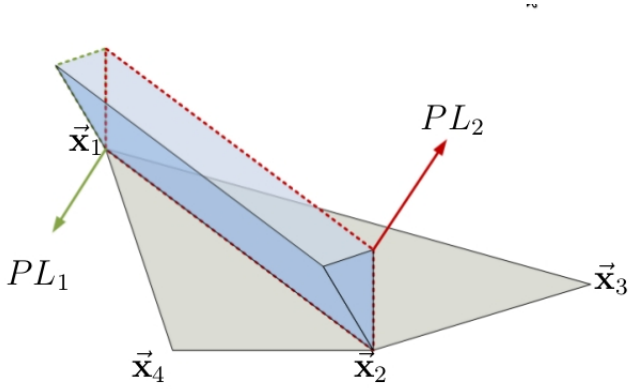


Figure 2.20 – The planes enclosing the points possibly closet to an edge of a triangulated surface.

As before, only three vertices on the desired plane that comply with the right hand rule, are required to calculate a normal vector and sufficiently describe each plane. Define vertices $\vec{\mathbf{x}}_{1,2}$ and $\vec{\mathbf{x}}_{1,4}$ as

$$\vec{\mathbf{x}}_{1,3} = \vec{\mathbf{x}}_1 + \vec{\mathbf{n}}_{1,2,3},$$
$$\vec{\mathbf{x}}_{1,4} = \vec{\mathbf{x}}_1 + \vec{\mathbf{n}}_{1,4,2}.$$

The respective normals for planes PL_1 and PL_2 can be found by

$$\vec{\mathbf{n}}_1 = (\vec{\mathbf{x}}_{1,3} - \vec{\mathbf{x}}_1) \times (\vec{\mathbf{x}}_{1,4} - \vec{\mathbf{x}}_{1,3})$$
$$\vec{\mathbf{n}}_2 = (\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_{1,3}) \times (\vec{\mathbf{x}}_2 - \vec{\mathbf{x}}_1).$$

See Figure 2.21 for an illustration of the calculated normals. Similarly, the respective normals for PL_3 and PL_4 can be found by defining $\vec{\mathbf{x}}_{2,3}$ and $\vec{\mathbf{x}}_{2,4}$.

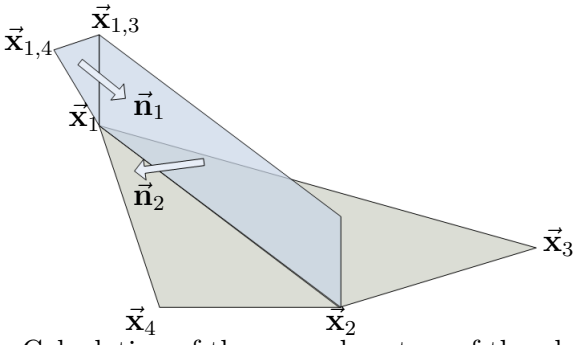


Figure 2.21 – Calculation of the normal vectors of the planes for scan conversion of points closest to an edge.

The scan conversion for defining the domain Ω_{x_1,x_2}, which contains the points possibly closest to the edge $L_{1,2}$, follows as,

$$\Omega_{x_1,x_2} = \{\vec{\mathbf{x}} \in \Omega_g \,|\, (\vec{\mathbf{x}} - \vec{\mathbf{x}}_1) \cdot (\vec{\mathbf{n}}_1) \geq 0, (\vec{\mathbf{x}} - \vec{\mathbf{x}}_1) \cdot (\vec{\mathbf{n}}_2) \geq 0, (\vec{\mathbf{x}} - \vec{\mathbf{x}}_2) \cdot (\vec{\mathbf{n}}_3) \geq 0,$$
$$(\vec{\mathbf{x}} - \vec{\mathbf{x}}_2) \cdot (\vec{\mathbf{n}}_4) \geq 0\}.$$

The distance d from any point $\vec{\mathbf{x}} \in \Omega_{x_1,x_2}$ to the edge $L_{1,2}$ can be found by,

$$d = \frac{|(\vec{\mathbf{x}} - \vec{\mathbf{x}}_1) \times (\vec{\mathbf{x}}_2 - \vec{\mathbf{x}})|}{|\vec{\mathbf{x}}_2 - \vec{\mathbf{x}}_1|}.$$

Note also that if the local surface at the edge is convex and the outside angle between the two adjacent patches is greater than π, the polyhedron will only contain points outside the surface and therefore the distance is signed positive, or vice versa. This can quickly be determined by comparing the distance $d_1 = |\vec{\mathbf{x}}_3 - \vec{\mathbf{x}}_4|$ and $d_2 = |(\vec{\mathbf{x}}_3 + \vec{\mathbf{n}}_{1,2,3}) - (\vec{\mathbf{x}}_4 + \vec{\mathbf{n}}_{1,4,2})|$. If $d_2 > d_1$ then the outside angle between the two patches is greater than π and the distances is signed positive, otherwise they are signed negative. More detail of the sign calculation can be found in Appendix A.

Case 3, vertex:

In the case of a vertex, $\vec{\mathbf{x}}_1$, assume that the vertex is common to $n \geq 3$ patches of γ with normal vectors $\vec{\mathbf{n}}_1, \vec{\mathbf{n}}_2, \ldots, \vec{\mathbf{n}}_n$, as illustrated in Figure 2.22.

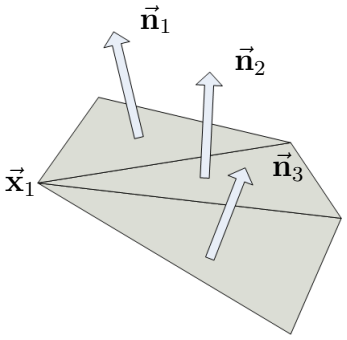


Figure 2.22 – A vertex of a triangulated surface, shared by a number of faces.

The points possibly closest to $\vec{\mathbf{x}}_1$ are enclosed by the r planes, $PL_{1,2}, PL_{2,3}, \ldots, PL_{n,1}$, as illustrated in Figure 2.23.

The vertex $\vec{\mathbf{x}}_1$ lies on each of the planes. To find the normals, define the following n points

$$\vec{\mathbf{x}}_{1,i} = \vec{\mathbf{x}}_1 + \vec{\mathbf{n}}_i, \qquad 1 \geq i \leq r$$

.

The normal of plane $PL_{i,i+1}$ can now be found by,

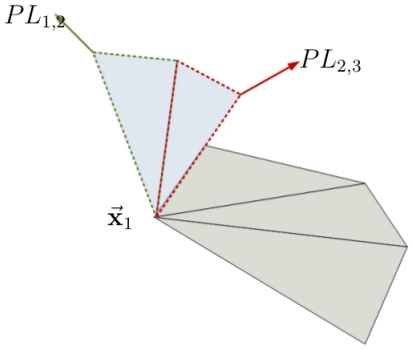


Figure 2.23 – Planes enclosing the points possibly closest to a vertex of a trianulated surface.

$$\vec{\mathbf{n}}_{i,i+1} = (\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_{1,i}) \times (\vec{\mathbf{x}}_{1,i+1} - \vec{\mathbf{x}}_1), \qquad 1 \leq i \leq n,$$

as illustrated in Figure 2.24.

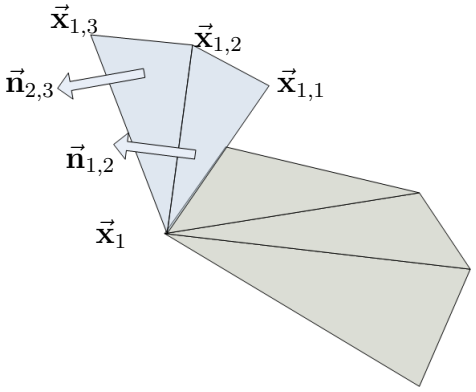


Figure 2.24 – Calculation of the normal vectors, of the planes enclosing points possibly closest to a vertex.

In the case of $i = r$, the value for $i + 1$ is found by $\mod(i + 1, r) = 1$. This leads to the scan conversion for $\vec{\mathbf{x}}_1$.

$$\Omega_{x_1} = \{\vec{\mathbf{x}} \in \Omega_g \,|\, (\vec{\mathbf{x}} - \vec{\mathbf{x}}_1) \cdot (\vec{\mathbf{n}}_{i,i+1}) > 0, \quad \forall\ 1 \leq i \leq r\}.$$

The distance d from $\vec{\mathbf{x}}$ to $\vec{\mathbf{x}}_1$ is simply found as the Euclidean norm,

$$d = |\vec{\mathbf{x}} - \vec{\mathbf{x}}_1|.$$

In order to determine whether the polyhedron lies inside or outside the surface, define a normal $\vec{\mathbf{n}}_{vert}$ on the vertex as the avarage of all the face normals, $\vec{\mathbf{n}}_1, \vec{\mathbf{n}}_2, \ldots, \vec{\mathbf{n}}_n$. Following a similar procedure as before, the angle between the each of the n faces and the avarage normal, n_{vert}, can be checked by comparing $d_1 = |\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_{2,i}|$ and $d_2 = |(\vec{\mathbf{x}}_1 + \vec{\mathbf{n}}_{vert}) - (\vec{\mathbf{x}}_{2,n} + \vec{\mathbf{x}}_i)|$, where $\vec{\mathbf{x}}_{2,i}$ is a vertex on the i'th patch common to $\vec{\mathbf{x}}_1$. If for each of the n patches, $d_1 < d_2$, the polygon will be outside the surfaces and the distances are signed positive. If for each of the n patches, $d_1 > d_2$, the polygon will be inside the surfaces and the distances are signed negative. More detail can be found in Appendix A.

An illustration of the final algorithm for computing the discrete SDF is illustrated in Figure 2.25.

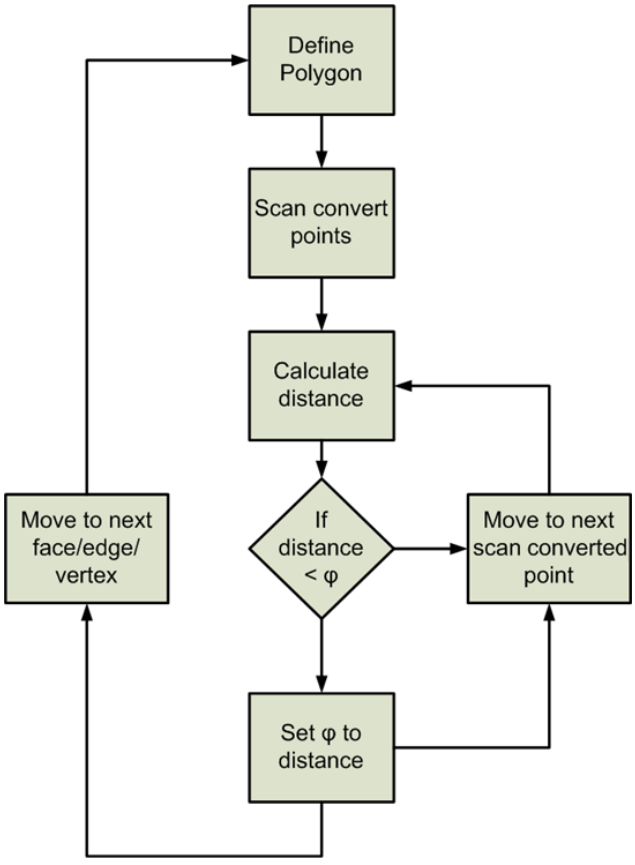


Figure 2.25 – A procedural layout of the algorithm for computing the 3-D SDFs of triangulated surfaces.

2.8 Global perimeters and level set extraction

2.8.1 Introduction

As discussed in Section 2.4.2, the LSM relies on an implicit representation of an interface in order to perform interface propagation. In this section the methods for the evaluation of geometric properties of the implicit interfaces are developed and discussed.

Let γ_s be a propagating 3-D interface at time t. Once the LSM calculations have been performed on the function ϕ_t into which γ_s has been embedded, the result is an implicit representation $\phi^{t+\Delta t}$ and the propagated interface is not yet known explicitly.

Recall the interface $\gamma_s^{t+\Delta t}$ is given by the zero-level set of $\phi_s^{t+\Delta t}$. In order to explicitly define the propagated interface $\gamma_s^{t+\Delta t}$, the zero level set needs to be extracted by means of a contour algorithm.

The explicit interface, however, is not necessarily the objective of an interface evolution procedure, and in some instances, properties such as volume is desired, irrespective of whether the interface is known explicitly or not. One of the major advantages of the LSM is its ability to provide fast and effective calculation of global properties such as the surface area, curvature and volume enclosed by the interface. Since the only properties of interest during Q-1D IB simulation is surface area and volume, these will be the only two geometric properties considered.

Let a 3-D interface γ_s be represented by the zero level set of ϕ_s. The surface area A_s and the internal volume Ψ_s of γ_s is found by,

$$A_s = \int_\Omega \delta(\phi) \tag{2.8.1}$$

$$\Psi_s = \int_\Omega H(\phi) \tag{2.8.2}$$

Where the Heavy-side function H is defined as,

$$H(x) = \begin{cases} 0, & \text{if } x > 0 \\ 1, & \text{if } x \leq 0 \end{cases}. \tag{2.8.3}$$

The Dirac-delta function can be seen as the spatial differentiation of the Heavy-side function,

$$\delta(x) = \frac{dH(x)}{dx}$$

and becomes,

$$\delta(x) = \begin{cases} 0, & \text{if } |x| > 0 \\ 1, & \text{if } x = 0. \end{cases} \tag{2.8.4}$$

The discretization of the Dirac-delta and Heavy-side functions to a discrete grid, is a well-researched subject and covered substantially in the literature. A popular approach was introduced by Osher and Fedkiw [32] where the values of the Dirac-delta and Heavy-side functions are based solely on the distance from the interface, or the local value of ϕ. Tomnberg and Enquist [33] showed this type of regularization causes an approximation error of order $O(1)$. In order to avoid this, Enquist et al. [34] proposed scaling the Dirac-delta according to the angle of the local gradient of ϕ with respect to the grid orientation. Further notable work on the Dirac-delta discretization was carried out by Smereka [35].

In his thesis Cavalinni [12] found the above mentioned techniques to be sensitive to small interface perturbations with respect to the grid and that they return irregular area calculations, making them unsuitable for coupling the LSM with internal ballistics solvers for SRM simulation. Instead, Cavalinni employed isosurface extraction and performed a numerical quadrature of the extracted interface as proposed by Min and Gibou [36].

In the following sections two methods of performing the surface integration are considered. The marching cubes method and the problem it suffers with regards to ambiguity is discussed, after which the Monte Carlo (MC) integration techniques are discussed and their use as the preferred method for calculation of area and volume is motivated.

2.8.2 Marching cubes method

The most popular form of level set or, in the case of 3-D data, isosurface extraction is the marching cubes method that was introduced by Lorensen [37] in 1987. There are a number of variations of the algorithm as well as some alternatives that include marching tetrahedron methods by Treece [38], marching diamonds by Anderson [39] and adaptive skeleton climbing of Poston [40]. For a good overview and description of these marching cube methods the reader is refered to Mielack [41]. For the current work, the built-in MATLAB© function `isosurface()`, which is also based on marching cubes, will be used as a standard isosurface extraction tool. The function returns a triangulated surface γ_{tri}, similar to the STL file format discussed in Section 2.7.2, as approximation to the evolved interface.

In order to extract a level set or isosurface, the marching cubes algorithm considers each 8 grid point cell, also known as a voxel, individualy and decides in which topological state the cell is with respect to the chosen isovalue. Each of the 8 neighbours can be categorized as above or below the chosen isovalue. This leads to $2^8 = 256$ possible states of each cube. This can be reduced to 14 unique cases with respect to rotation, reflection, and inversion. The form of these topological cases are dependent on the points of intersection of the isosurface with the edges of the cubes. Figure 2.26 illustrates the 14 possible states of a voxel, where the highlighted vertices are inside the interface.

Given a chosen rational number *isovalue* and an edge, say $L_{1,2}$, connecting the grid points $\vec{\mathbf{x}}_1$, and $\vec{\mathbf{x}}_2$, with $\phi(\vec{\mathbf{x}}_1) < isovalue$ and $\phi(\vec{\mathbf{x}}_2) > isovalue$, the point, $\vec{\mathbf{x}}$, at which the *isovalue*-isosurface cuts the edge $L_{1,2}$ can be found by linear interpolation:

$$\vec{\mathbf{x}} = \vec{\mathbf{x}}_1 + (\vec{\mathbf{x}}_2 - \vec{\mathbf{x}}_1) \cdot \left(\frac{isovalue - \phi(\vec{\mathbf{x}}_1)}{\phi(\vec{\mathbf{x}}_2) - \phi(\vec{\mathbf{x}}_1)}\right) \tag{2.8.5}$$

The voxel is divided into regions by triangular patches that intersect the edges of the voxel at these interpolated points. This is done for each voxel of the grid and the resulting triangles are added together to make up the final triangulated isosurface. The marching cubes algorithm is, however, not assured of returning

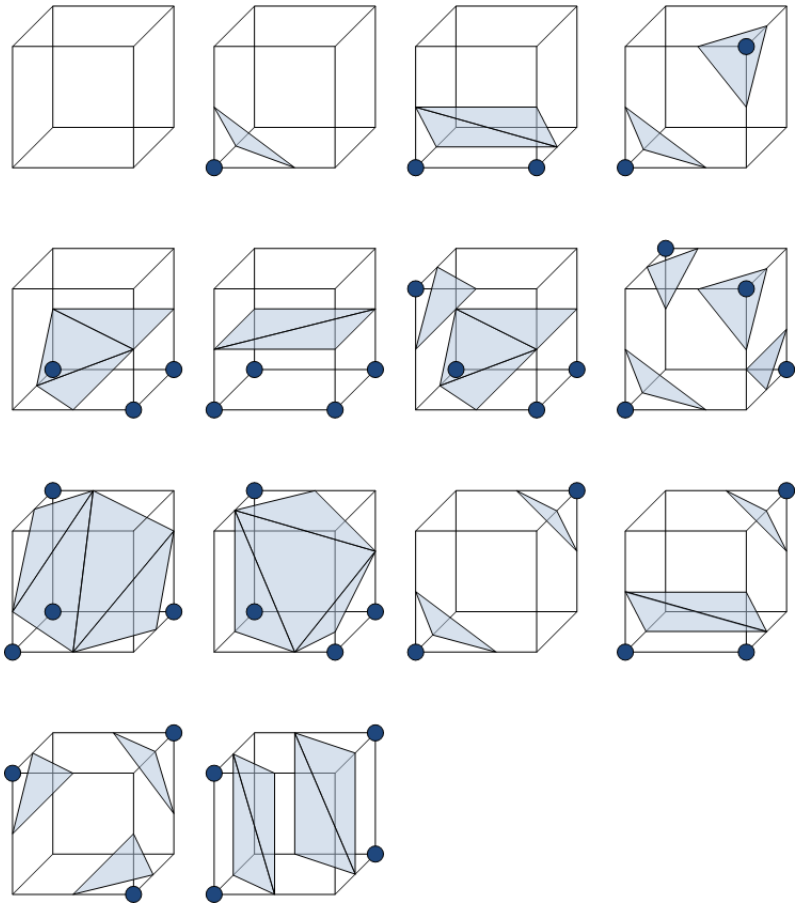


Figure 2.26 – 14 Unique topological states of the marching cubes method.

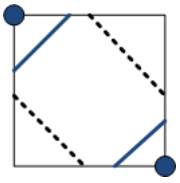


Figure 2.27 – 2-D ambiguity of the marching cubes method. Two cases that result in the same SDF values on 4 vertices of a square.

physically correct surfaces since there are a number of ambiguous cases. These ambiguities arise from the 2-D situation illustrated in Figure 2.27.

The two marked vertices of the square are located inside the isosurface and the two unmarked vertices are located on the outside. This could be a result of either one of two cases, as represented by the solid and dashed lines. Each case of Figure 2.26 that contains the ambiguity on one or more of its faces will itself be ambiguous. It can also result in a topologically incorrect surface that is not closed, where closed refers to the same definition of closed as the one given in Section 2.7.1.

A method for deciding which case to follow was developed by Nielson and Hamann [42], known as the asymptotic decider. Although this method cannot guarantee the physically correct choice be made, it provides a method for consistently making choices on these ambiguities and gives the marching cubes method the ability to consistently return closed surfaces.

2.8.2.1 Calculation of geometric properties using the marching cubes method

Once the marching cubes algorithm returns a triangulated surface $\gamma_{s_{tri}}$ as approximation to γ_s, the surface area of $\gamma_{s_{tri}}$, $A_{s_{tri}}$ can be calculated by summing each individual triangle surface area A_{tri^i},

$$A_{s_{tri}} = \sum_{tri^i \in \gamma_{s_{tri}}} A_{tri^i}. \tag{2.8.6}$$

The surface area of a triangle tri^i given by vertices $\vec{\mathbf{v}}_1, \vec{\mathbf{v}}_2$ and $\vec{\mathbf{v}}_3$ is found by

$$A_{tri^i} = |(\vec{\mathbf{v}}_2 - \vec{\mathbf{v}}_1) \times (\vec{\mathbf{v}}_3 - \vec{\mathbf{v}}_2)|/2. \tag{2.8.7}$$

The volume fraction inside the interface of each voxel cut by the isosurface can also be computed for each of the 14 cases.

Let vox be a voxel of the domain, Ω_g and define a modified Heavy-side function H_{vox}, such that it returns the correct fraction for each voxel cut by the isosurface. Each voxel of Ω_g that contains only negative/positive vertices will be completely inside/outside the interface and its volume fraction will be $H_{vox} = 1$ if inside, or $H_{vox} = 0$ if outside. Now the volume Ψ_s is found by,

$$\Psi_s = \sum_{vox \in \Omega_g} H(vox) \cdot \Psi_{vox}, \tag{2.8.8}$$

where Ψ_{vox} denotes the total volume of the voxel vox. Equations 2.8.6 and 2.8.8 return accurate results for most interfaces. However, in the case of two interface sections orientated close to parallel to each other, at a distance less than the grid spacing of the domain Ω_g apart, the marching cubes method may introduce significant errors to the surface area parameter due to non-physical choices in ambiguous voxels. An example of two circles with radii $r1$ and $r2$, with $|r1 - r2|$ less than the grid spacing, is illustrated in Figure 2.28. More evidence of the effect of these errors on the SRM simulation is given in Section 4.1.2.

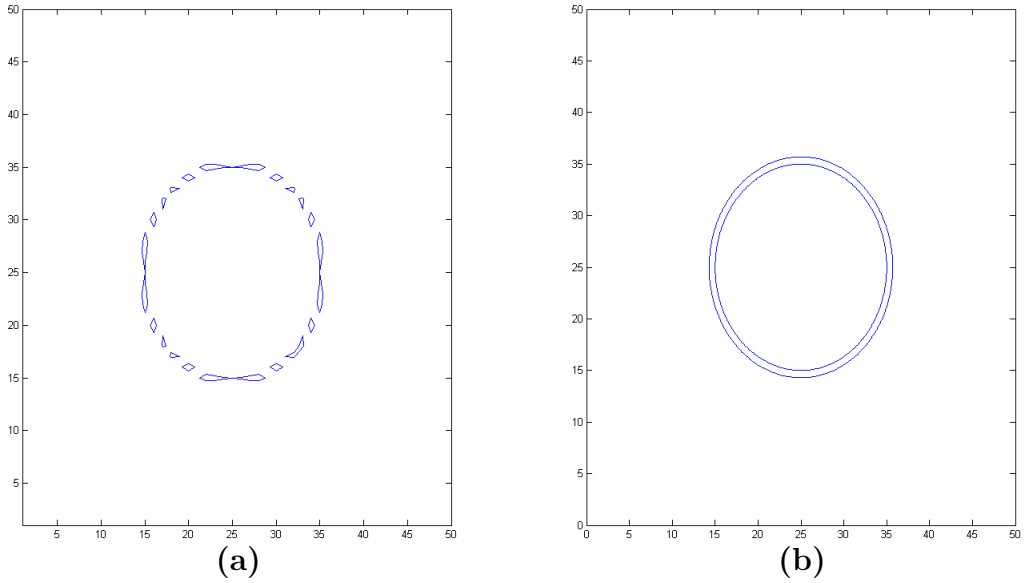


Figure 2.28 – The result of the marching cubes algorithm (a) compared to the physical solution (b).

2.8.2.2 Burn-out calculation using marching cube type numerical integration

A problem that is specific to SRM simulation is numerical integration of the interface during the burnout phase of the motor, during which certain segments of the grain are completely burnt away, exposing the motor casing or insulation and so not contributing to the physical burning surface. The calculation of the burning surface during the burn-out phase is performed in a number of

ways in the literature. For instance, Cavalinni's approach [12] is to only include segments that do not contain points on the outside of the casing interface. This makes the calculated burning surface a jagged function and requires a large amount of grid refinement to smooth the function. The illustrations in Figure 2.29 shows how a 2-D analogue situation with the burning surface interface propogating toward the motor casing. The linear patches that make up the interface representation are deleted sequentially, causing the jagged effect in the burning area profile.

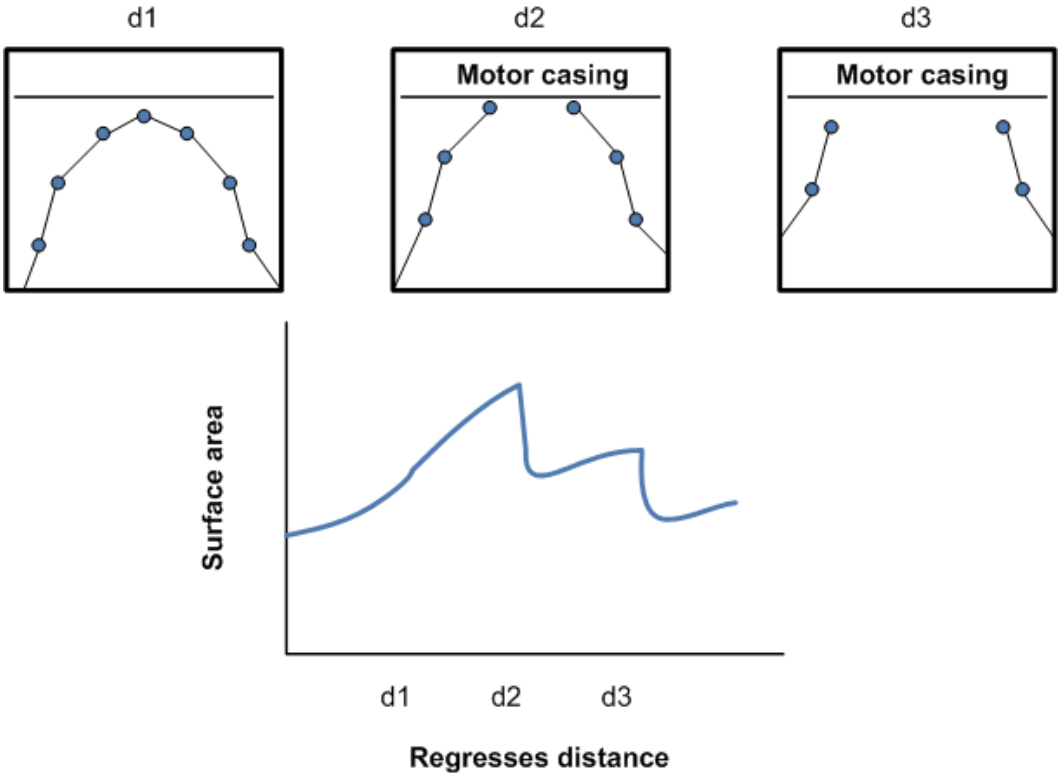


Figure 2.29 – A jagged area profile due to burnout procedure of the marching cube approuch.

The ROCGRAIN code [43] follows a similar procedure but instead of extracting a 3-D surface, 2-D slices are made perpendicular to the motor axis. The 2-D contour-lenght multiplied by the distance between consecutive slices are used in order to approximate the surface. The program calculates any inter-

section points of the burn surface and the motor casing contours which negates the need for a great deal of grid-refinement. In cases where the burning surface is orientated with a normal vector close to parallel to the motor axis, A 2-D slice approximation could, however, cause the burn surface to be substantially under or over approximated. Figure 2.30 gives a 2-D example of approximaiton errors introduced by approximating a 3-D surface by 2-D slices, the blue line represents the actual burning surface and the red line shows the slice approximation. The surface section between points x_2 and x_3 is heavily under approximated due to the acute surface angle.

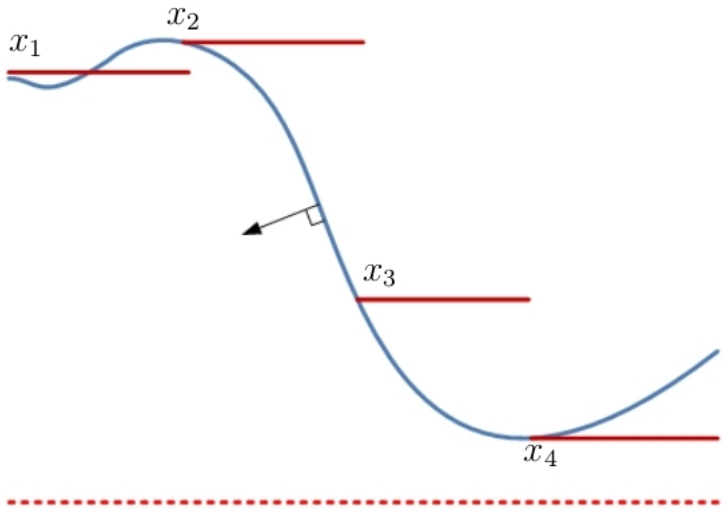


Figure 2.30 – 2-D slice approximaiton errors introduced for surfaces with a normal close to perpendicular to the motor axis.

2.8.3 Monte-Carlo integration

The class of statistical methods known as Monte Carlo (MC) methods were introduced by Ulam and Metropolis [44]. Three main branches to which MC methods are applied are: optimization, sample generation from non-uniform distributions and numerical integration, the latter being applicable to the area and volume integrations for A_s and Ψ_s calculation. For a broad overview of the techniques, the reader is referred to the text book of Rubinstein [45]. A

slightly less in-depth, yet insightful discussion on MC integration, in particular, is found in the notes of Edwards [46].

Monte Carlo integration for area or volume calculations of shapes inscribed to 2-D or 3-D domains with known areas or volumes follows a basic procedure. First the domain is cluttered with uniformly distributed random points that will be refered to as MC-points. The number of MC-points inside the shape is counted and the ratio of points inside the shape to total number of points is multiplied with the domain area or volume to approximate the area or volume of the inscribed shape. In order to perform unbiased approximations, it is vital to draw random numbers from a uniform distribution. The methods of generating random numbers will not be discussed in detail, however, some notes and examples of random number generators are given in Section 2.9. The built-in MATLAB© function `rand()` is utilized for random number generation throughout the presented work.

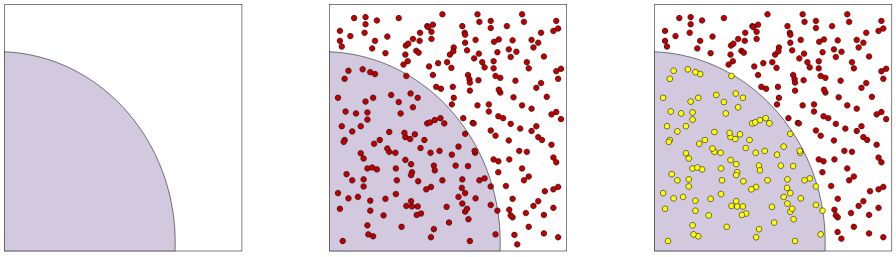


Figure 2.31 – MC integration of a quater circle inscribed to a unit square

In Figure 2.31, the shaded segment is inscribed to a unit square and the area A_{shade} can be approximated as follows:

250 uniformly distributed random points are scattered across the unit square and 111 of them are found to lie inside the shaded section. Thus,

$$A_{shade} = \frac{111}{250} = 0.444\,\text{units}^2.$$

The shaded area is a quater circle of radius 0.75 with an exact area of 0.4418 units^2. The MC area integration approximated the area to within 0.5% accuracy with 250 sample points.

The arc length of a shape in 2-D, or the surface area of a shape in 3-D, can also be approximated by a thin envelope around the interface by finding the

surface area, or volume, of the envelope and dividing by its width. In Figure 2.32, a thin envelope is placed around the perimiter of the inscribed area of Figure 2.31.

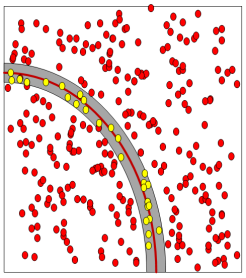


Figure 2.32 – Thin evelope approximation of arc length.

2.8.3.1 MC integration of implicit surfaces

Effective determination of whether an MC-point is located inside or outside a surface is vital to the MC integration procedure. In the case of LSM representation of 3-D interfaces, a surface is implicitly defined as a zero levelset of a SDF function and means all that is necessary to determine whether a random point is inside or outside the surface, is checking the sign of the SDF at the local position.

Since the MC-points are not co-located with the grid points on which the SDF is defined, some spatial interpolation is necessary to determine the local SDF value. The interpolation scheme used to interpolate the random points to the SDF determines the order of accuracy with which the front is approximated if the statistical error is disregarded. Yang et al. [47] performed a case study of twelve different spatial interpolation schemes of varying order and the interested reader is referred to the references therein for details about the various schemes. The case problem is not directly applicable to the interpolation of SDFs for MC integration of the represented surface, but the conclusions they arrive at are applicable to the manner in which a choice of interpolation scheme is made.

In their conclusions, Yang et al. [47] states that the performance of any one specific interpolation scheme is subject to the characteristics of the data and

the purpose for which it is applied, i.e. a relatively best interpolation scheme is chosen based on the characteristics of the specific data and the characteristics of the interpolation schemes available.

Tri-linear approximation gives a first order accurate representation of the surface, similar in accuracy to the marching cubes surface extraction. The fact that the initial surface in STL format is composed of linear patches added to the fact that the surface may contain sharp corners, makes trilinear interpolation a suitable scheme to employ for parameter calculation in SRM burn back simulation. This is true since the higher order interpolation schemes might smooth the sharp corners or discontinuities of the surface. A description of tri-linear spatial interpolation follows.

Consider a voxel, $vox = \{\vec{\mathbf{x}}_{0,0,0}, \vec{\mathbf{x}}_{0,0,1}, \vec{\mathbf{x}}_{0,1,0}, \vec{\mathbf{x}}_{0,1,1}, \vec{\mathbf{x}}_{1,0,0}, \vec{\mathbf{x}}_{1,0,1}, \vec{\mathbf{x}}_{1,1,0}, \vec{\mathbf{x}}_{1,1,1}\}$, where $\vec{\mathbf{x}}_{i,j,k} = (x_i, y_j, z_k)$, and a point $\vec{\mathbf{x}_p} = (x_p, y_p, z_p)$, with $x_0 \leq x_p \leq x_1$, $y_0 \leq y_p \leq y_1$ and $z_0 \leq z_p \leq z_1$, as illustrated in Figure 2.33.

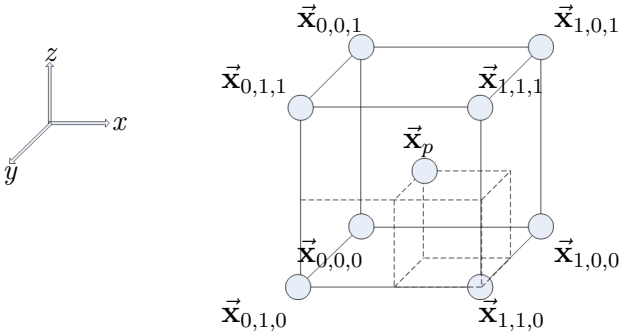


Figure 2.33 – A point inside a voxel

Let the function ϕ be defined at each of the 8 vertices of the voxel and denote the local value of the implicit representation $\phi(\vec{\mathbf{x}}_{i,j,k})$, by $\phi_{i,j,k}$. The value within the voxel at $\vec{\mathbf{x}}_p$, can be interpolated as follows. Let,

$$dx = (x_p - x_0)/(x_1 - x_0), \tag{2.8.9}$$

$$dy = (y_p - y_0)/(y_1 - y_0), \tag{2.8.10}$$

$$dz = (z_p - z_0)/(z_1 - z_0). \tag{2.8.11}$$

First the values are interpolated in the $y-$direction,

$$\phi_{0,0} = (1 - dx)\phi_{0,0,0} + dx\phi_{1,0,0}, \tag{2.8.12}$$

$$\phi_{0,1} = (1 - dx)\phi_{0,0,1} + dx\phi_{1,0,1}, \tag{2.8.13}$$

$$\phi_{1,0} = (1 - dx)\phi_{0,1,0} + dx\phi_{1,1,0}, \tag{2.8.14}$$

$$\phi_{1,1} = (1 - dx)\phi_{0,1,1} + dx\phi_{1,1,1}, \tag{2.8.15}$$

as in Figure 2.34.

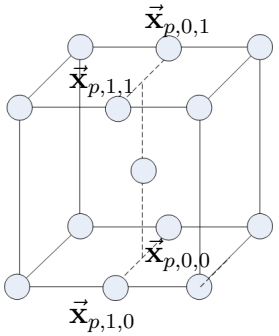


Figure 2.34 – x-dimension interpolation

The calculated values all lie on the plain $\{x = x_p\}$. Next the values are interpolated in the x-direction, as in Figure 2.35,

$$\phi_{p,p,0} = (1 - dy)\phi_{p,0,0} + dy\phi_{p,1,0}, \tag{2.8.16}$$

$$\phi_{p,p,1} = (1 - dy)\phi_{p,0,1} + dy\phi_{p,1,1}. \tag{2.8.17}$$

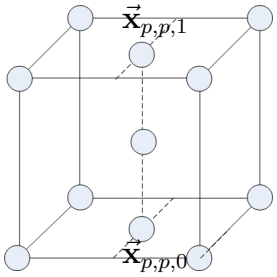


Figure 2.35 – y-dimension interpolation

Finally the values are interpolated on the line $\{x = x_p, y = y_p\}$, as in Figure 2.36,

$$\phi_p = \phi_{p,p,0} \cdot (1 - dz) + \phi_{p,p,1} \cdot dz.$$

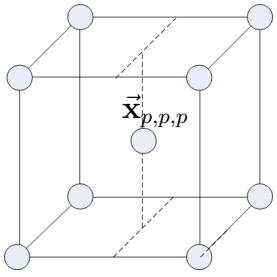


Figure 2.36 – z-dimension interpolation

The tri-linear interpolation scheme can now be employed to find the interpolated values of the discrete ϕ function of an interface at random points in the domain.

The MC integration of the volume parameter Ψ_s of the interface that is advanced by the LSM is done by finding the number of random points that interpolate to a negative value with respect to the ϕ function and dividing by the total number of random points used. Algorithms for the calculation of A_s and Ψ_s can now be developed.

2.8.3.2 Volume integration of implicit surfaces

Let ϕ be an implicit representation of a 3D interface γ_s, defined on the discrete grid Ω_g. An algorithm for computing Ψ_s, the volume enclosed by the interface, can now be developed as follows.

First N random MC-points $\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2, \ldots, \vec{\mathbf{x}}_N$ are generated. Trilinear interpolation is used to interpolate each point $\vec{\mathbf{x}}_i$ to ϕ in order to find $\phi(\vec{\mathbf{x}}_i)$. The points inside the interface are then summed to find the ratio N_{inside}/N. For the purpose of calculating the number of inside points, the result of the Heavy-side function, equation (2.8.3), applied to all the MC-points, is summed:

$$N_{inside} = \sum_{i=1}^{N} H(\phi(\vec{\mathbf{x}}_i)), \tag{2.8.18}$$

The volume Ψ_s inside γ_s can now be integrated as,

$$\Psi_s = \frac{N_{inside}}{N} \Psi_{\Omega_g}. \tag{2.8.19}$$

where Ψ_{Ω_g} refers to the volume of the domain.

Consider the grain burning surface γ_{star}, as illustrated from various angles in Figure 2.37.

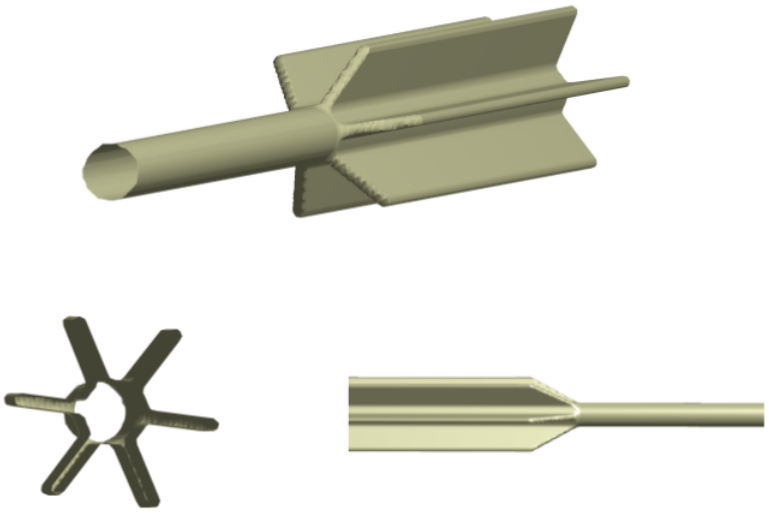


Figure 2.37 – An example of the grain burning surface, γ_{star}.

The MC-points included by the Heavy-side function to the volume calculation for γ_{star} is illustrated in Figure 2.38.

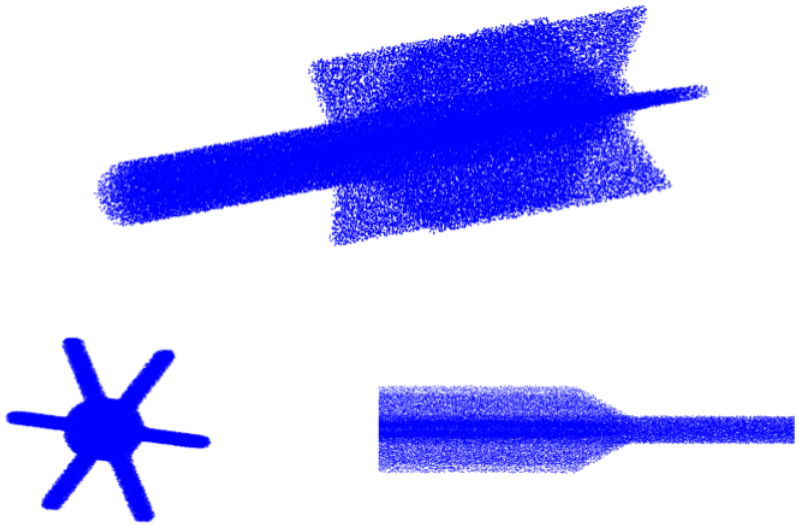


Figure 2.38 – The MC-points used for volume calculation of γ_{star}.

2.8.3.3 Area integration of implicit surfaces

For the purpose of calculating the surface area of a 3-D interface, the volume of a thin envelope is required. For the purpose of the integration of Ψ_s, a MC-point $\vec{\mathbf{x}}$ was determined to be inside S if $\phi(\vec{\mathbf{x}}) \leq 0$, since the zero contour line represents the interface γ_s. Conveniently, the characteristics of the implicit representation of a front allows a thin envelope with a width $w_{env} \in \mathbb{R}$ around γ_s to be found by considering the volume inside the two isosurfaces of value $\frac{w_{env}}{2}$ and $\frac{-w_{env}}{2}$. This leads to the algorithm for computing A_s, by employing MC integration to find the volume of the thin envelope around γ_s.

Again N random MC-points are generated and interpolated to ϕ_s. A modified Heavyside-function, H_{env} is employed to select the points inside a thin envelope of width w_{env},

$$H_{env}(y) = \begin{cases} 1 \text{ if } |y| \leq w_{env}/2 \\ 0 \text{ if } |y| > w_{env}/2 \end{cases} . \tag{2.8.20}$$

As was the case with the heavy-side function for volume integration, equation (2.8.20) is used to calculate N_{inside},

$$N_{inside} = \sum_{i=1}^{N} H_{env}(\phi_s(\vec{\mathbf{x}}_i)). \tag{2.8.21}$$

The surface area A_s is calculated as the volume of the thin envelope divided by its width,

$$A_s = \left(\frac{N_{inside}}{N}\right)\left(\frac{\Psi_s}{w_{env}}\right). \tag{2.8.22}$$

The MC-points included in the area calculation for γ_{star}, as illustrated in Figure 2.37, are shown in Figure 2.39.

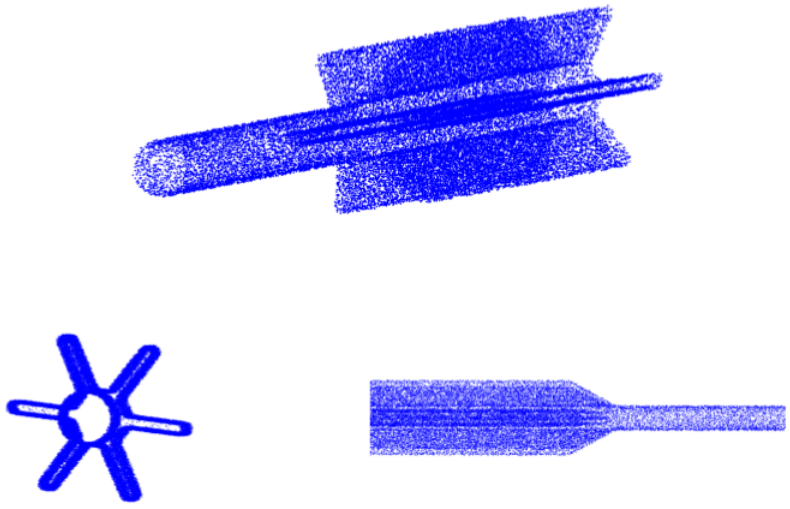


Figure 2.39 – The MC-points used for Area calculation of γ_{star}.

Figure 2.40 shows the MC-points, illustrated in Figure 2.39, all lies within the $w_{env}/2$ and $-w_{env}/2$ isosurfaces of ϕ_{star}, where ϕ_{star} is the implicit representation of γ_{star}.

2.8.3.4 Burn-out calculation using MC integration

Let γ_c be the interface that defines the motor casing. The burn-out calculation is handled elegantly with the MC integration method. The random MC-points

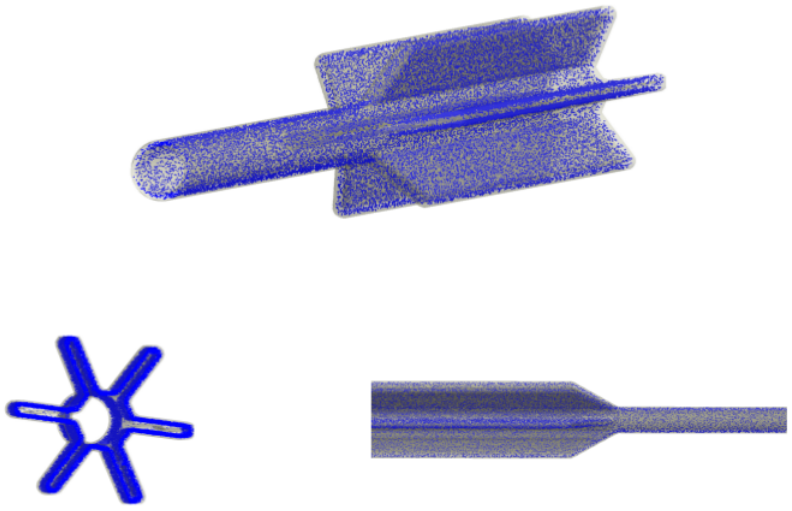


Figure 2.40 – The MC-points and the $w_{env}/2$, and $-w_{env}/2$ isosurfaces of ϕ_{star}.

$\vec{\mathbf{x}}_i$ used for the integration are interpolated to both ϕ_s and ϕ_c and discarded from the calculations of Ψ_s and A_s if $\phi_c(\vec{\mathbf{x}}_i) > 0$, i.e. outside the motor casing. The highlighted MC-points in Figure 2.41 are excluded from the MC integration.

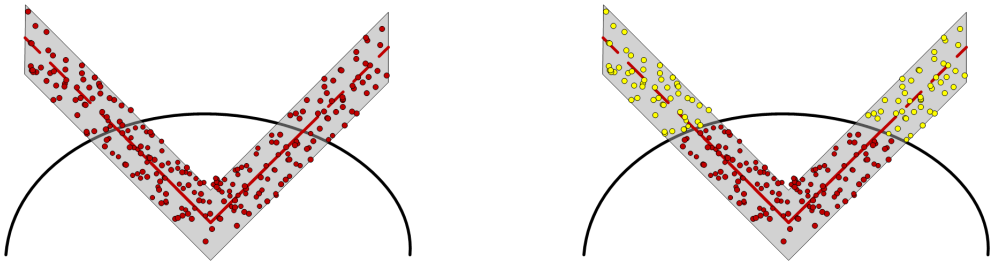


Figure 2.41 – Burn out calculations using MC integration.

An extra check is added to the H functions used for the calculation of N_{inside}. Equation (2.8.3) becomes,

$$H'(y_1, y_2) = \begin{cases} 1 \textbf{ if } y_1 \leq 0\,;\, y_2 \leq 0 \\ 0 \textbf{ if } y_1 > 0 \\ 0 \textbf{ if } y_2 > 0 \end{cases}, \tag{2.8.23}$$

and similarly equation (2.8.20) becomes,

$$H'_{env}(y_1, y_2) = \begin{cases} 1 \textbf{ if } |y_1| \leq w_{env}/2\,;\, y_2 \leq 0 \\ 0 \textbf{ if } |y_1| > w_{env}/2 \\ 0 \textbf{ if } y_2 > 0 \end{cases} , \tag{2.8.24}$$

Consider again, γ_{star} of Figure 2.37. Let the surface be at a regressed state such that the interface has moved beyond the interface that represents the casing of the motor, as illustrated in Figure 2.42. The interface representing the grain perforation inside the motor casing is also illustrated.

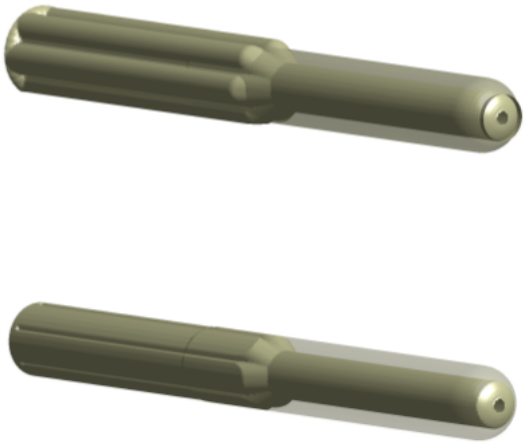


Figure 2.42 – The regressed interface and grain perforation of γ_{star} at a regressed state.

Figure 2.43 shows the MC-points that fall within a thin envelope around the regressed surface, then highlights the points that fall outside the casing and finally shows only the points that contribute to the partially burnt out burning surface.

2.8.3.5 Geometric value calculation for SRM simulation

In order to evaluate the grain burning area and chamber volume, the modified H functions of the previous section are employed. The rest of the integration

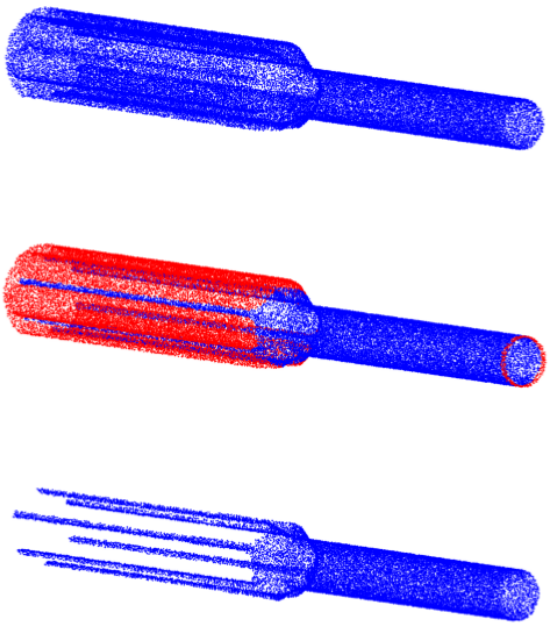


Figure 2.43 – Above: The MC-points that fall within the thin enevelope of the regressed interface. Center: Points that lie outside (highlighted) the casing interface are discarded. Below: The final MC-points used for integration of the physical burning surface approximation.

procedures remains unchanged. The entire algorithm for calculating the burning surface A_s and volume Ψ_s, is summmarized in Table 2.3 and Table 2.4, respectively.

Table 2.3 – An algorithm for the MC integration of the burning surface A_s of an SRM.

MC integration:
1.) Generate N random 3-D points $\vec{\mathbf{x}}_i$ for $i = 1, 2, \ldots, N$ such that $\vec{\mathbf{x}}_i \in \Omega \subset \Omega_g$ with $S \subset \Omega$ and Ψ_Ω the volume of Ω.
2.) Find $\phi(\vec{\mathbf{x}}_i)$ and $\phi_c(\vec{\mathbf{x}}_i)$ by tri linear interpolation.
3.) Calculate $N_{inside} = \sum_{i=1}^{N} H_{env}(\phi(\vec{\mathbf{x}}_i, \phi_c(\vec{\mathbf{x}}_i))$ $\Psi_{env} = \frac{N_{inside}}{N}\Psi_\omega$
$A_s = \frac{\Psi_{env}}{w}$

Table 2.4 – An algorithm for the MC integration of the volume Ψ_s of an SRM.

MC integration:
1.) Generate N random 3-D points $\vec{\mathbf{x}}_i$ for $i = 1, 2, \ldots, N$ such that $\vec{\mathbf{x}}_i \in \Omega \subset \Omega_g$ with $S \subset \Omega$ and Ψ_Ω the volume of Ω.
2.) Find $\phi(\vec{\mathbf{x}}_i)$ and $\phi_c(\vec{\mathbf{x}}_i)$ by tri linear interpolation.
3.) Calculate $N_{inside} = \sum_{i=1}^{N} H(\phi(\vec{\mathbf{x}}_i, \phi_c(\vec{\mathbf{x}}_i))$ $\Psi_s = \frac{N_{inside}}{N} \Psi_\omega$

2.8.3.6 Optimization through stratified MC integration and symmetry

The MC integration procedure can be optimized for efficiency by fairly simple techniques. First, stratified sampling can be applied to the integration, i.e. the sampling of smaller sections or strata of the domain. A discrete grid Ω_g is made up of a number of voxels that each can be integrated independently. The advantage lies in the fact that a voxel can be determined to be completely inside or completely outside the interface, i.e. with only negative or positive ϕ values on its eight vertices, and excluded from the MC integrated voxels. Since each point inside the voxel will automatically interpolate to a negative or positive value, the entire voxel can be added to the inside or outside volumes. This reduces the number of voxels that need to be integrated from the order $O(n^3)$ to $O(n)$, which significantly reduces the computational cost of the integration procedure. In Figure 2.44, a 2-D example shows the effect on the number of voxels, or in the 2-D case squares, that needs to be integrated by interpolation of the random MC points.

The stratified MC integration algorithm is illustrated in Figure 2.45. Let each voxel be integrated independently by N_{vox} MC-points if it is found that the minimum ϕ value on the vertices of the voxel is lower than zero, and the

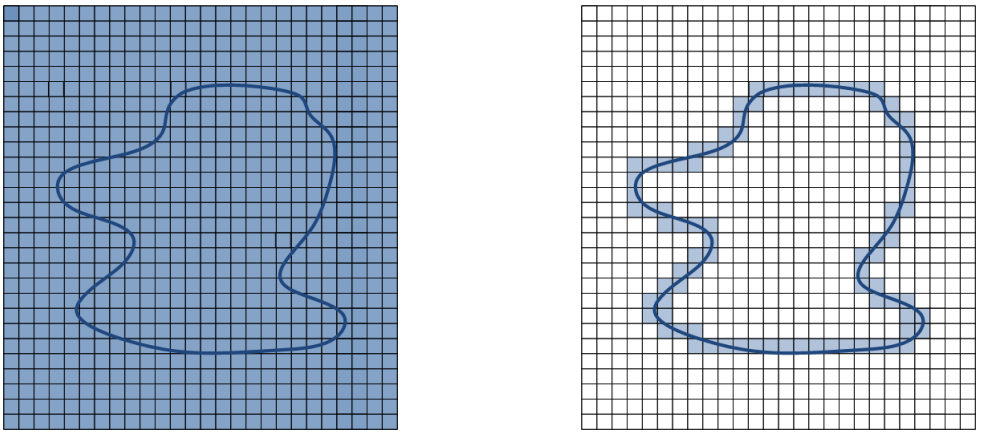


Figure 2.44 – Reduction of the number of integrated cells in a stratafied 2D MC intagration.

maximum value higher than zero. If the minimum value is higher than zero, the entire voxel will lie outside the interface and none of the MC-points inside the specific voxel will contribute to the number of points inside the interface N_{inside} , and therefore no interpolation is necesary. If the maximum value is lower than zero, then the entire voxel will lie inside the interface and therefore the number of points inside the interface can be increased by N_{vox} such that, $N_{inside} = N_{inside} + N_{vox}$, and again no interpolation is necessary.

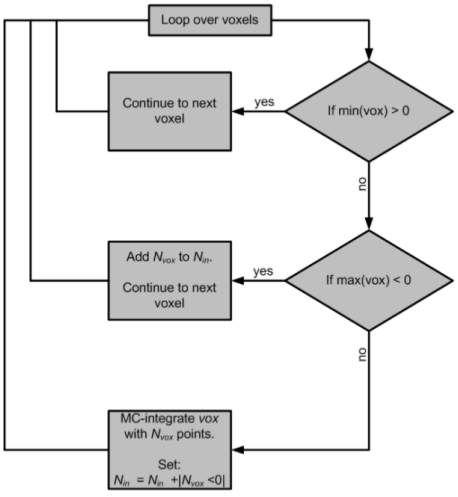


Figure 2.45 – A conceptual illustration of a stratified MC integration algorithm.

Another simple method of reducing the computational complexity of the MC

integration procedure takes advantage of the fact that the interface (burning surface) that is of interest in SRM simulation, only moves in one direction. In other words, once a grid point of Ω_g has been crossed, and lies inside the interface, it will remain inside for the duration of the simulation. This implies that once a cell or a voxel is completely inside the interface, it can be assumed to be inside the interface for all the subsequent MC integrations performed during the simulation, and therefore it is not necessary to check whether the voxel lies within the interface from that point onwards.

The exploitation of symmetry in the grain design, both in the axial and non-axial dimensions, could also, depending on the specific grain design, leads to significant gains in computational effeciency. Note that the axial dimension refers to the dimension parallel to the main motor axis.

Symmetry along the motor's axial dimension: For most popular grain designs, there exist sections of the grain that consists of a single 2-D contour along the length of the motor axis. For example, consider a finocyl grain design as illustrated in Figure 2.46. The front section of the grain consists of a tube section that consists of a single circular 2-D contour and the rear-most section of the grain consists of a star shape contour.

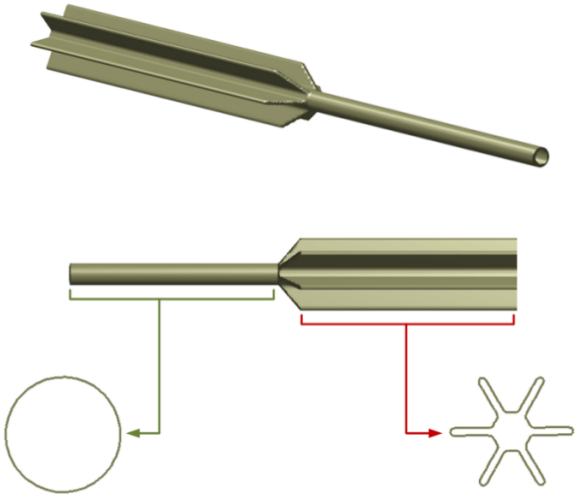


Figure 2.46 – A finocyl grain and the two 2-D contours that define the largest segments of the grain design.

Dividing the grain by planer slices perpendicular to the motor axis results in a

large number of identical segments. If the geometric integration is performed on a single segment of a group of identical segments, integrated values can be awarded to all of the segments contained in the group. Figure 2.47 illustrates the manner in which the grain can be divided and highlights two slices of the star segment that are equivalent and can be awarded equal geometric values.

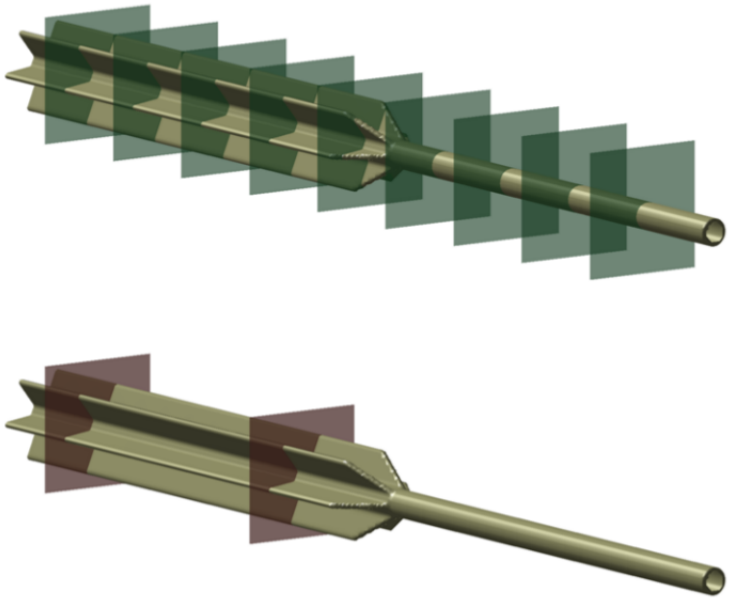


Figure 2.47 – A finocyl grain devided into 2-D slices perpendicular to the motor axis. The two highlighted slices below are equivalant and will have equivalent geometrical properties.

Symmetry in non-axial dimensions: The LSM procedure desribed in the current work is based upon a uniform rectangular grid framework. This does limit the possibility of exploiting symmetry in the non-axial dimensions to perpendicular lines of symmetry, i.e. halves and quarters of a design, and no wedge symmetries could be exploited as is the case with a polar coordinate framework employed by Cavalinni [12].

The finocyl grain design used as illustration in the previous section does, however, contain two perpendicular lines of symmetry, which results in a quarter-domain being used to represent the full design, as illustrated in Figure 2.48.

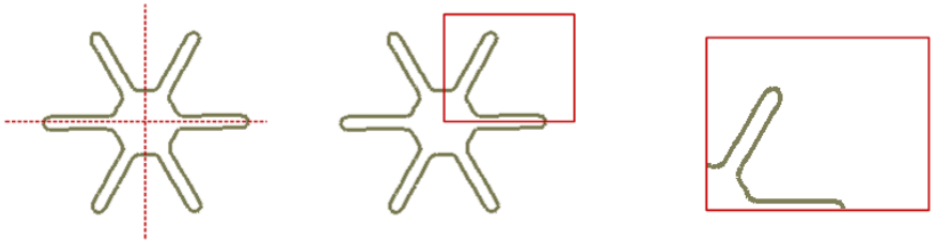


Figure 2.48 – A 2-D illustration of non-axial dimension symmetry in a finocyl grain design.

2.9 Error sources

In this section a discussion of the error sources introduced by the LSM and MC-integration techniques, including the initialization of the SDF, is presented. The errors introduced by the different segments of the grain regression analysis procedure all compound to form the total introduced error. The error sources are classified as one of the following types:

- Initialization and STL surface approximation.
- Discretization of the computational domain.
- LSM numerical integration schemes.
- Geometric evaluation of implicit surface representations.

Initialization and STL surface approximation The first error source is the choice of STL representation of grain designs and motor casing designs. As discussed in section 2.7.2, an STL file uses planer triangular patches to represent a smooth surface. Typically CAD programs allow a user to set the refinement of an exported STL file by controlling two parameters [48]. The first is the maximum angle allowed between two surface patches that approximate a smooth surface. The second is the maximum distance that a vertex of the STL surface approximation is allowed to deviate from the true smooth surface.

The calculation of the SDF values at discrete grid points, to the STL surface is done by analytical functions of vector geometry and is accurate up to machine precision.

Discretization of the computational domain The LSM relies on an implicit representation of a surface on a discrete grid. This could possibly introduce significant errors on jagged surfaces with curvature changes within the discrete grid spacing. Figure 2.49 shows two interfaces that will result in identical SDFs for the chosen discrete grid. The right hand side interface is smooth and the geometric evaluation of the SDFs will return an accurate arc

length approximation of the interface. The arc length of the jagged interface on the left will, however, be substantially under-estimated.

Because even extremely complex grain designs are a compilation of relatively smooth surfaces, this negates the possibly large errors that might be introduced due to the discrete nature of the implicit representations.

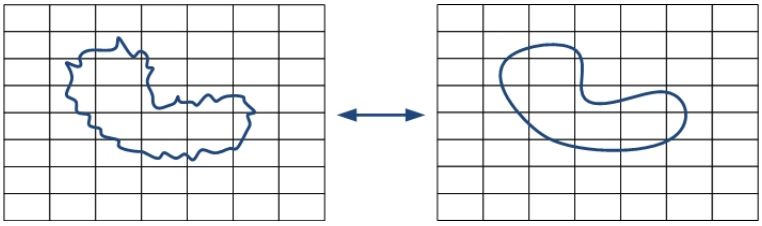


Figure 2.49 – Two distinct interfaces resulting in identical discrete SDFs for a given grid resolution.

LSM numerical integration schemes The integration schemes used to solve the LSM equation, as with most integration schemes, have their own order of accuracy that affects the error introduced. For the current work a first and second order upwind scheme is utilized and it is noted by Sethian [13] that these schemes tend to be sufficient for almost all applications of the LSM.

Geometric evaluation of implicit surface representations The Monte-Carlo integration that is used for parameter calculation of the surface parameters are statistical methods with an expected value that approximates a surface with the same spatial order of accuracy as the interpolation scheme employed to interpolate the MC points to the ϕ function. The number of MC points used to perform the integration affects the variance of the returned data and an increase in number of points increases the accuracy of the integration approximations in a logarithmic fashion. The dependency of the accuracy of the MC-integration is investigated in Section 4.2.

A note on the distribution of the random points used for the MC-integration is also necessary at this point. In order to perform an unbiased approximation of geometric integrals by means of MC-integration, a uniform random number

distribution is required. A number of random number generators exist [49 – 51]. In his article LÉcuyer [52] performs a review of a number of random number generators. The interested reader is also referred to Niederreiter [53].

For the presented work, the built in MATLAB© function `rand()` is used to generate uniform random numbers.

Chapter 3

Internal ballistics coupling

In this chapter, a method of coupling the grain regression module and the internal ballistics (IB) simulations is developed.

The grain regression analysis module and the IB solver simulate two coupled physical processes. The connection between the two processes stem from the arguments made in Chapter 1. A positive feedback loop is caused by the burn rate's dependence on pressure, which in turn depends on mass flow. Mass flow is subsequently dependent on the burning surface which evolves at the burn rate, thereby completing the 'loop'.

The two processes occur during the same time frame, however, at vastly different speeds. The rate of grain regression for most SRMs, relative to the motor casing is typically in the order of $O(10^{-}2)$m/s, whereas the internal flow field travels at speeds in the order of $O(10^{3})$m/s.

In order to couple two discrete numerical procedures that simulate these continuous physical mechanisms, a number of approximations need to be made. The first approximation is inherently present in the IB solver which progresses at discrete time steps and for the duration of the time step assumes the geometry of the grain burning surface to remain fixed. The details of the IB solver are discussed in Section 3.1.1. It can be argued that approximation is likely to introduce negligible errors, since the physical speed of geometric change is very low in comparison to the characteristic speeds of the internal flow field,

as discussed above. The extent of any extra approximation errors introduced depends largely on the method of coupling the two numerical solvers.

The traditional method for coupling a grain regression module to IB code is the so called 'off-line' coupling method, where lookup tables of the geometrical parameters required for IB simulations are created and stored before the IB simulations begin, see Figure 3.1. This method is computationally efficient since the grain regression is done once and only interpolations of the data in the tables is required during the IB simulation.

This method assumes uniformly distributed burn rates for the entire motor and may introduce significant geometrical errors, especially in complex grain designs where changes in topology that are dependent on the rate of surface regression, are a common occurrence.

The effects of phenomena such as erosive burning may increase the variations in the spatial distribution of burning rates, making them difficult to capture using the 'off-line' coupling methodology.

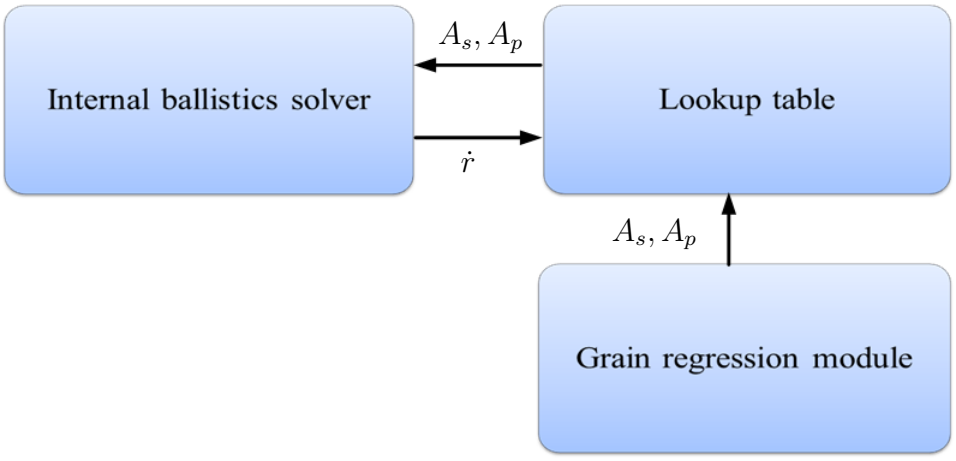


Figure 3.1 – An in direct coupling of the IB and grain regression modules.

The use of numerical front tracking techniques capable of handling spatially varying speeds have made is possible to employ an alternative method, referred to as a direct or 'on-line' coupling, as illustrated in Figure 3.2. The two solvers are employed iteratively, allowing the front to be advanced at more physically relevant, spatially varying, burn rate distributions. Because of the

relatively high characteristic speed of the internal flow field, the discrete time steps of the motor simulation is likely to be small increments determined by a stability condition for the IB solver. This method will be computationally more expensive, since every time step requires the integration of the geometric parameters as well as another discrete evolution of the interface itself.

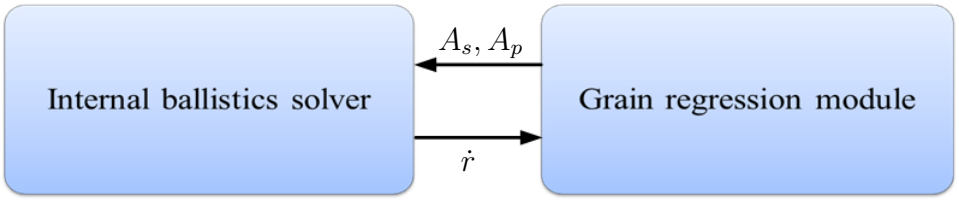


Figure 3.2 – Direct coupling of the IB and grain regression modules.

It was suggested by Stewart et al. [54], as well as noted by Cavalinni [12], that a multi time scale approach similar to the methods developed for computational aerodynamics [55], could be applied to the quasi steady state burning regimes for SRMs, see Figure 3.3. This makes for a hybrid method of the two above mentioned coupling methodologies. The internal flow field is solved at a fine time scale so as to ensure stability of the CFD type calculations, whereas the grain regression analysis is solved at a coarser time scale facilitated by the lower characteristic speeds of the grain surface evolution. The geometric parameters calculated at the coarse time scale is interpolated to the finer time scale of the IB solver. A more detailed look at multi time scale coupling is given in section 3.4.

To the best of our knowledge, at the time that the current work was conducted there was no evidence in the literature of a level set grain regression module that has been successfully coupled to a IB solver, using either a direct or multi time scale coupling methodology. In this study the grain regression module is directly coupled to a IB solver with and without the use of multi time scales.

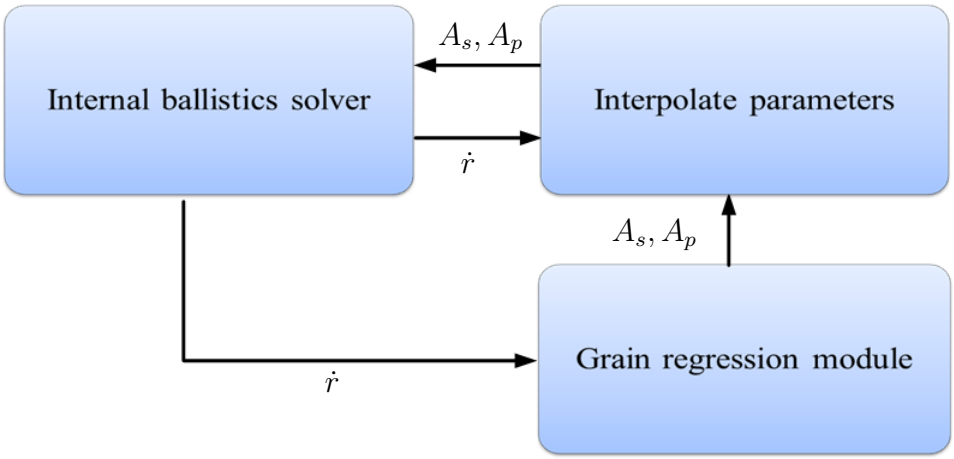


Figure 3.3 – A multi time scale coupling of the IB and grain regression modules.

3.1 Internal ballistics simulation

3.1.1 1-D internal ballistics description

The IB solver chosen for coupling the grain regression module developed in this study is based on the solver employed by the SPP (solid performance program) and is described by Lamberty [56]. The solver relies on a ballistic element method to analyse the internal flow fields of SRMs.

The grain is divided into elements along the motor axis and one dimensional gas dynamic relationships are used to describe the flow conditions in each element. Figure 3.4 illustrates the manner of discretizing the internal flow field and the descriptors of the flow conditions in a single element.

Let $\dot{m}$ be a mass flow rate, ρ the propellant density, A_s the burning surface area $\dot{r}$ the burning rate, M the Mach number, P the pressure and P_0 the stagnation pressure. Also let c and e be empirical constants of the Saint Robert's/Vielle's burn rate model, equation (1.1.3) and R_{gas}, T_f and Γ_{gas} be the specific gas constant, flame temperature and specific heat ratio as derived from the thermodynamic relationships.

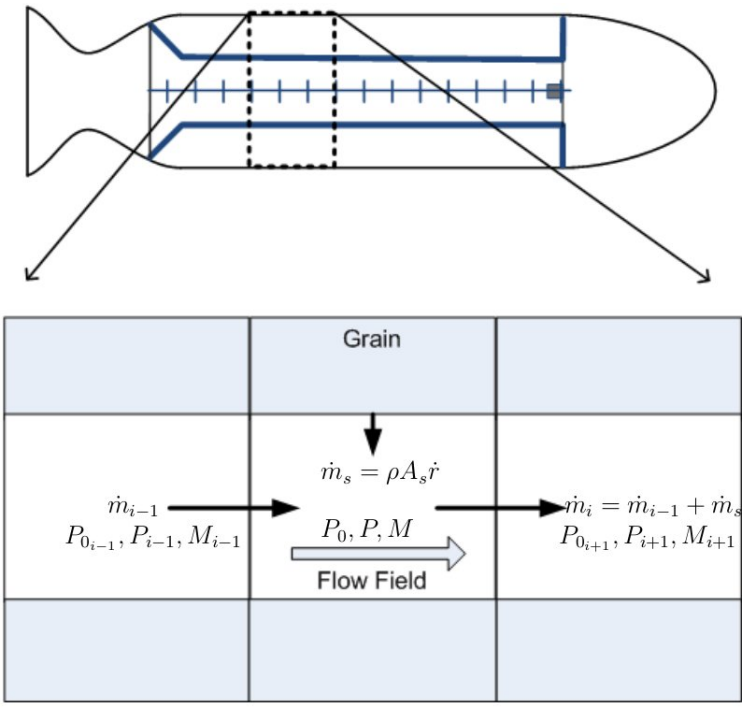


Figure 3.4 – The method used by the internal ballistics module to describe the internal flow field of a SRM

The standard 1-D gas dynamic relationships are written as:

$$\dot{m}_{s_i} = \rho A_{s_i} \dot{r}_i, \tag{3.1.1}$$

$$\dot{r}_i = cP_i^e + f(i), \tag{3.1.2}$$

where $f(i)$ is an erosive burning model,

$$\dot{m}_i = \dot{m}_{i-1} + \frac{1}{2}\dot{m}_{s_i}, \tag{3.1.3}$$

$$\Delta P_0 = \gamma M^2 \frac{\dot{m}_{s_i}}{\dot{m}_i} \times f_l(M_i) \times P_{0_i}, \tag{3.1.4}$$

where $f_l(m)$ is a correction function for compressible flow effects,

$$P_{o_i} = P_{0_{i-1}} - \frac{1}{2}\left(\Delta P_{0_{i-1}} + \Delta P_{0_i}\right), \tag{3.1.5}$$

$$P_i = P_{0_i}\left(1 + \frac{\gamma - 1}{2} M_i^2\right)^{-\gamma/(\gamma-1)}, \tag{3.1.6}$$

$$M_i = \frac{\dot{m}_i}{P_i A_{p_i}} \sqrt{\frac{R_{gas} T_f}{\Gamma_{gas}}} \left(1 + \frac{\gamma - 1}{2} M_i^2\right)^{-1/2}. \tag{3.1.7}$$

Equations (3.1.1) to (3.1.7) are solved by iteration on the Mach number. Once convergence for a specific element is reached, the calculation is advanced and the flow accumulated along the motor axis until the last element is reached. The the nozzle stagnation pressure $P_{0_{NZ}}$, the mass flow rate through the nozzle $\dot{m}_{NZ}$, and the total mass flow from the grain $\dot{m}_s$, is calculated by:

$$P_{0_{nz}} = P_{0_{end}} - \frac{\Delta P_{0_{end}}}{2}, \tag{3.1.8}$$

$$\dot{m}_{nz} = \frac{P_{0_{nz}} \times A_{nz}}{C^*}, \tag{3.1.9}$$

where C^* is the characteristic velocity and

A_{nz} the nozzle throat area,

$$\dot{m}_s = \sum_{i=1}^{end} \dot{m}_{s_i}. \tag{3.1.10}$$

A time unsteady method is employed, i.e. the equilibrium condition $W_{nz} = W_G$ is not forced upon the system. Rather, the foreword end pressure, P_1, at the succeeding time step, is determined by finding the time rate of pressure change, $\frac{dP}{dt}$. This can be done by,

$$\frac{dP}{dt} = \frac{R_{gas}T_f}{\Psi_g}(W_G - W_{NZ}) - \frac{1}{2}\frac{P_1}{\rho}\frac{\dot{m}_s}{\Psi_s}. \tag{3.1.11}$$

where Ψ_s refers to the total volume of the burning chamber or grain perforation. Now a time step Δt is calculated as a fraction of a chamber filling constant,

$$\Delta t = \frac{1}{5}\left(\frac{P_{0_{nz}}\Psi_s}{(1-end)(R_{gas}T_f\dot{m}_{nz})}\right), \tag{3.1.12}$$

where the fraction $1/5$ was chosen from experience as a compromise between accuracy and computational efficiency. The head and pressure at time $t + \Delta t$ is finally calculated as,

$$P_1(t + \Delta t) = P_1(t) + \Delta t\frac{dP}{dt}. \tag{3.1.13}$$

Equations (3.1.1) to (3.1.10) describes in full the modeling of a single time step of the internal flow filed. Once equation (3.1.13) is calculated, the procedure is

re-initiated for the next time frame, this continues until the grain is completely burnt out. An illustration of the computational procedure of the IB module is given in Figure 3.5.

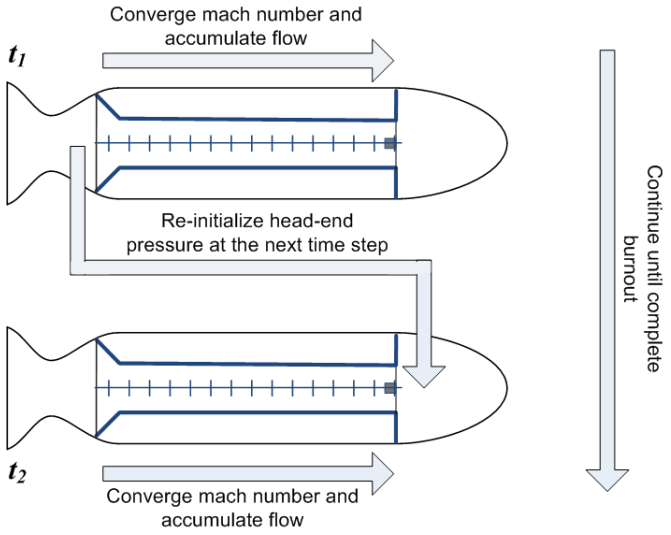


Figure 3.5 – Computational procedure of the IB module.

3.1.2 Burn rate models

The rate at which a solid propellant regresses during combustion, or simply the burn rate, is another domain for which a substantial amount of research exists. During combustion, a solid propellant undergoes an exothermic reaction that is activated by a heat flux into the propellant. Once an activation heat, or energy is reached, the chemical compound which consists of both fuel and oxidizer starts to react in an exothermic manor. This causes a flame zone in the gaseous phase close to the solid phase surface, see Figure 3.6.

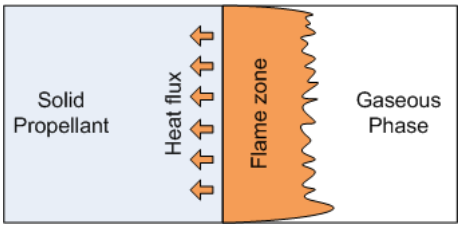


Figure 3.6 – An illustration of the heat flux and flame-zone during solid propellant combustion.

The burn rate of a solid propellant can be approximated by a number of approaches. Most analytical approaches consider the heat flux between the solid and gaseous phases that are contributed by the heat flux from the gaseous flow field, the exothermic reaction occurring at the propellant surface, as well as a heat flux due to radiation from a flame zone, caused by the exothermic reactions taking place. Though there is value in the analytical description of burn rates, IB solvers, for the most part, rely on empirical models based on the Saint Robert's/Vielle's law:

$$\dot{r} = cP_0^n. \tag{3.1.14}$$

The burn rate is described as a function of pressure and the constants c and n are determined through experimental propellant burns at various pressures.

The sheer effects of a cross flow velocity is known as erosive burning and affects most SRMs. These effects are not described in equation 3.1.14 and thus, most commercial IB modules include options for erosive burning terms to be added to the burn rate module. A short description of two popular erosive burning models, as described by Nakka [3], follows.

Additive law The additive law for describing erosive burning is defined as follows; let a_c be an empirically defined constant and U the velocity of the flow field at a given point along the grain surface. An erosive term a_cU is added to the right hand side of the equation (1.1.3). The equation becomes:

$$\dot{r} = cP_0^n + a_cU. \tag{3.1.15}$$

Multiplicative law The IB solver described in Section 3.1.1 does not calculate the flow velocity U explicitly. The multiplicative law for erosive burning is a function of mass flow rate $\dot{m}$ rather than flow field velocity. Let m_c be an empirically defined constant and $\dot{m}^*$ be a threshold flow rate. The erosive term $[1 + \max(0, m_c((\dot{m}) - (\dot{m})^*))]$ is multiplied with the right hand side of equation (1.1.3) and the equation becomes:

$$\dot{r} = cP_0^n[1 + \max(0, m_c((\dot{m}) - (\dot{m})^*))]. \tag{3.1.16}$$

More elaborate burn rate models exist [57 - 60].

3.2 Domain discretization

The grain regression module advances the grain burning surface in a fully 3-D domain, discretized on a uniform rectangular grid Ω_g. The IB solver relies on a Q-1D method for solving the internal flow field and is hence solved on a 1-D grid Γ_{IB} along the motor axis, see Figure 3.7. The grids are co-located such that the 1-D sections of the IB grid concur with a set number of slices of the 3-D grain regression grid (Ω_g) perpendicular to the motor axis. This is done to avoid any unnecessary interpolation during the parameter exchange between the two coupled solvers.

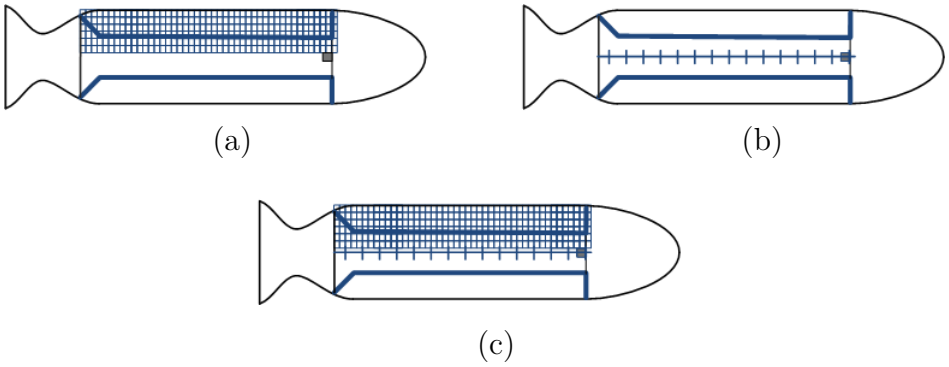


Figure 3.7 – (a) Rectangular grid discretization of LSM grain regression module. (b) 1D grid along the motor axis for IB solver. (c) The co-located domain discretization of the coupled numerical techniques for SRM simulation.

For further simplification only uniform grids will be utilized and the grids are characterized by their grid spacing, dx_g, dy_g, dz_g of Ω_g and dz_{IB} of Γ_{IB}, where it is assumed that the motor axis is parallel to the z-coordinate axis. For the grids to be co-located, dz_g and dz_{IB}, need to adhere to either

$$dz_{IB} = dz_g \times n$$

or,

$$dz_g = dz_{IB} \times n$$

for some natural number n.

3.3 Parameter exchange

The direct coupling of the grain regression and IB modules requires the transfer of parameters between the two solvers. The burning surface area S as well as the port area A_p is transferred from the grain regression module to the IB solver as A_s^{IB} and A_p^{IB}, and the burn rates $\dot{r}$ are transferred back to the grain regression module as $\dot{r}_{LSM}$ from the IB solver. In section 3.2, the co-located domain discretization is explained. For the vast majority of motor simulations it can be expected that the Ω_g grid spacing will be smaller than the Γ_{IB} grid spacing.

For the purpose of illustrating the procedure of parameter exchange between the grain regression and IB module, assume that the grain regression is performed on the grid Ω_g and the IB solver operates on the grid Ω_{IB}. Assume also that n a natural number and that $dz_{IB} = dz_g \times n$. Further let the $(\Omega_g)_j$ be the j'th plane of Ω_g along the motor axis, and $(\Omega_{IB})_i$ be the i'th element of Ω_{IB} and the grid be located so that $(\Omega_{IB})_1$ and $\{(\Omega_g)_1, (\Omega_g)_2, \ldots, (\Omega_g)_n\}$ precisely overlap, as illustrated in Figure 3.8.

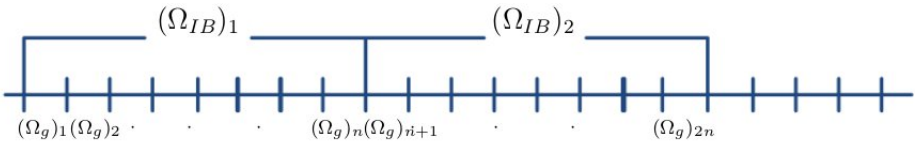


Figure 3.8 – Co-located $\Omega_I B$ and Ω_g grids.

The A_s^{IB} parameter is calculated as the sum of A_s values of Ω_g that lie within a specific discrete segment of the $\Omega_I B$ grid, and the A_p^{IB} parameter is calculated as the average port area of the Ω_g The parameters sent to the IB module can be calculated by,

$$(A_s^{IB})_i = \sum_{j=(i-1)n+1}^{j=in} (A_s)_j, \tag{3.3.1}$$

and

$$(A_p^{IB})_i = \frac{1}{n} \sum_{j=(i-1)n+1}^{j=in} (A_p)_j. \tag{3.3.2}$$

The rate at which the interface is advanced, $\dot{r}_g$ is set equal to the burn rate $\dot{r}$ of the segment of the Ω_{IB} grid with which the interface is co-located along

the motor axis,

$$(\dot{r}_g)_j = \dot{r}_{\text{ceil}(j/n)}. \tag{3.3.3}$$

Equations (3.3.1) to (3.3.3) describe a procedure for exchanging the relevant parameters between the two coupled modules of the SRM simulation, as discussed in this study. A variation on equation (3.3.3), where the burn rates $\dot{r}$ are interpolated from the coarse Ω_{IB} grid to the finer Ω_g grid of the grain regression module is investigated, as this might improve the accuracy of the physical description of a SRM. The rate of interface advancement $\dot{r}_g$ is then given by,

$$(\dot{r}_{gr})_j = \dot{r}_{\text{floor}(j/n)} + (\dot{r}_{\text{ceil}j/n} - \dot{r}_{\text{floor}(j/n)}) \times \frac{\text{mod }(j,n)}{n} \tag{3.3.4}$$

Note that ceil(j/n) and floor(j/n) denotes the fraction j/n rounded up or down to the nearest integer.

3.4 Multi time scale coupling

In order to reduce the computational cost of coupled SRM simulation, a multi time scale procedure may be employed. As mentioned above, the assumption that the time scale as determined for the IB solver is excessively fine for the purpose of grain regression and so in this section the use of a coarser time scale is considered.

The reduction in computational cost will be equivalent to the cost of integration of the area parameters A_s and A_p multiplied by the average ratio of the two time scales used. The conditions for determining the grain regression time scale must primarily adhere to the conditions for stability of the integration schemes used to solve equation (2.4.6). For instance the first order upwind scheme with Euler time integration as described in Section 2.5 requires the CFL condition, $dt \leq \frac{dx}{V}$, be satisfied.

If the stability conditions are the only means by which the grain regression time scale is chosen, the coupling algorithm is developed as follows.

- The simulation is initiated and with the initial grain regression parameters calculated form the SDF before any interface evolution.

- Initialize t_1 and t_2 to 0.
- The IB module returns initial burn rates and the grain is regressed for a time step Δt_1, determined as a function of the limitation imposed by the stability conditions on the maximum possible time step for interface advancement.
- The grain parameters are saved at times t_1 and $t_1 + \Delta T_1$.
- The IB code is advanced along with time t_2 and the area parameters are interpolated to $t_1 > t_2 > t_1 + \Delta t_1$.
- At the first instance that $t_2 \geq t_1 + \Delta t_1$, t_1 is set to $t_1 + \Delta t_1$ and a new Δt_1 is calculated using the burn rates at t_2 and the interface advanced to the new time $t_1 + \Delta t_1$, and the process continues until the grain is completely burnt out.

The procedure is illustrated in Figure 3.9.

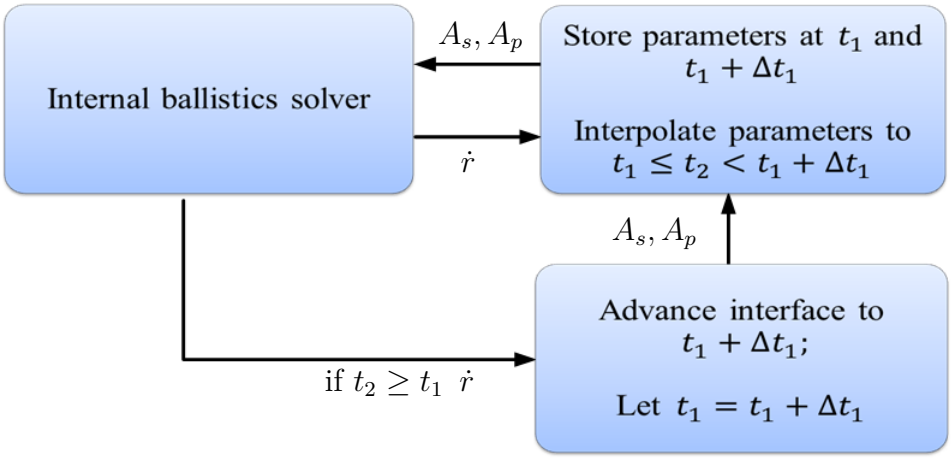


Figure 3.9 – A multi time scale coupling of the IB and grain regression modules.

Chapter 4

Results and validation

In this chapter a number of results are presented in order to validate the techniques applied to the geometric property evaluation of implicit surfacesng with the use of Monte-Carlo integration and the numerical interface advancement through the LSM, as well as their application to grain regression analysis in SRM simulation.

Validation by comparison to experimental data is useful, however it should be understood that there are often a number of external factors that contribute to the level of concurrency with experimental data. This is especially true during the comparison of full motor simulations (both coupled and uncoupled simulations), with static test-bed results. The accuracy of the IB module, the empirical burn rate model, and most notably the effect of erosive burning and nozzle erosion terms need to be taken into account before any conclusions are drawn from such comparisons. As a result, a larger portion of the validating results are relatively simple analytical examples.

In the first section a grid dependency study is conducted on the discrete SDF generation from STL surfaces. A number of examples of actual SRM motor geometries in the form of triangulated STL files from CAD designs are used to conduct the investigation. Various grid resolutions are used to resolve the same designs and the effect on the accuracy of the surface approximations are discussed

The following section focuses on the MC integration techniques. The depen-

dency of accuracy of surface approximation on the number of random integration points (MC points), the spatial computational grid resolution and the width of the thin envelope used for surface integration in 3-D, is discussed. The reduction in computational cost due to optimizations of the integration, such as stratification, is also presented and discussed. Both analytical examples and SRM grain designs are used as interfaces in the cases presented.

Next the ability of the LSM numerical interface propagation technique to handle traditionally challenging situations is confirmed through the investigation of three 2-D examples. The examples expand directly to analogue 3-D situations and they are given in 2-D form simply for ease of illustration.

Finally an uncoupled, as well as a coupled motor simulation is performed, and compared to actual experimental results. The effects of the addition of erosive burning models and nozzle erosion terms to the IB module are also briefly discussed.

4.1 SDF grid dependency

In order to conduct the investigation, the MATLAB© function `isosurface()` is used to return a triangulated representation of the SDF's zero isosurface. It was shown in Section 2.8.2, that changes in topology and sharp discontinuities in the surface gradient could possibly cause physically incorrect surfaces to be returned by a marching cube type algorithm, such as `isosurface()`.

In order to ensure accurate representations of the implicit surfaces, the result of `isosurface()` was visually inspected for any discrepancies with the STL surface due to marching-cube ambiguities, as discussed in Section 2.8.2. An example where an incorrect surface was returned by the `isosurface()` function is illustrated later in this seciton (Figure 4.5).

The grid dependency of the implicit surface representation by means of a discrete SDFs, could thus be investigated. The surface areas of both the STL surfaces and the triangulated implicit surface representations are calculated by performing a numerical quadrature on the triangulated surfaces.

4.1.1 Motor 1 - gf

Figure 4.1 is an illustration of the motor geometry, as veiwed from various angles. The motor design contains radial slots that are tapered toward the nozzle of the motor.

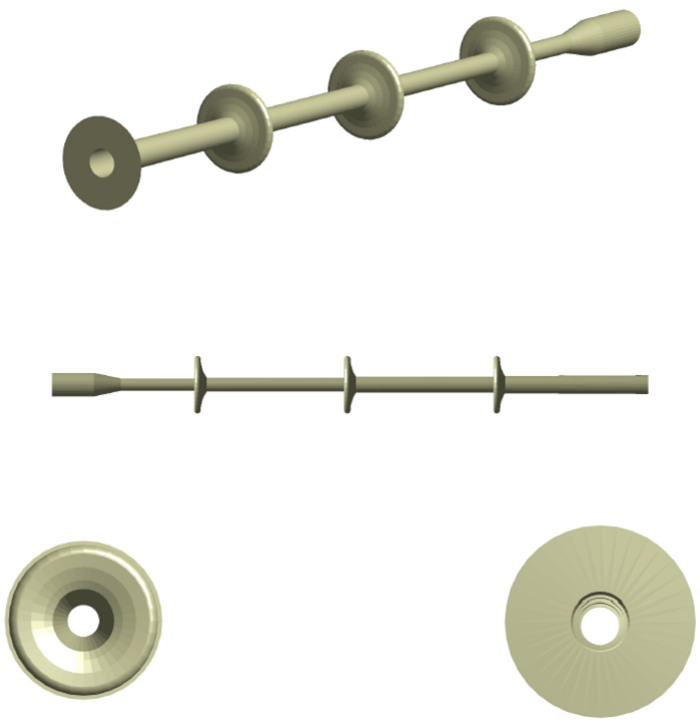


Figure 4.1 – The burning surface of Motor 1 - gf, as viewed from various angles.

The grid spacing of the discrete SDF was varied between 1 and 12 units resulting in grid domains of sizes $140 \times 140 \times 450$ down to $11 \times 11 \times 37$. The resulting errors are recorded in Table 4.1. A plot of the errors versus the grid spacing is given in Figure 4.2. From the figure, a large increase in the errors between the grid spacings of 5 and 8, and again between 11 and 12 can be observed. This is possibly due to features of the STL surface, such as the radial slots, that are separated by less than the grid spacing. Tubular sections of the grain are likely to be increasingly underestimated as the grid spacing is increased. This is true since the curved surface is approximated by larger sections of linear planes.

Table 4.1 – The resulting errors in the numerical quadratures of the implicit representation of motor 1 - gf, for various grid sizes.

Grid spacing	Grid size	Percentage errors
12	$11 \times 11 \times 37$	46.4828 %
11	$12 \times 12 \times 40$	31.6872 %
10	$14 \times 14 \times 45$	29.7793 %
9	$15 \times 15 \times 50$	29.7305 %
8	$17 \times 17 \times 56$	30.9707 %
7	$20 \times 20 \times 64$	20.7866 %
6	$23 \times 23 \times 75$	11.7535 %
5	$28 \times 28 \times 90$	3.1020 %
4	$35 \times 35 \times 112$	3.4256 %
3	$46 \times 46 \times 150$	2.0299 %
2	$70 \times 70 \times 225$	0.7494 %
1	$140 \times 140 \times 450$	0.1939 %

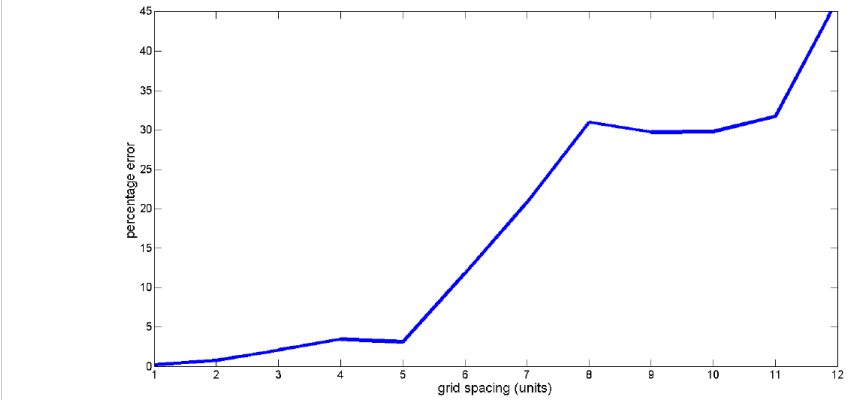


Figure 4.2 – A plot of the errors in the implicit representation of motor 1-gf.

4.1.2 Motor 2 - A1

The next motor used for SDF grid dependency investigation is composed of a star and a cylindrical section. The burning surface, as viewed from various angles, is illustrated in Figure 4.3.

The investigation was conducted for grid spacings varying from 12 to 2 leading to grid domains ranging from $90 \times 90 \times 655$ down to $15 \times 15 \times 54$. The absolute errors in the implicit surface areas are recorded in Table 4.2.

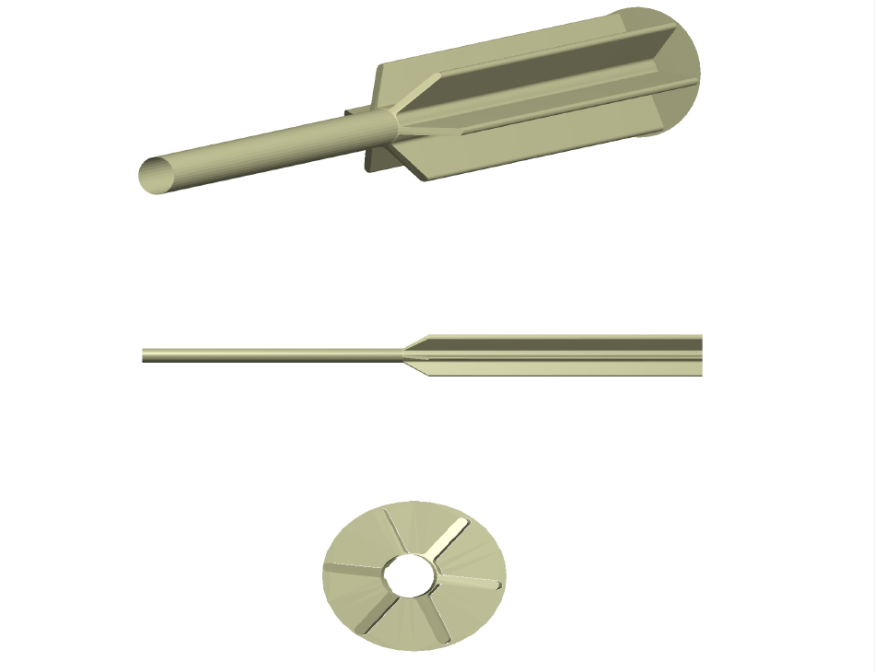


Figure 4.3 – The burning surface of Motor 2 - A1, as seen from various angles.

Table 4.2 – The resulting errors in the numerical quadratures of the implicit representation of motor 2 - A1, for various grid sizes.

Grid spacing	Grid size	Percentage errors
12	$15 \times 15 \times 54$	17.3896 %
11	$16 \times 16 \times 59$	5.7804 %
10	$18 \times 18 \times 65$	2.1107 %
9	$20 \times 20 \times 72$	0.1647 %
8	$22 \times 22 \times 81$	4.0964 %
7	$25 \times 25 \times 93$	4.585%
6	$30 \times 30 \times 109$	3.0544 %
5	$36 \times 36 \times 131$	1.7978 %
4	$45 \times 45 \times 163$	1.6576 %
3	$60 \times 60 \times 218$	0.4337 %
2	$90 \times 90 \times 327$	0.1069 %

A plot of the percentage error versus the grid spacing is given in Figure 4.4.

Note in Figure 4.4 that the error for a grid spacing of 8 and 9 is very low in comparison to that for grid spacings of 8 or 11. This is due to the fact that the fin sections at the aft end of the motor are set to a width of 9 units,

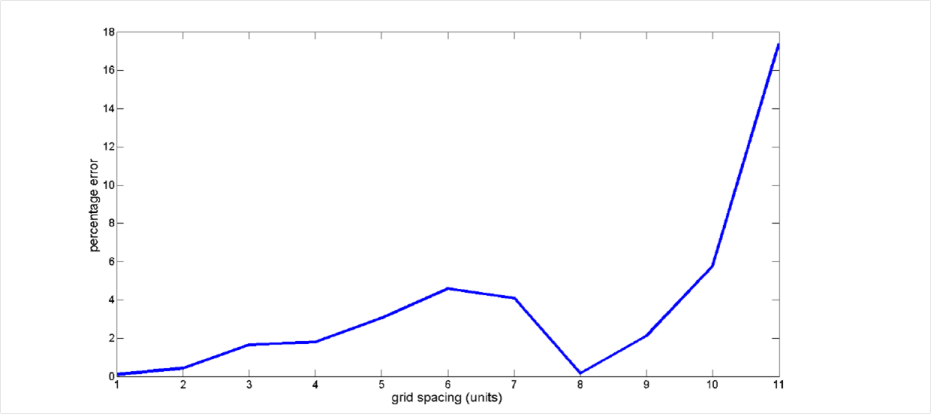


Figure 4.4 – A plot of the errors in the implicit representation of motor 2 - A1.

and are overestimated for a grid spacing of 9 due to incorrect results of the `isosurface()` function, as illustrated in Figure 4.5. As was the case for the previous grain design, section 4.1.1, the tubular sections are underestimated for large grid spacings. The overestimation of the fin surface together with the underestimation of the tubular section possibly gives the false impression of an accurate approximation.

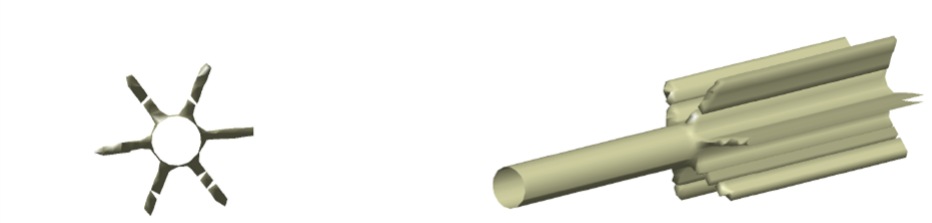


Figure 4.5 – An incorrect result from the `isosurface()` function.

4.2 Validation of Monte-Carlo integration techniques

The dependency of the accuracy of the MC integration on number of MC integration points, and the width of the thin envelope used for surface approximation, is investigated.

4.2.1 Single voxel volume integration

The integration of a single voxel is performed by means of MC-integration. A voxel cut by an interface through a central plane, as illustrated in Figure 4.6, such that half of the volume of the voxel lies within the interface, was considered.

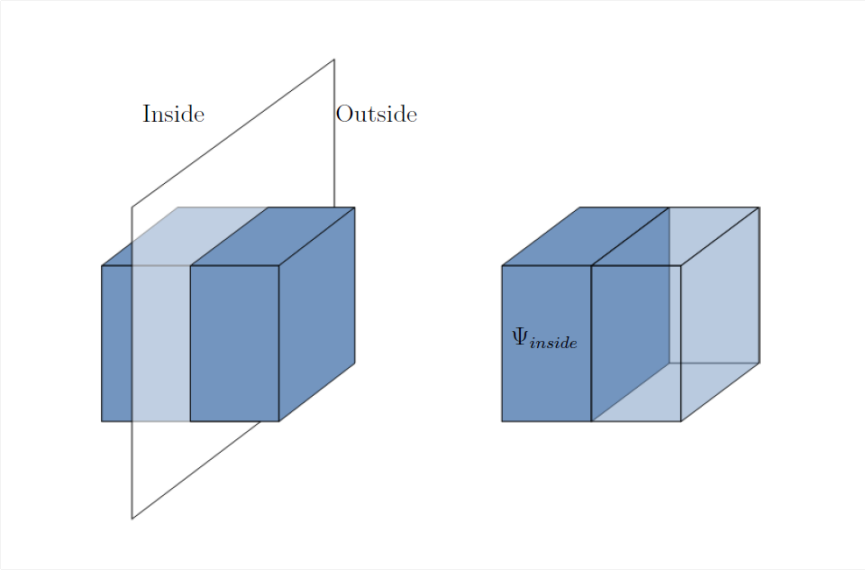


Figure 4.6 – A voxel cut in half by an interface through a central plane.

An implicit representation, ϕ_{vox}, of the interface in Figure 4.6, was defined on the voxel vertices by setting the vertices of the voxel that lie inside the interface equal to -0.5, and the vertices on the outside equal to 0.5. Figure 4.7 gives a graphical illustration of the implicit representation.

The voxel is a cube of volume $\Psi_{vox} = 1$ and the volume inside the interface is equal to $\Psi_{inside} = 0.5$. The number of MC-points used for the purpose of the volume approximation was varied from 1 to 500 000 and each simulation was repeated multiple (100) times, in order to construct statistical confidence intervals on the results of the MC-integration technique. The results of the MC-integration of the implicit interface ϕ_{vox}, of Figure 4.7, is summarized in Table 4.3. The expected volume $E(\Psi_{inside})$, variance $VAR(\Psi_{inside})$, as well as a 95% confidence interval is given for an extract of the experiments.

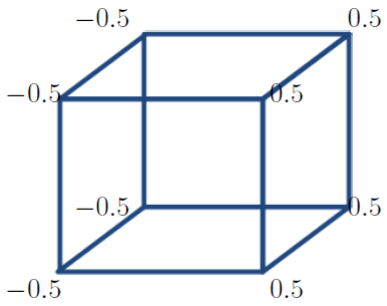


Figure 4.7 – An implicit interface representation of the interface which cuts the voxel in half.

Table 4.3 – An extract from the results of the MC-integration of ϕ_{vox} for various numbers of MC-points.

MC-points	True Area	$E(\Psi_{inside})$	$VAR(\Psi_{insode})$	95%-conficence interval
1	0.5	0.58	0.4986	$\{-0.2476; 1.4076\}$
5	0.5	0.5160	0.2526	$\{0.0966; 0.9354$
10	0.5	0.5160	0.1595	$\{0.2512; 0.7808\}$
50	0.5	0.4832	0.0714	$\{0.3647; 0.6017\}$
100	0.5	0.5010	0.0447	$\{0.4268; 0.5752\}$
500	0.5	0.4984	0.0199	$\{0.4653; 0.5315\}$
1000	0.5	0.5014	0.0135	$\{0.4790; 0.5238\}$
5000	0.5	0.5004	0.0070	$\{0.4887; 0.5121\}$
10000	0.5	0.5002	0.0050	$\{0.4920; 0.5085\}$
50000	0.5	0.5003	0.0021	$\{4969; 0.5037\}$
100000	0.5	0.5002	0.0013	$\{4980; 0.5024\}$
500000	0.5	0.5000	0.0007	$\{0.4988; 0.5011\}$

A plot of the expected values as well as the variance of the volume versus the number of MC-points ranging from 1 to 5000 is given in Figure 4.8.

Similar plots where the number of MC-points range from 100, in steps of 100, to 60 000, are given in Figure 4.9.

95% confidence intervals can be constructed by $E(\Psi_{inside}) \pm 1.66 VAR(\Psi_{inside})$. A plot of the resulting confidence intervals is given in Figure 4.10 for MC-points ranging from 1 to 5000.

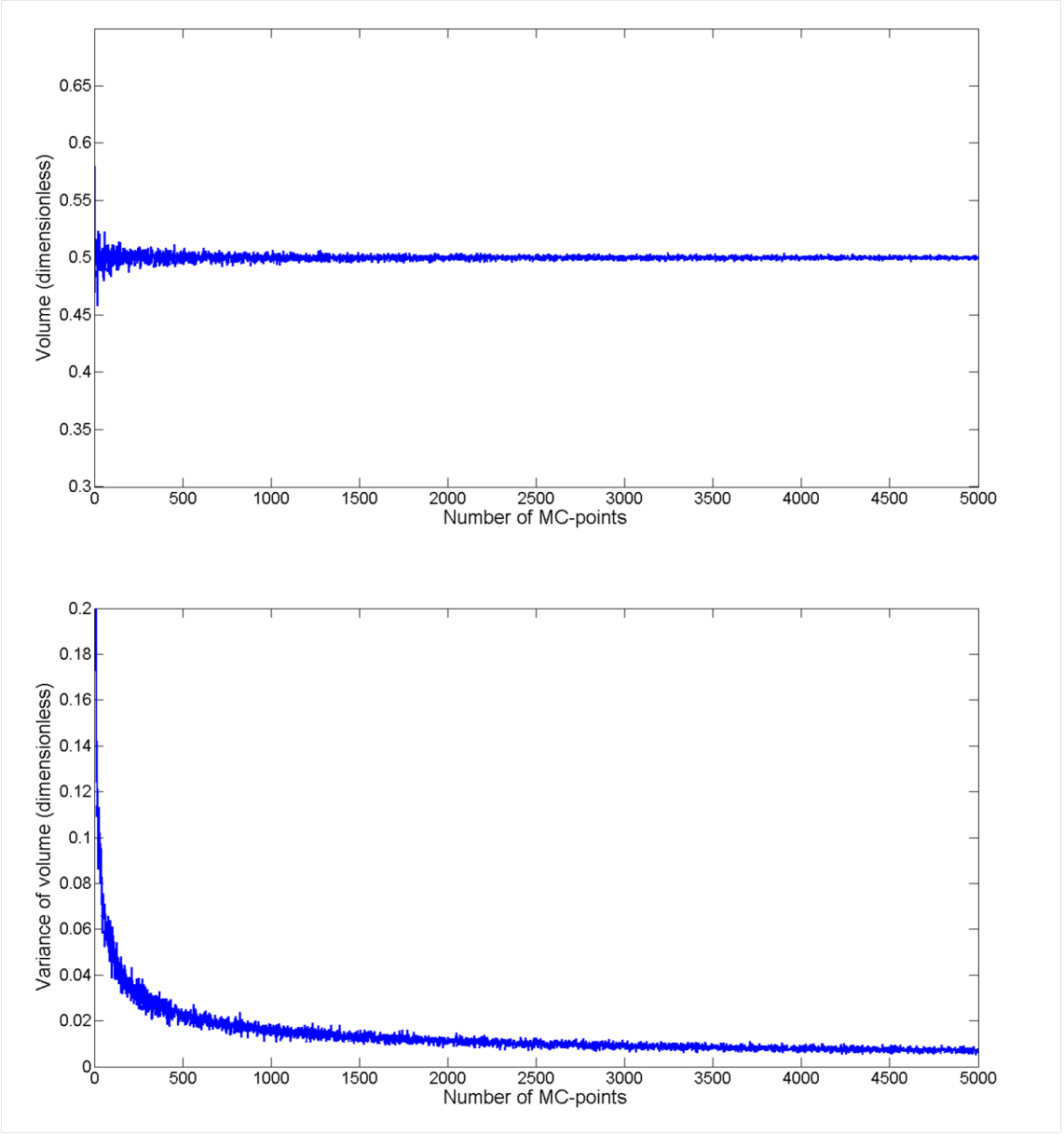


Figure 4.8 – A plot of expected value $E(\Psi_{inside})$ (top), and variance $VAR(\Psi_{inside})$ (bottom), versus the number of MC-points used for MC-integration.

A similar plot where the number of MC-points ranges from 100, in steps of 100, to 60 000, is given in Figure 4.11.

It is now possible to give the maximum error within the 95%-confidence interval. I.e. there is 95% certainty that the error in the approximated volume

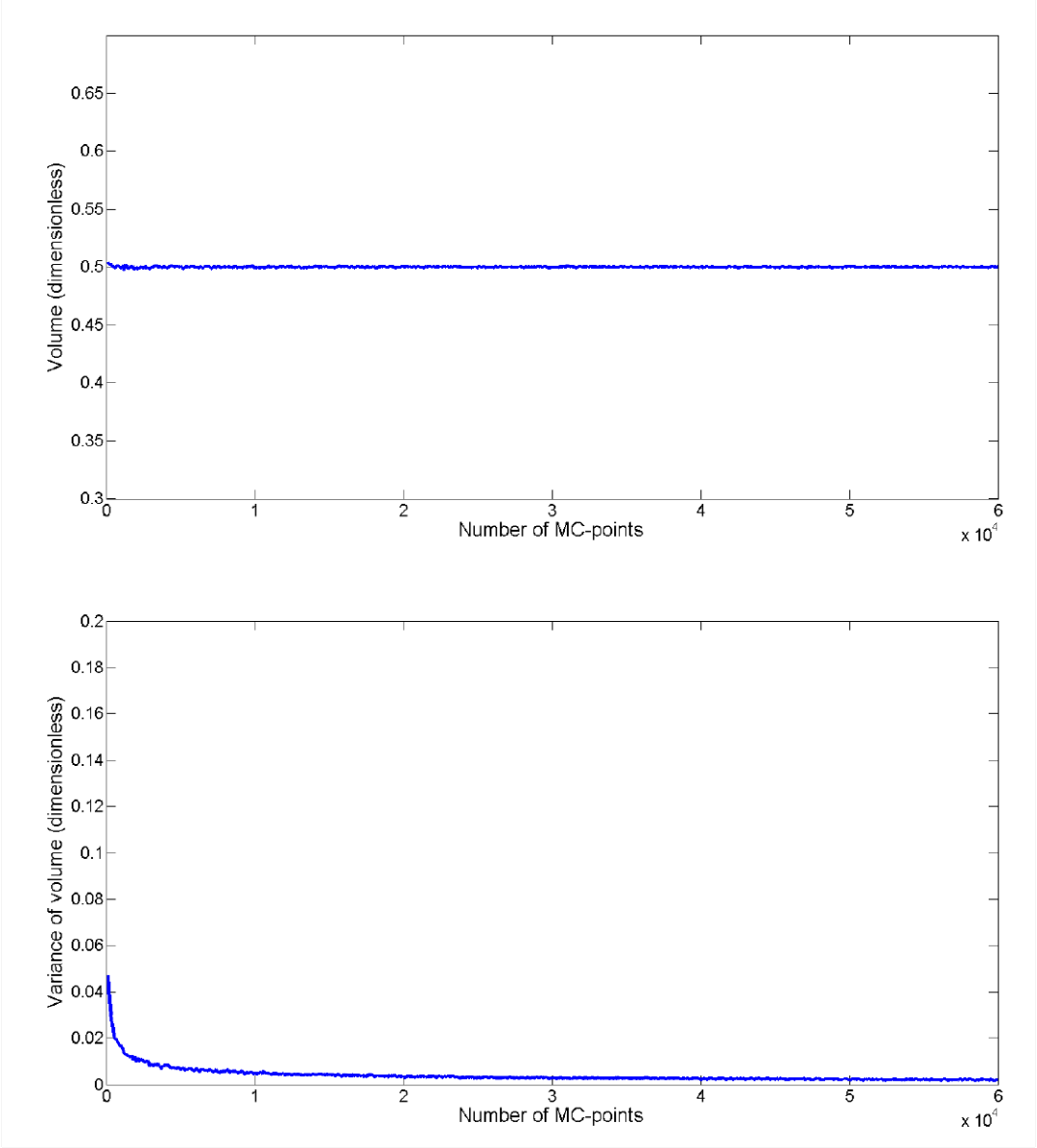


Figure 4.9 – A plot of expected value $E(\Psi_{inside})$ (top), and variance $VAR(\Psi_{inside})$ (bottom), versus the number of MC-points used for MC-integration.

for the corresponding number of MC-points will be smaller than the maximum error. The Maximum error for both 1 to 5000 and 100 to 60 000 points is given in Figure 4.12 and Figure 4.13, respectively.

From the results it can be concluded that the expected absolute error intro-

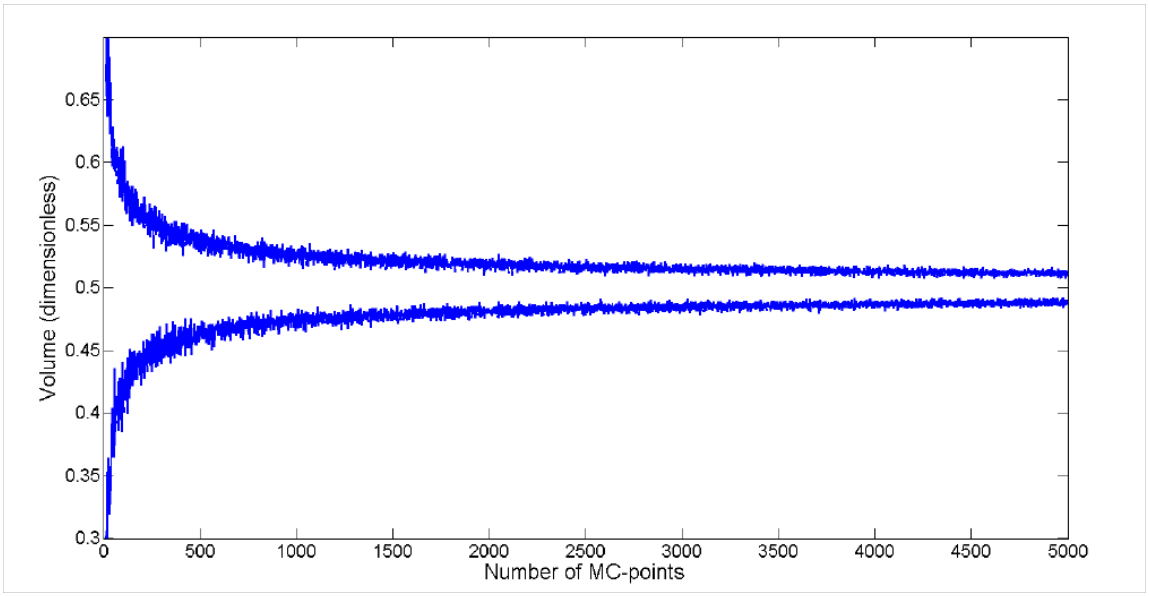


Figure 4.10 – A plot of the 95%-confidence intervals versus the number of MC-points used for MC-integration.

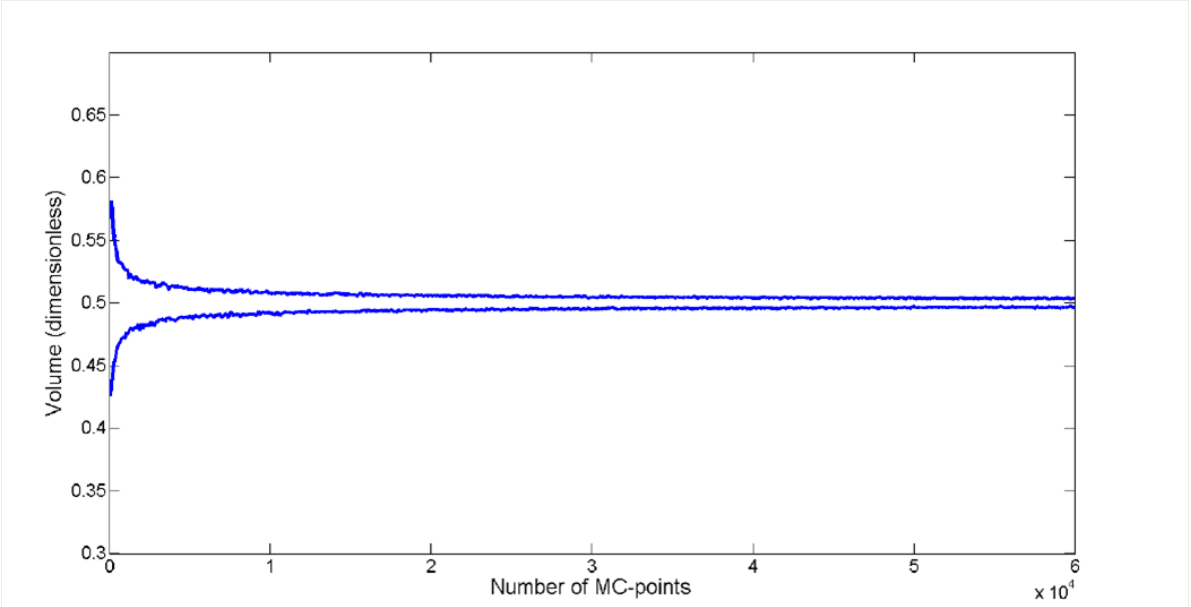


Figure 4.11 – A plot of the 95%-confidence intervals versus the number of MC-points used for MC-integration.

duced by MC-integration approximation, decreases in a logarithmic fashion with the increase of the number of MC-points used for the integration. For a voxel with volume $\Psi_{vox} = 1$, and a volume fraction inside an interface such that

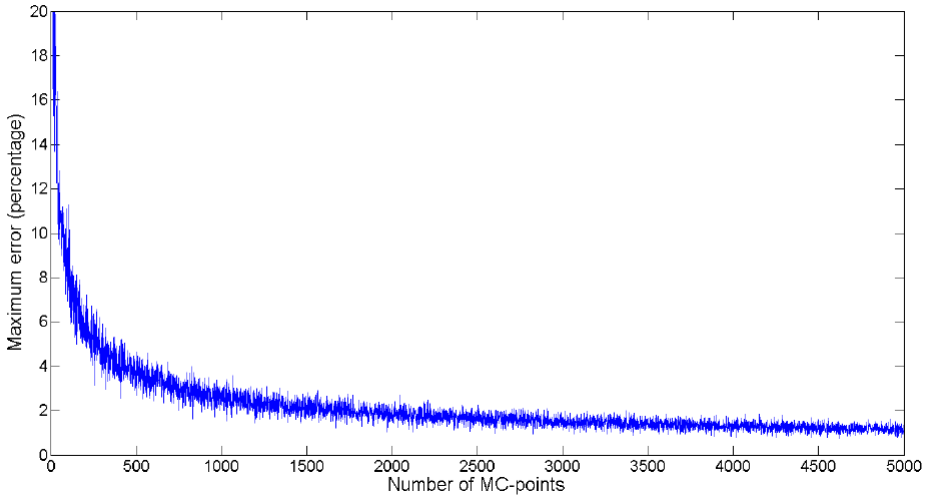


Figure 4.12 – The maximum error within a 95%-confidence interval versus the number of MC-points used for MC-integration.

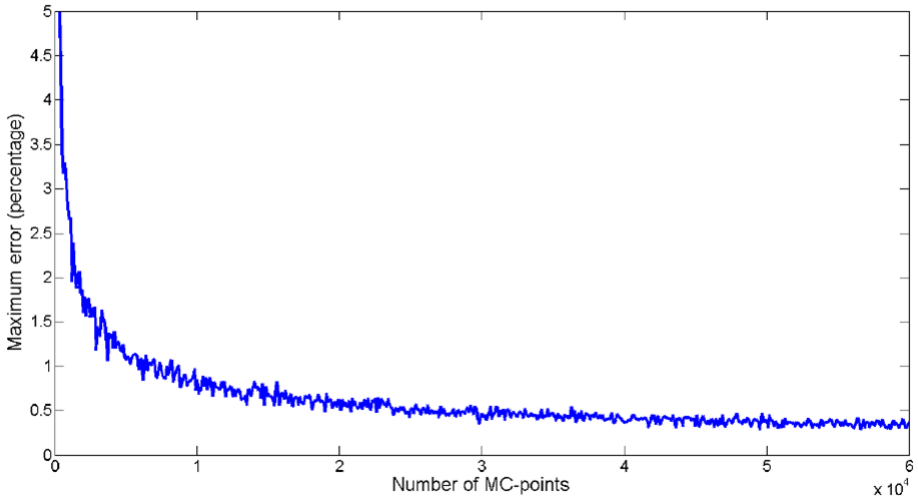


Figure 4.13 – The maximum error within a 95%-confidence interval versus the number of MC-points used for MC-integration.

$\Psi_{inside} = 0.5$, the approximation error will be below 1% with 95% certainty, if more than 10 000 MC-points are used in the integration process.

4.2.2 Area integration: Two merged spheres

In order to investigate accuracy of surface integration by means of thin-envelope approximation and the effect of the width of the envelope used, an analytical case study was set up as follows:

Two spheres of radii 3- and 4-units respectively were set up with center points a distance 5-units apart, as illustrated in Figure 4.14. The union of the two spheres was taken as the analytical interface to be investigated.

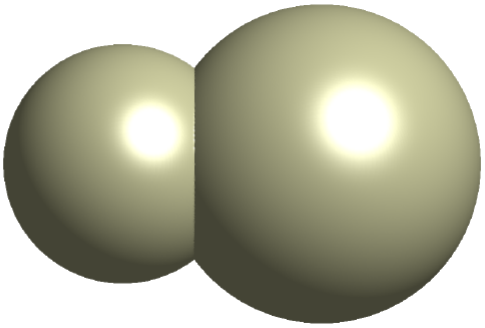


Figure 4.14 – The union of two merged spherical interfaces.

The interface has a true surface area of 271.4336 units^2. MC-surface-integration by means of thin-envelope approximation was performed on an implicit representation of the analytical surface, defined on discrete grids with varying domain sizes and for varying thin envelope widths.

A surface plot of the absolute errors in the surface approximation to the analytical solution, for surface integration of implicit function with domain sizes, or grid resolution, ranging from $20 \times 20 \times 30$ to $160 \times 160 \times 240$ and thin envelope widths ranging from 0.25 to 6.0 is illustrated in Figure 4.15. Note that the thin envelope widths is given in terms of the grid spacing and not necessarily the global units used to define the implicit function.

The errors tend to decrease with an increase in grid resolution and seem to be only be influenced to a lesser extend by the thin envelope width, especially for higher grid resolutions. The general trend is that the errors decrease with a decrease in envelope width. For envelope widths of 0.25 and 0.5 however, the results tend to be less stable.

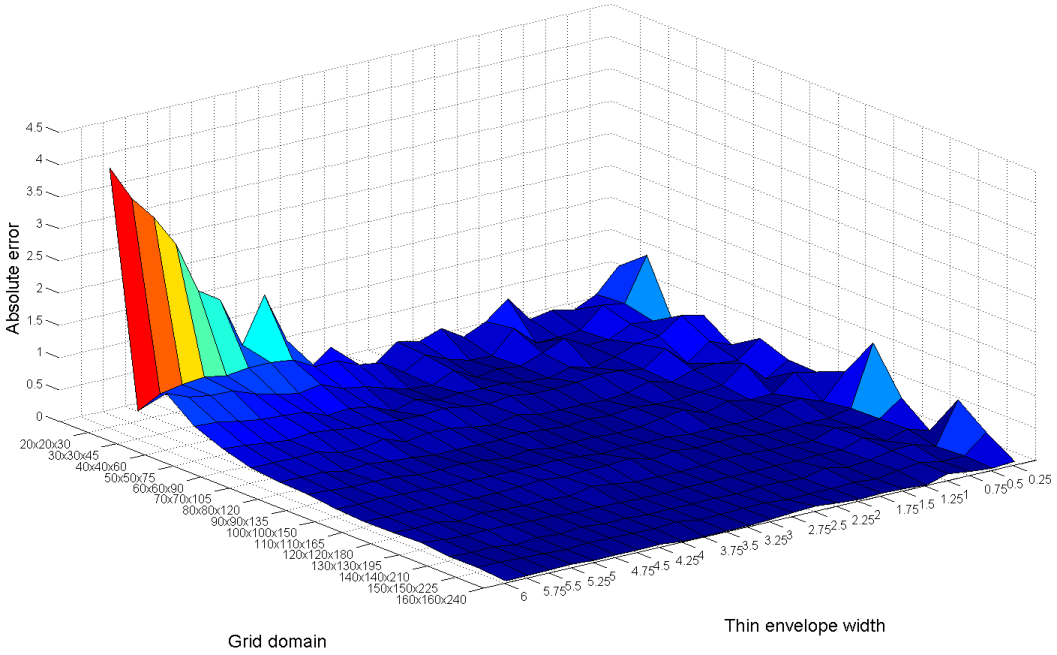


Figure 4.15 – Absolute errors of the MC-integration of the merged spheres interface, for various envelope widths and a grid resolutions.

The average absolute error for each grid resolution, and for each envelope width, is given in Figure 4.16 and Figure 4.17, respectively.

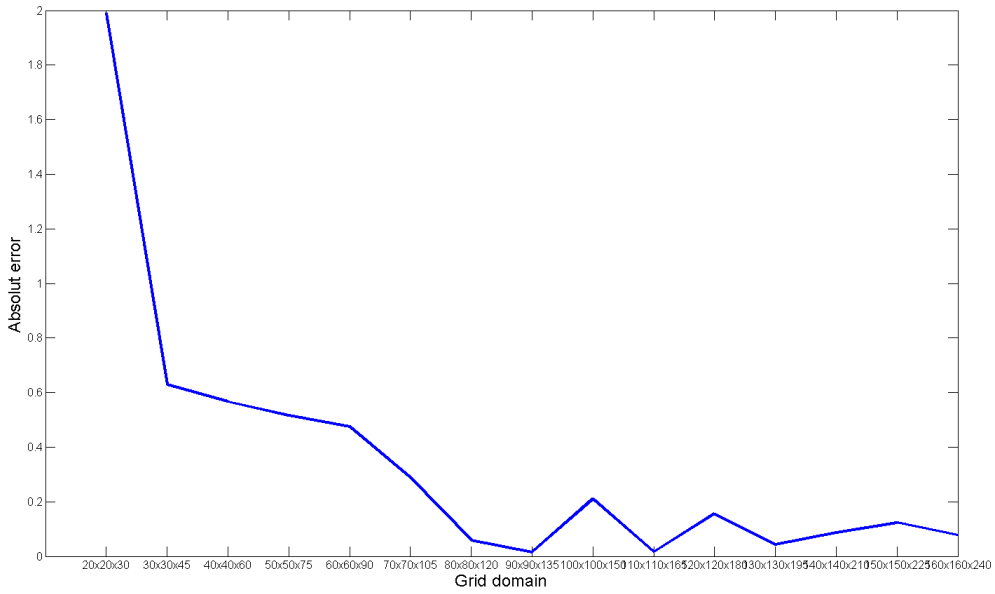


Figure 4.16 – Average absolute errors of the MC-integration of the merged spheres interface, for varying grid resolutions.

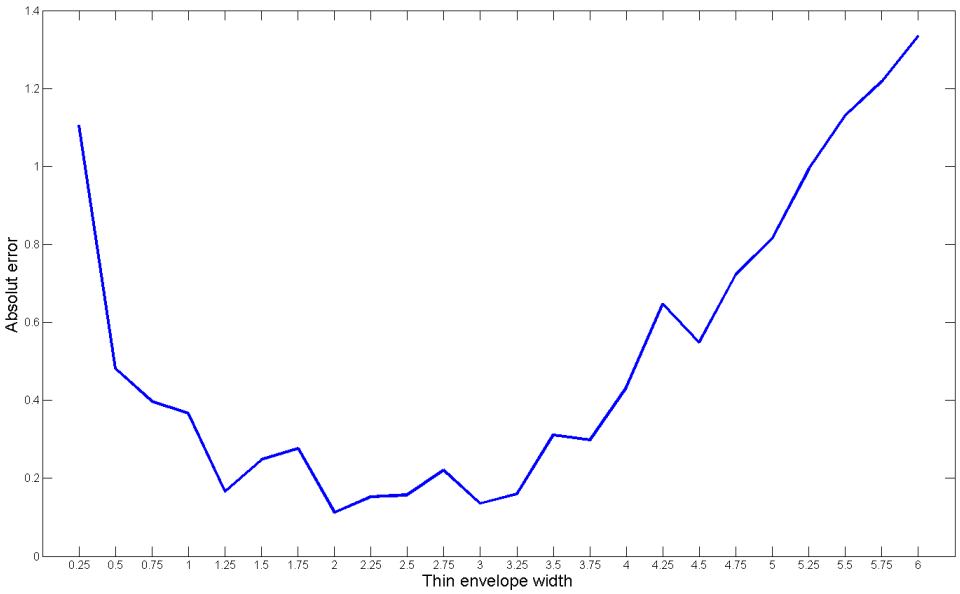


Figure 4.17 – Average absolute errors of the MC-integration of the merged spheres interface, for varying thin envelope widths.

The approximated surface areas all lie within a 1% of the analytical surface area, except for the results for the the thin envelope widths of 5.0 and 6.0 at the lowest grid resolution, where the thin envelope actually intersects the domain boundary.

4.2.3 Area integration: SRM grain surface

Two separate SRMs are integrated by the MC-integration procedure, as further validation of the technique. The two grains are integrated using varying thin envelope widths. The first, a tubular type grain with radial slots (Figure 4.18) and the second a finocyl design (Figure 4.19).

The grid domain used to describe the two grain designs are of the size, $85 \times 85 \times 700$ for the radially slotted design, and $55 \times 55 \times 494$, for the finocyl design.

The percentage errors of the area integration of the finocyl design are plotted as a function of the thin envelope width in Figure 4.20

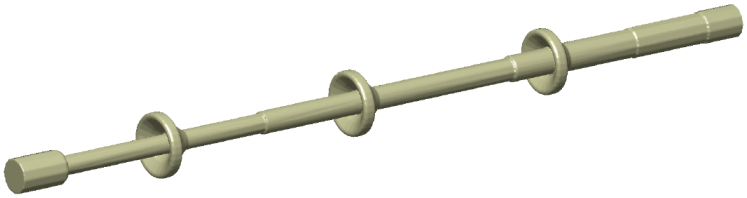


Figure 4.18 – A radially sloted grain design.

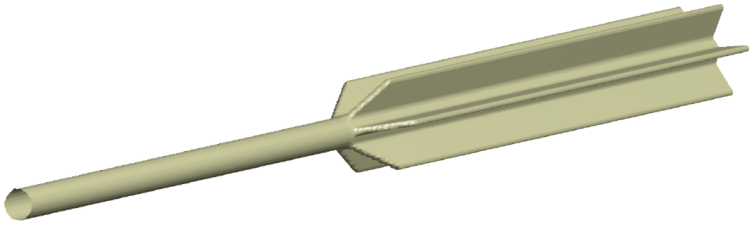


Figure 4.19 – A finocyl grain design.

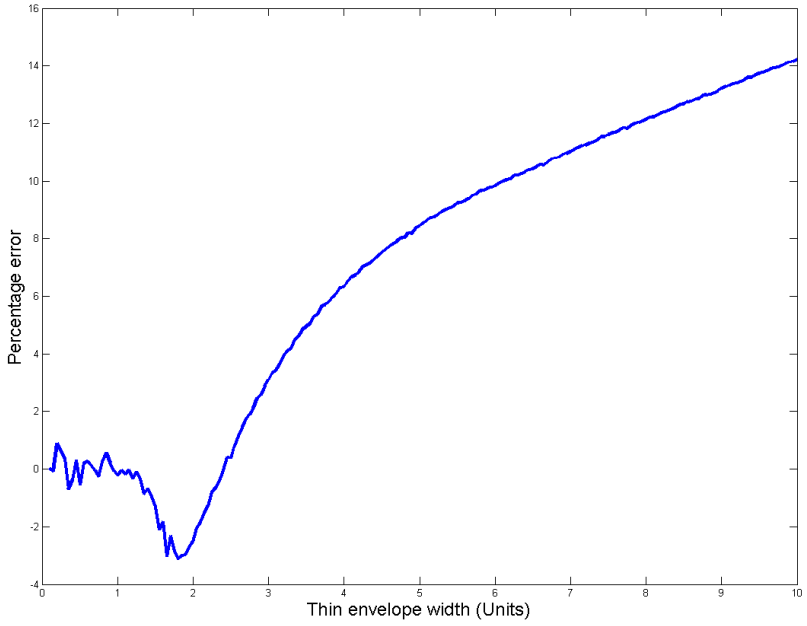


Figure 4.20 – A plot of the percentage error of the MC surface integration of a finocyl motor design

From the figure it can be seen that for relatively large envelope widths, the error decreases with a decrease in the width. For the finocyl design the area is under-approximated for thin envelope width in the region of two units. This is most likely due to the precense of sharp corners along the fins. The absolute error does however decrease for envelope widths in the region of 1 unit, and the error is well within the 1% range. As the wdith is decreased further, the area integration becomes less stable as the number of random points within a very thin envelope decreases and the uncertainty in the area approximation becomes larger. A more detailed illustration of the error for lower envelope widths is given in Figure 4.21.

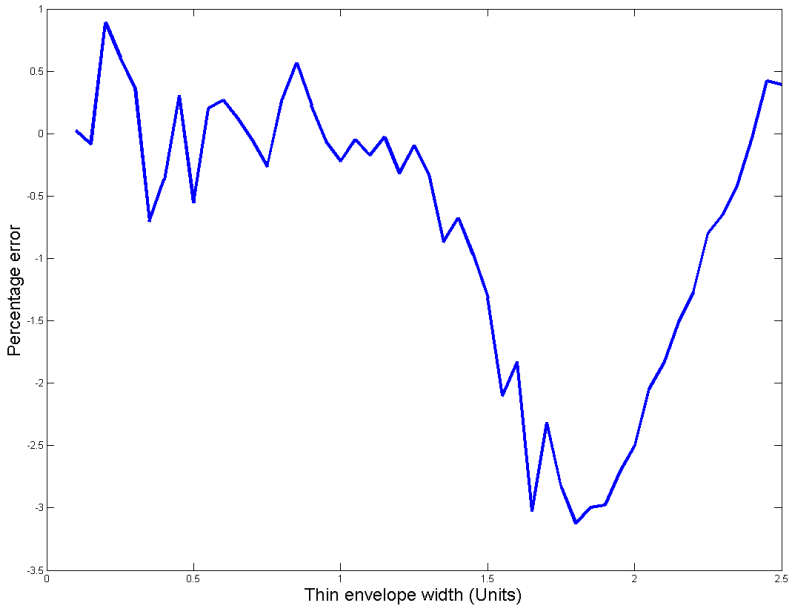


Figure 4.21 – A plot of the percentage error of the MC surface integration of a finocyl motor design, for low envelope widths.

The results of the area integration of the radially slotted motor are illustrated in Figure 4.22. Again the percentage errors are mostly positively correlated to the envelope widths. Unlike the finocyl grain design, the radially slotted burning surface is not significantly underestimated for any given envelope width, however the errors do seem to sugest that the integration does again become unstable for very low envelope widths.

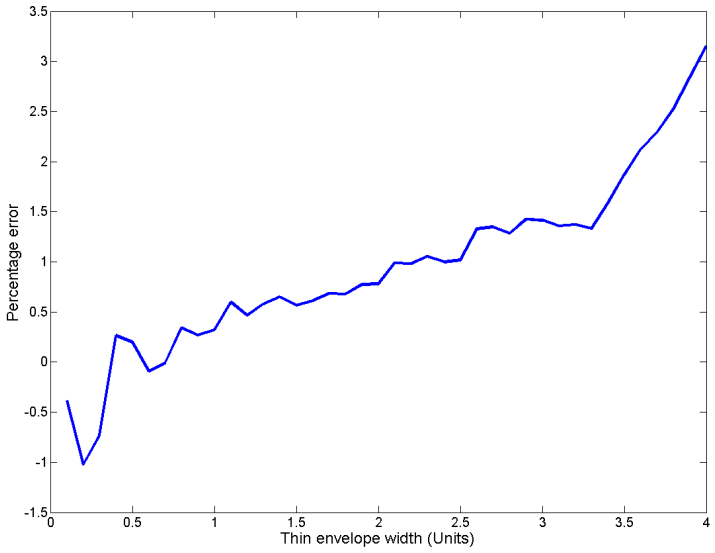


Figure 4.22 – A plot of the percentage error of the MC surface integration of a finocyl motor design

4.2.4 Improvements in efficiency due to stratification and exploitation of symmetry

In this section, the improvements in efficiency and computational cost of MC-integration, as a result of the optimization techniques of Section 2.8.3.6, i.e through stratified MC integration and the exploitation of symmetry in the grain design, is investigated.

The two SRM grain designs utilized in Section 4.2.3, will again be investigated. Recal the domain sizes for the implicit interface representations of the two motor designs are $85 \times 85 \times 700$ and $55 \times 55 \times 494$, respectively. When stratified MC-integration is employed, the number of voxels of the domain that needs to be integrated is reduced from 5 057 500 to 51 130, and from 1 494 350 to 73 512, for the radially slotted and finocyl grains designs, respectively. In the event that the axial symmetry is exploited the numbers can further be reduced to 21 987 and 6 952 voxels.

The sections of each grain design that is not integrated due to their equivalence to previous sections of the grain is highlighted in Figure 4.23 and Figure 4.24, for both grain designs investigated, see also Figure 2.47.

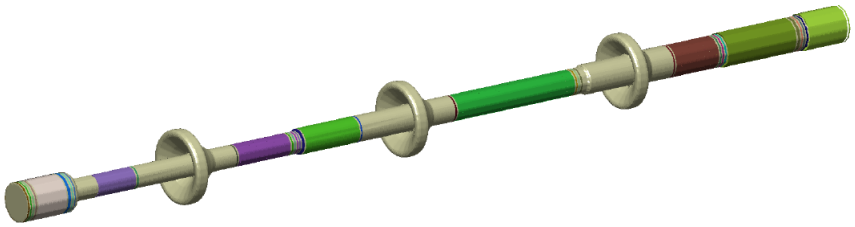


Figure 4.23 – Sections of the radially slotted grain design that are equivalent to previous sections.

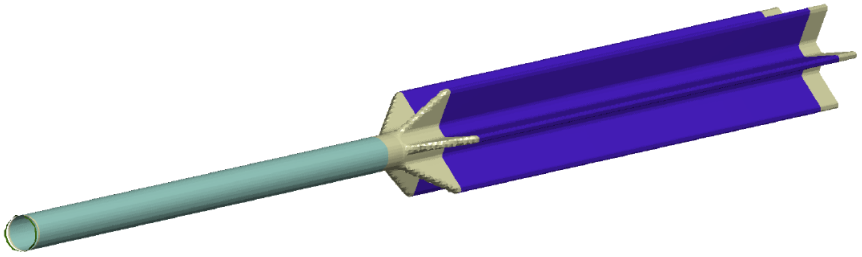


Figure 4.24 – Section of the finocyl grain design that are equivalent to previous sections.

Both grain designs also poses two perpendicular planes of symmetry w.r.t the non axial dimensions, so that a quarter section of the grain can be used to simulate the full design. This implies a further 4-fold reduction in the number of voxels that need to be integrated. The number of integrated voxels now becomes 5497, and 1738, respectively. A summary of the reductions in the number of integrated voxels is given in Table 4.4. Table 4.5 lists the results as a percentage of the total number of voxels in the domain.

Table 4.4 – The number of integrated voxels for various optimization techniques employed, during the geometric evaluation of both a radially slotted and finocyl grain design.

Optimization techniques	Radially slotted	finocyl
None	5 057 500	1 494 350
Stratified domaiin	51 160	73 512
stratified domain with no axial symetry	21 987	6 952
quarter stratified domain with no axial symetry	5 497	1738

Table 4.5 – The percentage of the total number of voxels integrated for various optimization techniques employed, during the geometric evaluation of both a radially slotted, and finocyl grain design.

Optimization techniques	Radially slotted	finocyl
None	100 %	100 %
Stratified domaiin	1.011 %	7.919 %
stratified domain with no axial symetry	0.435 %	0.465 %
quarter stratified domain with no axial symetry	0.109 %	0.116 %

4.3 Grain burnout: Monte-Carlo versus Marching cubes

In Section 2.8.2.2, the calculation of surface areas using a marching cubes method during the burnout phase is discussed. The effects of patches of the surface as it reaches the motor casing cause a jagged area profile, a problem which is overcome by the MC integration techniques of Section 2.8.3. In this section this effect is investigated, and a comparison of burn area profiles, specifically during the burnout phase, as calculated by both the marching cubes integration and MC-integration is done. The radially slotted grain used in the previous section, as illustrated in Figure 4.18, was used as an initial burning surface.

Figure 4.25 illustrates the surface patches that contribute to the calculated burning surface area in the marching cubes integration. The surface patches are deleted in a discrete fashion, inducing the jagged area profile.

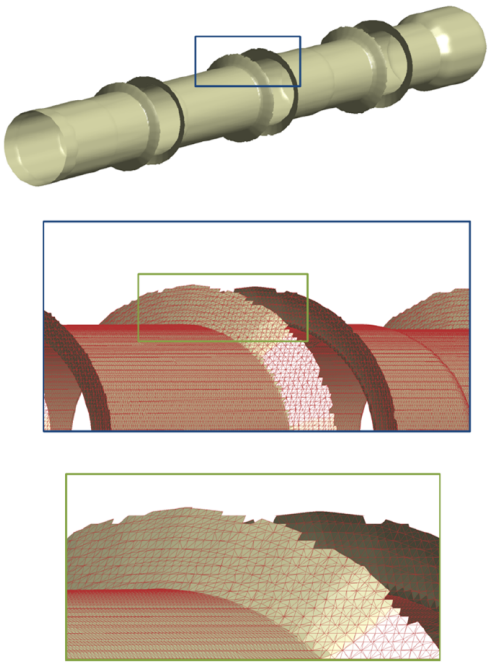


Figure 4.25 – Unburnt patches of a triangulated grian burning surface, used to compute the burning surface area in a marching cube surface integration.

The area profiles for both integration techniques are shown in Figure 4.26.

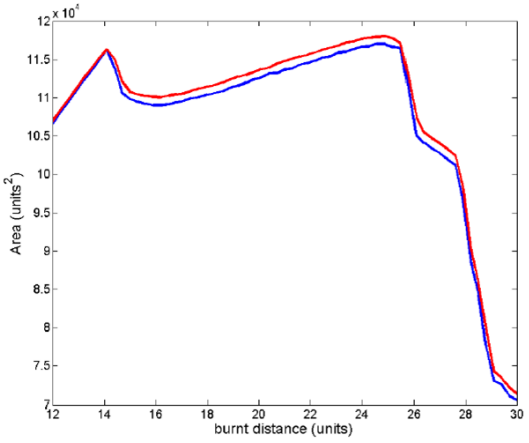


Figure 4.26 – Area profiles of a motor, as calculated by the marching cube integration and MC-integration techniques. The MC-integration results are shown in red and the marching cubes integration results in blue.

Note how the area profile resulting from the marching cubes integration, shows the jagged effect of the integration procedure. This is illustrated clearly in

Figure 4.27, where a more detailed illustration of a section of the area profile is given.

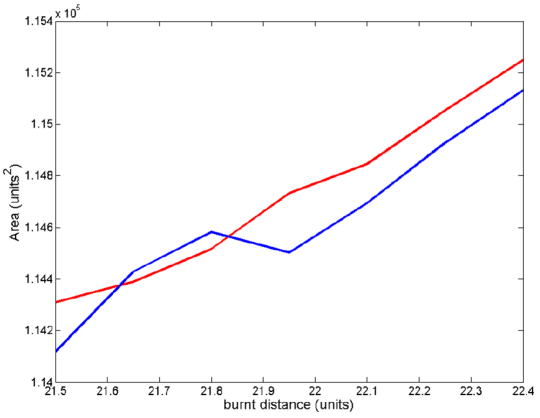


Figure 4.27 – A detailed view of area profiles resulting from marching cubes and MC-integration. The MC-integration results is shown in red and the marching cubes integration results in blue.

4.4 Level set methods

In this section, the validity of the numerical schemes for solving the level set equation, including the conditions for numerical stability, and their ability to select the 'entropy satisfying' solution, is investigated. Three 2-D cases were set up and solved using the first order scheme developed in Section 2.5. Euler time integration was utilized, as this was also the time integration scheme used to couple with the internal ballistics module.

The cases were chosen such that they represent the typical areas of difficulty for numerical front tracking techniques, namely corners, cusps and topological changes.

4.4.1 Corners

The formation of a rarefaction wave is expected around a corner. Figure 4.28 illustrates the initial interface in bold, along with the evolution of the interface for 24 time-steps by the LSM.

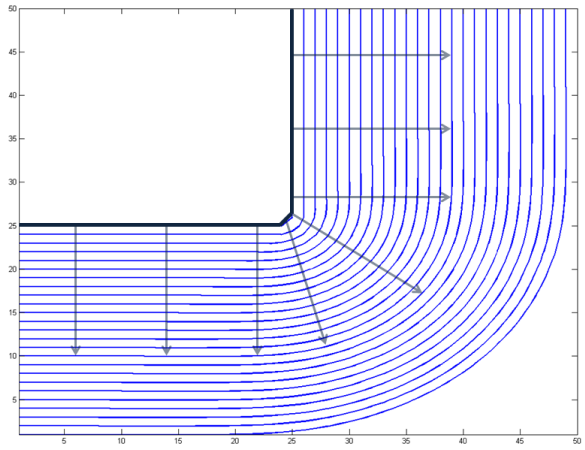


Figure 4.28 – A corner propagated by the LSM. The initial interface is highlighted in bold and the direction of propagation is indicated.

4.4.2 Cusp

The cusp is expected to remain a sharp discontinuity in the interface gradient for all time. Figure 4.29 illustrates the initial interface in bold, along with the evolution of the interface for 24 time steps, by means of the LSM.

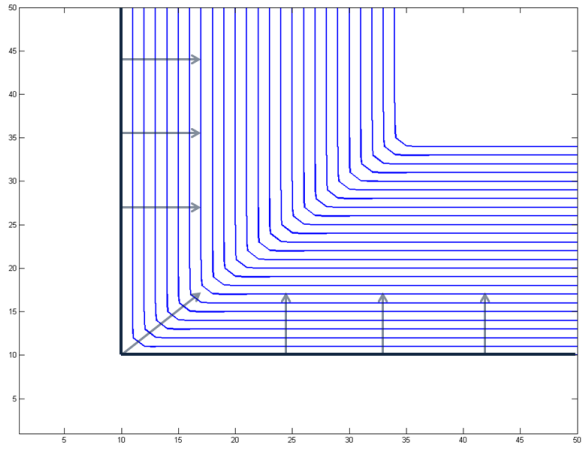


Figure 4.29 – A cusp propagated by the LSM. The initial interface is highlighted in bold and the direction of propagation is indicated.

4.4.3 Topological change

Two circles are advanced outward and is expected merge and form a single closed interface, thus changing the topological state of the total interface. Figure 4.30 illustrates the initial interface in bold, along with the evolution of the interface for 24 time steps, by means of the LSM.

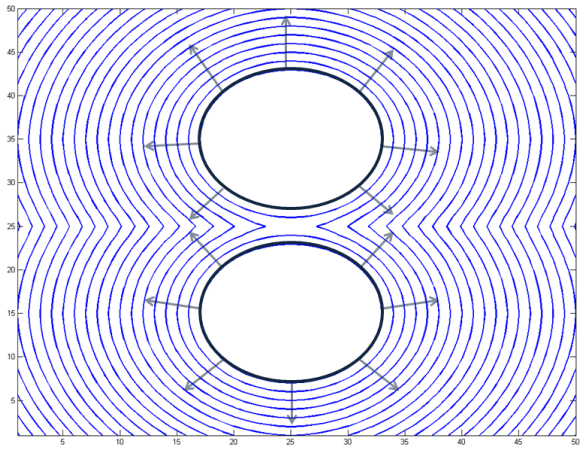


Figure 4.30 – Two neighboring circles propagated by the LSM. The initial interface is highlighted in bold and the direction of propagation is indicated.

The results confirm the LSM's ability to handle the propagation of interfaces that pose the traditional areas of difficulty encountered by numerical interface propagation schemes. The three situations investigated above are all encountered in typical SRM grain designs and illustrates the effectiveness of the LSM for grain regression analysis. Finally in the following section, the results of a coupling of the grain regression procedures, LSM interface propagation and MC-integration, with a Q1-D IB module is given.

4.5 Grain-IB coupling

A test bed experiment was performed using the radially slotted grain illustrated in Figure 4.18. The pressure was was registered at the head-end of the motor and recorded. The results in dimensionless units are illustrated in Figure 4.31

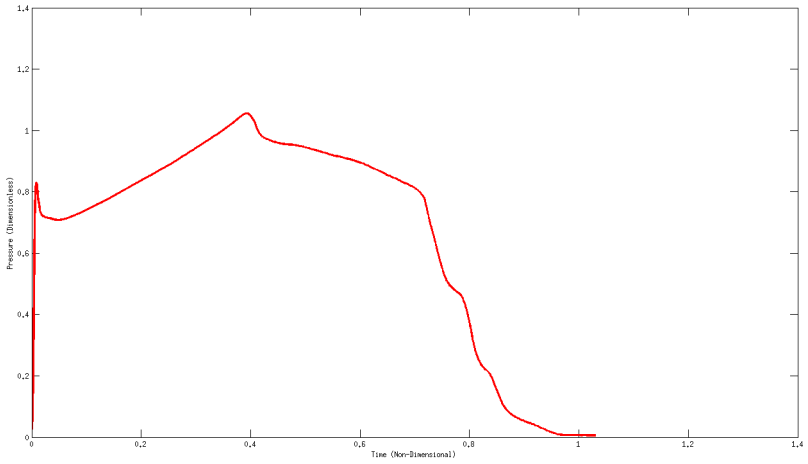


Figure 4.31 – A plot of Pressure vs Time (dimensionless units), for an experimental burn of a radially slotted grain

The grain regression module, made up of the LSM interface advancement and MC-integration, is coupled to the IB solver described in Section 3.1.1. A full motor simulation is performed using both the 'off-line' and 'on-line' (with and without muti-timescales) coupling schemes of Chapter 3, and compared to the results. As noted before, the results do not provide conclusive evidance of the grain regression module, since the accuracy of the IB module is subject to factors such as errosive burning terms and the addition of nozzle erosion to the simulation. They do illustrate the abbility of an IB module, coupled to a numerical grain regression module, to generate pressure-time history predictions that are compareble to experimental results.

4.5.1 'Off-line' IB coupling

Simulations using the 'off-line' scheme was performed and the simulated pressure is compared to the experimental results. Figure 4.32 gives a plot of the resluts obtained.

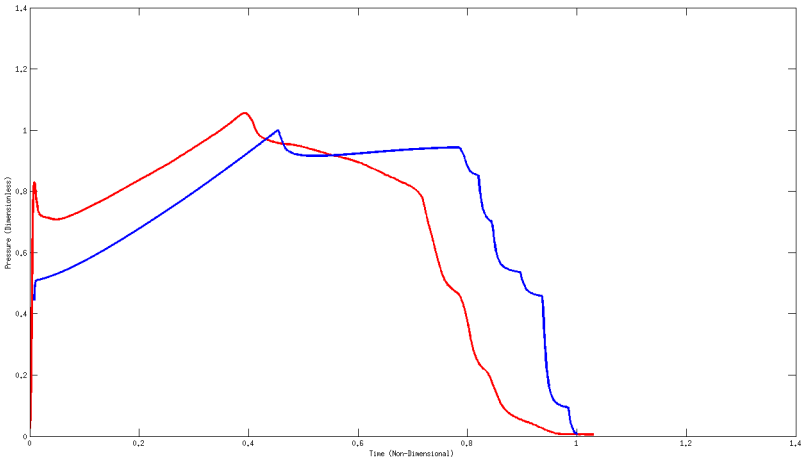


Figure 4.32 – A plot of Pressure vs Time (dimensionless units), for an experimental burn (Red) and an 'off-line' IB simulation (blue), of a radially slotted grain.

The results show the simulation initially underpredicts the pressure and only reaches the burnout phase at a later time than is the case for the experimantal data. Some relatively sharp discontinuieties (when compared to experimental results) is observable in the predicted pressure curve, especially during the burnout phase of the motor operation. This is likely due to the fact that when using the 'off-line' coupling scheme, spatially constant burnrates for the grain sections cause each section to reach the burnout stage adn become fully burnt-out over a single time step. This is not physically correct and the burnout phase should be more accurately captured by using an 'on-line' coupling scheme.

The addition of the multiplicative errosive model descibed in Section 3.1.2 does improve the concurence between simulated predictions and the experimantal data, as shown in Section 4.5.3.

4.5.2 'On-line' IB coupling

4.5.2.1 Single time-scale simulation

The results of the single time-scale simulation is illustrated in Figure 4.33.

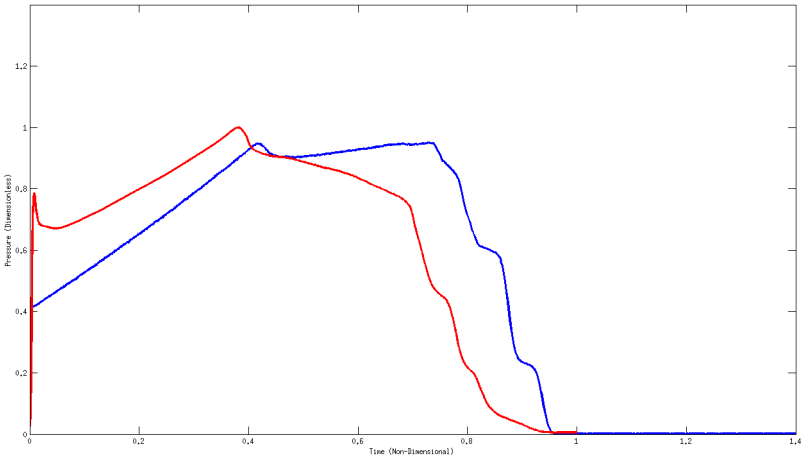


Figure 4.33 – A plot of Pressure vs Time (dimensionless units), for an experimental burn (Red) and an a single time-scale 'on-line' IB simulation (blue), of a radially slotted grain.

The predicted pressure curve does infact show some subjective improvement w.r.t the sharp discontinuities observed during the 'off-line' simulation, and seems to more accurately describe the reality. The number of geometric evaluations of the burning surface during the single time-scale simulation was 1117.

4.5.2.2 Multi time-scale simulation

The results of the multi time-scale simulation is illustrated in Figure 4.34.

The results are comparebly similar to the results obtained by the single time-scale simulation, altough it is true that some smoothing of the pressure curve as a result of the coarser time-scale for geometric evaluations of the burning surface should be present. The number of geometric evaluations during the simulation were 173.

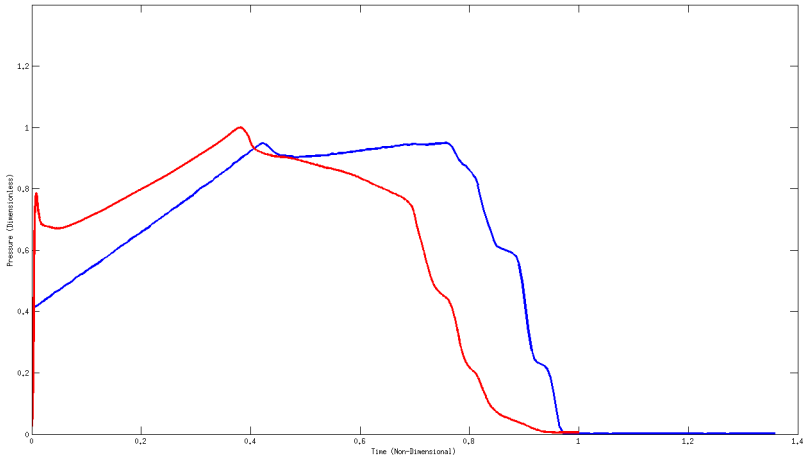


Figure 4.34 – A plot of Pressure vs Time (dimensionless units), for an experimental burn (Red) and an a multi time-scale 'on-line' IB simulation (blue), of a radially slotted grain.

4.5.3 Errosive burning effects

In this section an 'off-line' simulation with the addition of errosive burning terms, in the form of the multiplicative errosive burning model of Section 3.1.2, was performed. The parameters for the errosive burning model was inferred from the experimental data and the results serve solely as a indication of the effects of errosive burning terms. The results are illustrated in Figure 4.35.

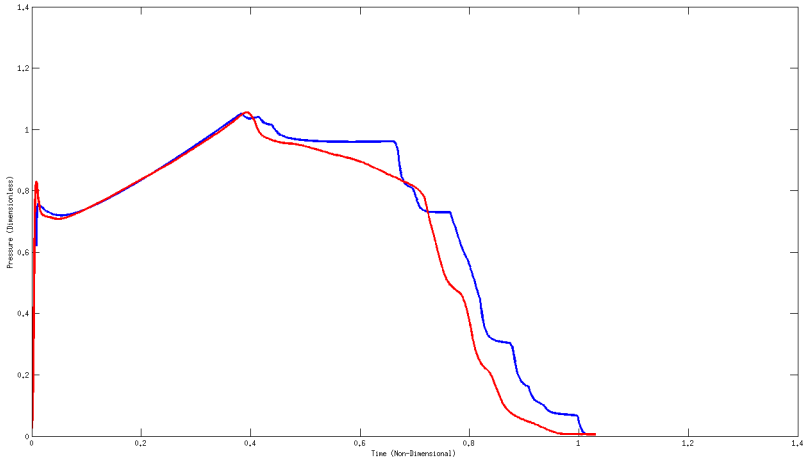


Figure 4.35 – A plot of Pressure vs Time (dimensionless units), for an experimental burn (Red) and an IB simulation with the addition of errosive burning terms (blue), of a radially slotted grain.

Chapter 5

Conclusions and recommendations

5.1 Conclusions

The need for an accurate automated grain regression module, for the purpose of SRM simulation, was identified. The use of implicit interface representations, by means of SDFs, together with novel MC-integration of the implicit functions and the LSM was used in order to achieve the goals set out. The possibility of performing both coupled and uncoupled (with respect to IB simulations) grain regression analysis was maintained.

Some conclusive remarks on each of the above mentioned techniques, incorporated into the grain regression module, are given in the following sections.

5.1.1 SDF implicit interface representation

The use of implicit surface representation by means of an SDF makes it possible to perform grain regression, for complex grain designs where changes in topology may occur, without having to perform time consuming area profile calculations by means of analytical function. It allows for rapid area profiling of a number of designs, and frees the designers to investigate numerous possible solutions to the grain design.

The SDF representation can be obtained by performing scan conversion on a rectangular grid, and can be generated from STL surfaces which are commonly exported by most CAD packages.

Since SDFs of interfaces can be intersected with ease by performing simple Boolean procedures, the SDF representation allows for simulation of the burn out phase with relative ease.

5.1.2 MC-integration

The ability to accurately integrate the surface area of an implicitly defined interface is vital to the proposed grain regression module. Previous approaches, such as the marching-cubes type integration methods described in Section 2.8, are not robust against incorrect topologies and inaccurate, burn-out phase, area integration. The proposed MC-integration methodology provides a robust alternative, whilst maintaining high levels of accuracy at a reasonable computational cost. Methods of exploiting symmetry in a burning surface are also applied to the integration methodology.

The surface area integration of analytical cases, as well as examples of existing SRMs, are performed in order to provide evidence of the validity of the MC-integration technique and show the robustness of the methodology. The comparison of results with analytical models and other numerical integration techniques support the preceding claims. The results lead to the conclusion that MC-integration is applicable to the surface area integration of SDF type implicit surface representations, for the purpose of SRM grain regression analysis. Alternative applications are yet to be explored, however, with the rise in popularity of the LSM for interface advancement, future applications might be plentiful, given the robustness of the technique.

5.1.3 LSM interface advancement

The LSM is shown to be suitable for performing the evolution of a burning surface interface in SRM simulation. The ability to return physically correct

'entropy satisfying' interface evolutions facilitate the posibility of performing grain regression analysis for arbitrarely complex grain designs.

The ability of the LSM to advance an interface at non-uniform spatial speed distributions make direct coupling of the grain regression and IB modules possible. It provides a great step towards simulation, and hopefully a better understanding of phenomena such as erosive burning and 3-D internal flow effects due to the changing velocity. The ability to advance an interface at non-uniform speeds and subsequently perform area and volume integration of the resulting grain surface and perforation might lead to further studies into the effects on burning rates and the combustion process.

5.2 Recommendations for future work

A number of areas of further research have been identified. Work both specifically applicable to SRM simulation, as well as more general work on implicit surface evaluation, is proposed in the following sections.

5.2.1 Automated area profile optimization

For most SRM grain design iterations, the primary goal of the designer is to match a target area profile, in order to achieve a desired thrust curve to meet a set of mission requirements, as discussed in Chapter 1. For some specific types of thrust curves standard generic layouts have been developed. For example a boost sustain curve is typically achieved by a finocyl design.

The finocyl design comprises of two sections that are characterized as a tube and a star design. Let ϕ_{tube} and ϕ_{star} be the implicit representations of γ_{tube}, a tube grain design, and γ_{star}, a star grain design, respectively. By employing the weighted averaging of implicit interfaces as desribed in Section 2.6.2 it is possible to describe a finocyl grain by setting λ equal to the correct weights at each segment along the motor axis.

For instance let the desired grain design have an axial length of say n_{axis} and let,

$$\phi_{finocyl_i} = \lambda_i \phi_{star} + (1 - \lambda)\phi_{tube} \tag{5.2.1}$$

for $i \in \{1, 2, \ldots, n\}$, where $\phi_{finocyl_i}$ denotes the i'th segment of unit length along the motor axis, and λ_i the value of λ at the same position. If λ is set equal to 0 for a section at the front of the grain and allowed to increase, linearly or otherwise, until it reaches 1 and kept equal to 1 for the remainder of the length of the grain, $\phi_{finocyl}$ will be the implicit representation of a traditional boost-sustain curve. An illustration of the interfaces γ_{tube}, γ_{star} and $\gamma_{finocyl}$ along with the values of λ along the various sections of $\gamma_{finocyl}$ is given in Figure 5.1.

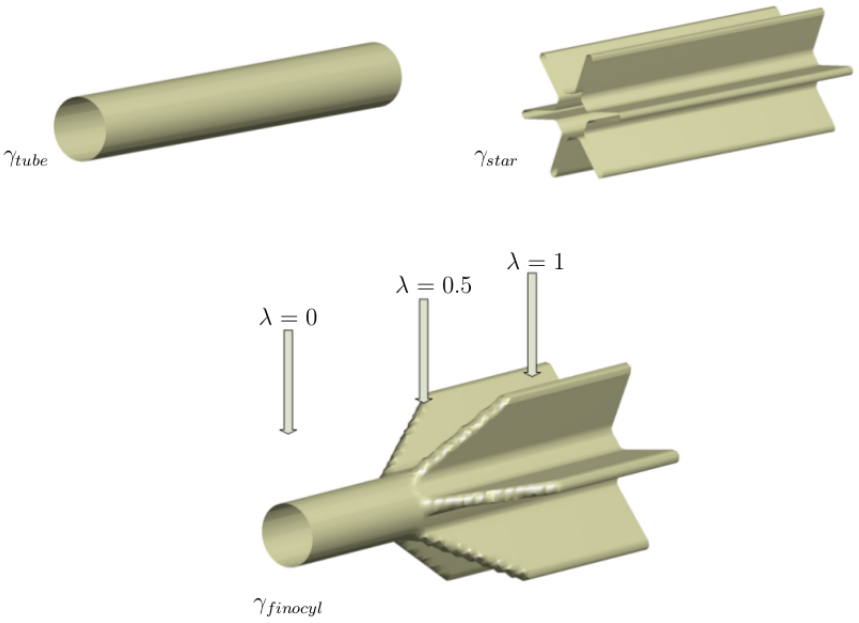


Figure 5.1 – An illustration of the weighted averaging of γ_{tube} and γ_{star} in order to define $\gamma_{finocyl}$. The weights of the averaging along the axis is given by λ.

Since the grain regression module described in this work is capable of performing a geometric evaluation of an entire regression of $\phi_{finocyl}$ and return the resulting area profile, it might be possible to perform a Latin-Hypercube type experiment on the parameter λ and optimize the grain design $\phi_{finocyl}$ to approximate a given area profile.

This could possibly further reduce the number of design iterations and therefore the time it takes to develop a novel grain design for a given mission.

5.2.2 3-D grid generation

It was previously stated that the coupling to a 3-D IB-solver is to be facilitated by the grain regression module. In order to perform fully 3-D IB simulations of an SRM internal flow-field, an efficient and robust method of grid generation, for the purpose of discretizing the evolving internal geometry of an SRM at each point in time during the operational phase, is required. Two possible methods, not excluding other possibilities, have been identified. The first being a VOF approach, and the second, an automated grid generator that relies on implicit interface representations similar to the grid generator described by Persson [61].

5.2.2.1 VOF grid generation

The VOF grid approach is simply a weighting of cells such that the fraction of a cell that forms part of the flow field is designated by the weight assigned to it. In the case of SRM simulation the flow field corresponds to the area inside the burning interface and so the VOF method for interface advancement described in Section 2.4.1 does exactly that. Furthermore, the MC-integration described for the evaluation of geometric in the grain regression module, approximates the fraction of each cell in the domain that falls inside the burning surface interface.

The rectangular grid framework on which the grain regression procedure is based could possibly be employed directly for IB-simulation, using the VOF technique to describe the boundaries of the internal flow-field. This would mean that, in essence, no additional grid generation is needed to perform the coupling of of the grain regression.

5.2.2.2 Implicit grid generation

In his doctoral thesis Persson [61] introduces a method of grid generation based on implicit interface representation. The basic procedure relies on initializing a structure of points and edges, similar to a truss structure, and letting the edges act as springs and the points as joints. By enforcing a force equilibrium in the edges or 'springs', the points are transported into a equidistant state. The implicit representation provides the exterior forces or edge lengths from the boundary or interface. The edges to the boundary are necessarily perpendicular on the interface since the implicit representation returns the minimum distance to the interface.

In his thesis, Person uses a Delaunay triangulation [62] as an initial structure for the points and edges, however, any initial structure could be utilized within the same philosophy. Another advantage would be that at each time step of the SRM simulation, the mesh structure from the superseding time step could be used as initialization of the current mesh generating procedure. This would eliminate the need to perform multiple initializations for the grid generation procedure, and since the change in geometry between time steps would necessarily be small. The amount of iteration of solving the force equilibrium equations should be relatively small.

5.2.3 Higher-order shock capturing interpolation schemes

The MC-integration techniques described in Section 2.8.3, relies on interpolation of random points to a discrete implicit interface representation or signed distance function. Tri-linear interpolation was employed in order to determine the position of a random point with respect to the zero-level-set or interface. The use of higher order interpolation schemes would be beneficial for smooth regions of the represented interface, allowing more accurate approximations to the actual interface. There would however be a significant error introduced at sharp discontinuities, or shocks in the interface, due to smoothing as a result of the high order interpolation.

A possible solution might be to perform a weighted interpolation, analogue to the weighted stencil of the weighted essentially non-oscillatory or WENO integration schemes, developed for integration of flow fields that may contain shock discontinuities.

Simply put, the interpolation scheme would utilize a high order procedure in areas where it is determined that the gradient of the SDF does not contain discontinuities, i.e. the gradient in a specific location of the neighbourhood of the point being interpolated does not vary greatly from any other location within the local neighbourhood. Where the neighbourhood is of the size equivalent to the order of the integration scheme multiplied by the grid spacing of the discrete SDF domain. If this is not the case the interpolation scheme systematically reverts to a lower order interpolation until the discrepancies in the gradient of the SDF within the appropriate neighbourhood is such that a chosen criteria for accurate interpolation is met. Such criteria might for example be the absolute difference in local gradients for any two locations within the neighbourhood must be smaller than a chosen threshold.

Consider for example, the implicit interface illustrated in Figure 5.2. Point $\vec{\mathbf{x}}_2$ lies in the center of a 5×5 neighbourhood that falls on a smooth section of the implicit function, where as a 5×5 neighbourhood centered around point $\vec{\mathbf{x}}_1$ contains a sharp change in the gradient.

If the neighbourhood around $\vec{\mathbf{x}}_1$ is decreased to size 3×3, then it no longer contains the sharp gradient change and the interpolation can proceed, as illustrated in Figure 5.3.

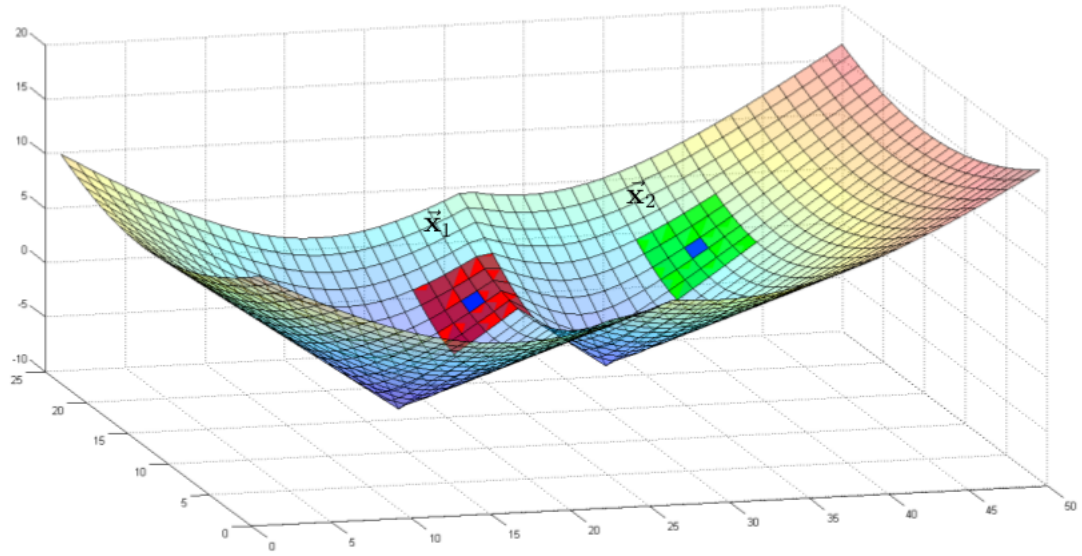


Figure 5.2 – Two points with their neighbourhoods highlighted on an implicit interface representation.

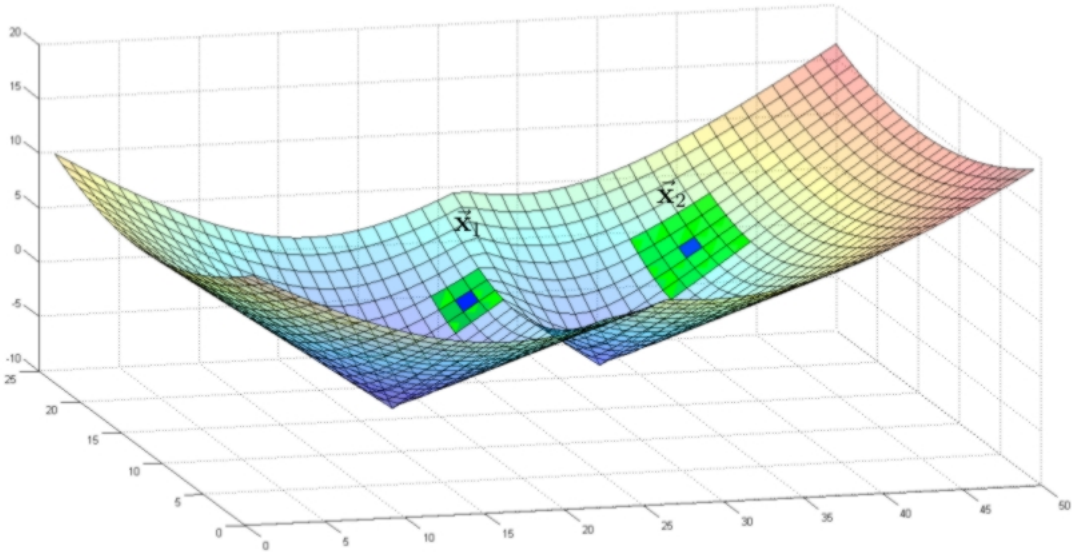


Figure 5.3 – Two points with their neighbourhoods highlighted on an implicit interface representation.

Bibliography

[1] Mike Gruntman. *Blazing the trail: the early history of spacecraft and rocketry*. AIAA, 2004.

[2] George P Sutton and Oscar Biblarz. *Rocket propulsion elements*. Wiley. com, 2011.

[3] Richard Nakka. Richard nakka's experimental rocketry website. http://www.nakka-rocketry.net/, 2004.

[4] R.H.W Waesche and J Wenograd. Calculation of solid-propellant burning rates from condensed-phase decomposition kinetics. *Combustion, Explosion and Shock Waves*, 36(1):125–134, 2000.

[5] David R Greatrix. Transient burning rate model for solid rocket motor internal ballistic simulations. *International Journal of Aerospace Engineering*, 2008, 2007.

[6] Matthew R Umbel. An exact geometric analysis of the generalized anchor grain configuration. 44th AIAA/ASME/SAE/ASEE Joint Porpulsion Conference and Exhibits, Hartford, CT, 21-23 July 2008. AIAA Paper No. AIAA 2008-4697, 2008.

[7] JT Lamberty. A report on the grain design and internal ballistic module of the improved solids performance program. In *AIAA 19th Aerospace Sciences Meeting, AIAA Paper*, volume 34, page 1981, 1981.

[8] Michael A Willcox, M Quinn Brewster, KC Tang, and D Scott Stewart. Solid propellant grain design and burnback simulation using a minimum distance function. *Journal of propulsion and power*, 23(2):465–475, 2007.

[9] Frederic Dauch and Dominique Ribereau. A software for srm grain design and internal ballistics evaluation, pibal. 2002.

[10] Xiangmin Jiao. Face offsetting: A unified approach for explicit moving interfaces. *Journal of computational physics*, 220(2):612–625, 2007.

[11] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 79(1):12–49, 1988.

[12] Enrico Cavallini. *Modeling and Numerical Simulation of Solid Rocket Motors Internal Ballistics*. PhD thesis, 2010.

[13] James Albert Sethian. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, volume 3. Cambridge university press, 1999.

[14] William F Noh and Paul Woodward. Slic (simple line interface calculation). In *Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics June 28–July 2, 1976 Twente University, Enschede*, pages 330–340. Springer, 1976.

[15] Alexandre Joel Chorin. Flame advection and propagation algorithms. *Journal of Computational Physics*, 35(1):1–11, 1980.

[16] Bruno Lafaurie, Carlo Nardone, Ruben Scardovelli, Stéphane Zaleski, and Gianluigi Zanetti. Modelling merging and fragmentation in multiphase flows with surfer. *Journal of Computational Physics*, 113(1):134–147, 1994.

[17] Cyril W Hirt and Billy D Nichols. Volume of fluid (vof) method for the dynamics of free boundaries. *Journal of computational physics*, 39(1):201–225, 1981.

[18] David Adalsteinsson and James A Sethian. The fast construction of extension velocities in level set methods. *Journal of Computational Physics*, 148(1):2–22, 1999.

[19] James A Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595, 1996.

[20] Ravi Malladi, James A Sethian, and Baba C Vemuri. Shape modeling with front propagation: A level set approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(2):158–175, 1995.

[21] Mark Sussman, Peter Smereka, and Stanley Osher. A level set approuch for computing solutions to incompressible two-phase flow. *Journal of Computational physics*, 114(1):146–159, 1994.

[22] Liron Yatziv, Alberto Bartesaghi, and Guilermo Sapiro. O (¡ i¿ n¡/i¿) implementation of the fast marching algorithm. *Journal of computational physics*, 212(2):393–399, 2006.

[23] Peter Schwartz and Phillip Colella. A second-order accurate method for solving the signed distance function equation. *Commun. Appl. Math. Comput. Sci. v5 i1*, pages 81–97, 2010.

[24] Seongjai Kim. An \calo(n) level set method for eikonal equations. *SIAM journal on scientific computing*, 22(6):2178–2193, 2001.

[25] Hongkai Zhao. A fast sweeping method for eikonal equations. *Mathematics of computation*, 74(250):603–627, 2005.

[26] TOMÁŠ Oberhuber. Numerical recovery of the signed distance function. In *Czech-Japanese Seminar in Applied Mathematics, Prague, Czech Republic*, pages 148–164, 2004.

[27] David Eberly. Distance between point and triangle in 3d. *Magic Software, http://www. magic-software. com/Documentation/pt3tri3. pdf*, 1999.

[28] Bradley A Payne and Arthur W Toga. Distance field manipulation of surface models. *IEEE Computer graphics and applications*, 12(1):65–71, 1992.

[29] J Andreas Bærentzen and Henrik Aanæs. Generating signed distance fields from triangle meshes. *Informatics and Mathematical Modeling, Technical University of Denmark, DTU*, 20, 2002.

[30] A Guezlec. meshsweeper: dynamic point-to-polygonal mesh distance and applications. *Visualization and Computer Graphics, IEEE Transactions on*, 7(1):47–61, 2001.

[31] Sean Mauch. A fast algorithm for computing the closest point and distance transform. *Go online to http://www. acm. caltech. edu/seanm/software/cpt/cpt. pdf*, 2000.

[32] Stanley Osher and Ronald Fedkiw. *Level set methods and dynamic implicit surfaces*, volume 153. Springer, 2003.

[33] Anna-Karin Tornberg and Björn Engquist. Numerical approximations of singular source terms in differential equations. *Journal of Computational Physics*, 200(2):462–488, 2004.

[34] Björn Engquist, Anna-Karin Tornberg, and Richard Tsai. Discretization of dirac delta functions in level set methods. *Journal of Computational Physics*, 207(1):28–51, 2005.

[35] Peter Smereka. The numerical approximation of a delta function with application to level set methods. *Journal of Computational Physics*, 211(1):77–90, 2006.

[36] Chohong Min and Frédéric Gibou. Geometric integration over irregular domains with application to level-set methods. *Journal of Computational Physics*, 226(2):1432–1443, 2007.

[37] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM Siggraph Computer Graphics*, volume 21, pages 163–169. ACM, 1987.

[38] Graham M Treece, Richard W Prager, and Andrew H Gee. Regularised marching tetrahedra: improved iso-surface extraction. *Computers & Graphics*, 23(4):583–598, 1999.

[39] John C Anderson, Janine C Bennett, and Kenneth I Joy. Marching diamonds for unstructured meshes. In *Visualization, 2005. VIS 05. IEEE*, pages 423–429. IEEE, 2005.

[40] Tim Poston, Tien-Tsin Wong, and Pheng-Ann Heng. Multiresolution isosurface extraction with adaptive skeleton climbing. In *Computer Graphics Forum*, volume 17, pages 137–147. Wiley Online Library, 1998.

[41] C Mielack. Isosurfaces, 2009.

[42] Gregory M Nielson and Bernd Hamann. The asymptotic decider: resolving the ambiguity in marching cubes. In *Proceedings of the 2nd conference on Visualization'91*, pages 83–91. IEEE Computer Society Press, 1991.

[43] Michael A Willcox, M Quinn Brewster, KC Tang, and D Scott Stewart. Solid propellant grain design and burnback simulation using a minimum distance function. *Journals of propulsion and power*, 23(2):465–475, 2007.

[44] Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.

[45] Reuven Y Rubinstein. *Simulation and the Monte Carlo method*. Wiley, 1981.

[46] D Edwards. Monte-carlo integration. *Google Search*, 2012.

[47] Chin-Shung Yang, Szu-Pyng Kao, Fen-Bin Lee, and Pen-Shan Hung. Twelve different interpolation methods: A case study of surfer 8.0. In *Proceedings of the XXth ISPRS Congress*, volume 35, pages 778–785, 2004.

[48] Competence center for inovative manufacturing. Cad users guide to stl export.

[49] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1):3–30, 1998.

[50] Stephen K. Park and Keith W. Miller. Random number generators: good ones are hard to find. *Communications of the ACM*, 31(10):1192–1201, 1988.

[51] P Goutam and M Subhamoy. Rc4 stream cipher and its variantsa, 2012.

[52] Pierre L'Ecuyer. Uniform random number generators: a review. In *Proceedings of the 29th conference on Winter simulation*, pages 127–134. IEEE Computer Society, 1997.

[53] Harald Niederreiter. *Quasi-Monte Carlo Methods*. Wiley Online Library, 1992.

[54] D Scott Stewart, Kung-Chyun Tang, Sunhee Yoo, M Quinn Brewster, and Igor R Kuznetsov. eoretical an. 2004.

[55] Antony Jameson. Time integration methods in computational aerodynamics. *AFOSR Work Shop on Advances and Challenges in Time-Integration of PDEs. Providence, RI*, 2003.

[56] JT Lamberty. A report on the grain design and internal ballistic module of the improved solids performance program. In *AIAA 19th Aerospace Sciences Meeting, AIAA Paper*, volume 34, page 1981, 1981.

[57] Qunzhen Wang. Development of erosive burning models for cfd predictions of solid rocket motor internal environment. *AIAA Paper*, 4809:2003, 2003.

[58] B McDonald. *The developement of an erosive burning model for solid rocket motors using direct numerical simulation*. PhD thesis, Gorgie institute of technology, 2004.

[59] RHW Waesche and J Wenograd. Calculation of solid-propellant burning rates from condensed-phase decomposition kinetics. *Combustion, Explosion and Shock Waves*, 36(1):125–134, 2000.

[60] Victor Topalian, Ju Zhang, Thomas L Jackson, and Amir H G. Isfahani. Numerical study of erosive burning in multidimensional solid propellant modeling. *Journal of Propulsion and Power*, 27(4):811–821, 2011.

[61] Per-Olof Persson. *Mesh generation for implicit geometries*. PhD thesis, Citeseer, 2004.

[62] Boris Delaunay. Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7(793-800):1–2, 1934.

[63] Matthew A Grayson. The heat equation shrinks embedded plane curves to round points. *Journal of Differential geometry*, 26(2):285–314, 1987.

[64] Wichard Sulwald, Francois Smit, Adriaan Steenkamp, and Werner Rousseau. Solid rocket motor grain burn back analysis using level set methods and monte-carlo volume integration. *Preceedings of the 49th Joint Propulsion conference*, 2013.

[65] Werner Rousseau, Francois Steyn, Wichard Sulwald, Erhart De Kock, Francois Smit, and Hansie Knoetze. Rapid solid rocket motor design. *Preceedings of the 49th Joint Propulsion conference*, 2013.

Appendix A

SDF generation

A.1 Right-hand rule convention

The right hand rule, first used in the study of electro-magnetics, is a popular convention in both mathematics and physics for defining the direction of three vectors perpendicular to each other. This can also be applied to finding the positive normal direction of a plane, by using two non-parallel vectors on the plane and finding a third vector perpendicular to both. This is done by defining three sequential verticies, placed anti-clockwise on the plane as viewed from the positive side of the plane. Figure A.1 illustrates the vertices $\vec{\mathbf{v}}_1$, $\vec{\mathbf{v}}_2$ and $\vec{\mathbf{v}}_3$, which satisfy the afore mentioned.

Once the verticies are defined, the normal vector $\vec{\mathbf{n}}$ is simply found by,

$$\vec{\mathbf{n}} = (\vec{\mathbf{v}}_2 - \vec{\mathbf{v}}_1) \times (\vec{\mathbf{v}}_3 - \vec{\mathbf{v}}_2). \tag{A.1.1}$$

A.2 Sign calculation

In Section 2.7.4, the generation of a discrete SDF from an STL file is discussed. Further detail of the calculation of the sign awarded to points that were scan converted by polygons to find points possibly closest to edges or vertices, is given in the following section.

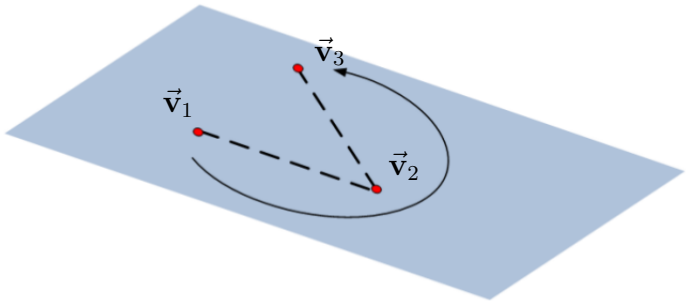


Figure A.1 – Three sequential vertices placed in an anti-clockwise direction on a plane.

Edge: First an edge of an STL surface is considered. Recall that the polyhedron for scan conversion of the points, in the case of an edge, is a triangular prism, as illustrated in Figure A.2.

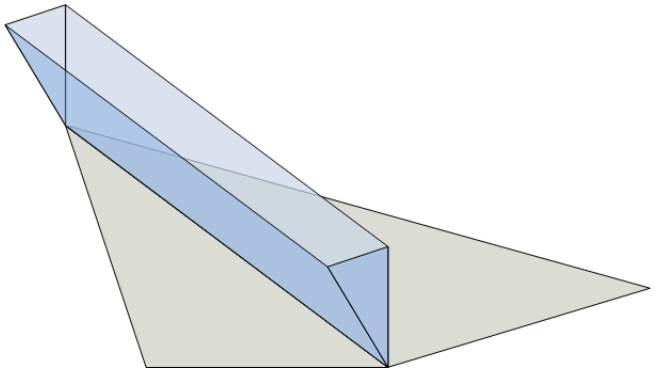


Figure A.2 – An illustration of the polyhedron used to scan convert points closest to an edge of an STL surface.

Depending on the local curvature of the surface at the edge the polyhedron will either lie inside or outside the surface. If the surface is convex the polyhedron lies outside and if the surface is concave, inside, as illustrated in Figure A.3.

Determining whether the surface is convex or concave at the edge relies on the relation between two distances. Refer to Figure A.4, where an edge is defined by points $\vec{\mathbf{x}}_1$ and $\vec{\mathbf{x}}_2$. The two faces that share the edge are completed by vertices $\vec{\mathbf{x}}_3$, and $\vec{\mathbf{x}}_4$, and their normals defined by $\vec{\mathbf{n}}_{123}$ and $\vec{\mathbf{n}}_{142}$, respectively.

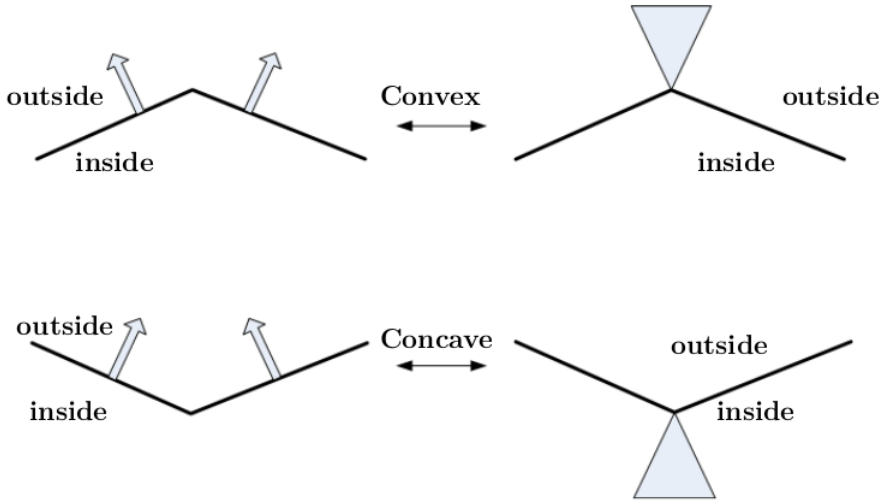


Figure A.3 – An illustration of a polyhedron outside a convex surface and inside a concave surface.

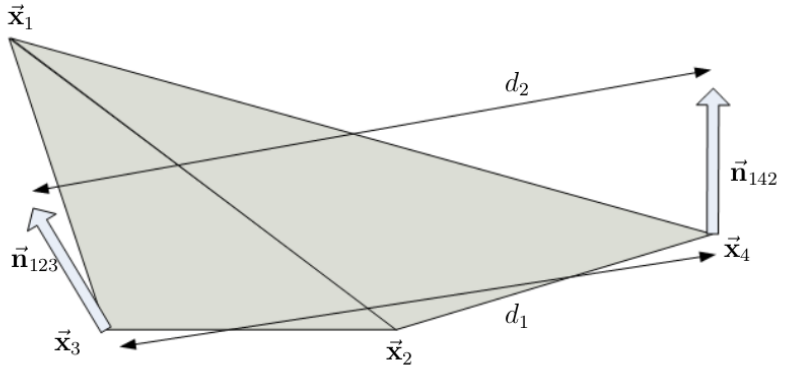


Figure A.4 – Two distances used to determine concavity over an edge.

Let

$$d_1 = |\vec{\mathbf{x}}_3 - \vec{\mathbf{x}}_4| \tag{A.2.1}$$

and

$$d_2 = |(\vec{\mathbf{n}}_{123} + \vec{\mathbf{x}}_3) - (\vec{\mathbf{n}}_{124} + \vec{\mathbf{x}}_4)| \tag{A.2.2}$$

From Figure A.5 it can be deduced that if $d_1 > d_2$, the surface curvature around the edge will be convex and if $d_2 > d_1$, the surface is concave.

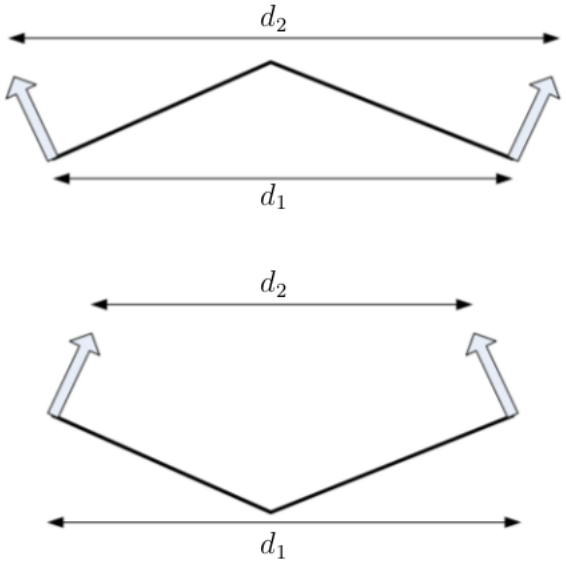


Figure A.5 – The distances d_1 and d_2, for concave and convex surfaces.

Vertex: Recall points possibly closest to a vertex $\vec{\mathbf{x}}_1$ of an STL surface is scan converted by a polygonal pyramid, as illustrated in Figure A.6.

First an averaged normal $\vec{\mathbf{n}}_{vert}$ is calculated as the average of all the face normals of the patches that are common to the vertex. It should be noted that the normals are normalized unit normals of length 1 units. For each of the patches a procedure similar to described in the case of an edge is followed. Refer to Figure A.7, the distances d_1 and d_2 are calculated as,

$$d_1 = |\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_{2,n}|, \tag{A.2.3}$$

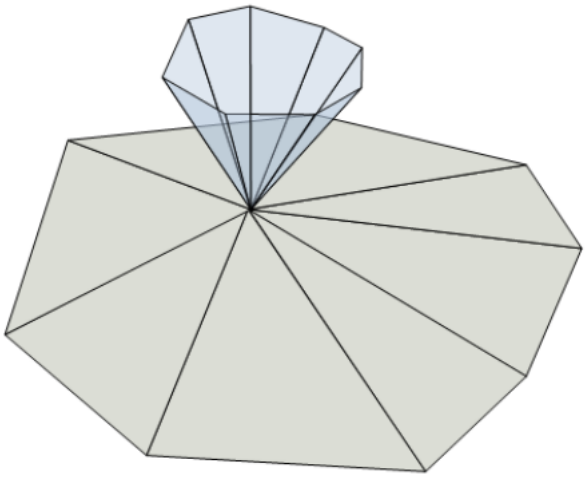


Figure A.6 – Polygonal pyramid used for scan-conversion of points possibly closest to a vertex.

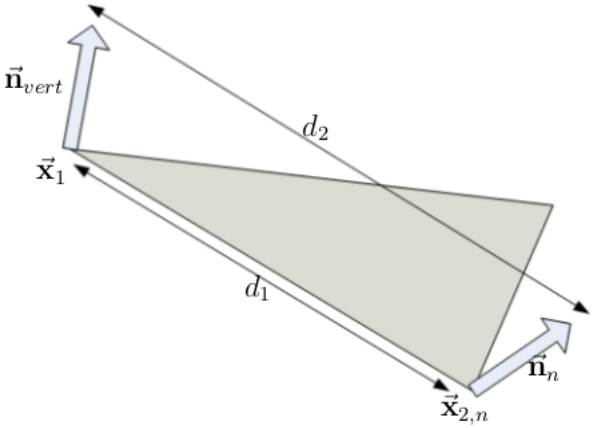


Figure A.7 – The distances d_1 and d_2, as defined for a vertex and a patch of an STL surface.

and

$$d_2 = |(\vec{\mathbf{x}}_1 + \vec{\mathbf{n}}_{vert}) - (\vec{\mathbf{x}}_{2,n} + \vec{\mathbf{x}}_n)|, \tag{A.2.4}$$

where $\vec{\mathbf{x}}_{2,n}$ is a vertex on the patch not equal to the shared vertex $\vec{\mathbf{x}}_1$.

From Figure A.8 we can deduce that it, if $d_1 > d_2$ for all the patches that share

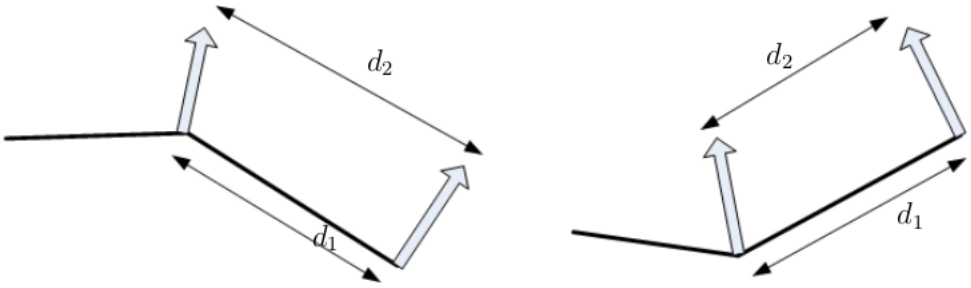


Figure A.8 – The distances d_1 and d_2, for convex and concave surfaces.

$\vec{\mathbf{x}}_1$, the polyhedron will lie inside the surface. If $d_1 < d_2$ for all the patches that share $\vec{\mathbf{x}}_1$, the polyhedron will lie outside the surface. Otherwise the vertex will be on a saddle point, and there will not be points inside the polyhedron described for $\vec{\mathbf{x}}_1$.

Appendix B

Entropy satisfying schemes for interface propogation

The development of an 'entropy satisfying' physically relevant interface advancement scheme was conducted by Osher and Sethian [11]. The connection between interfaces, entropy and a viscous limit solution of hyperbolic conservation laws is discussed in detail by Sethian [13], the following sections are a summation of the dicussion.

B.1 The role of entropy conditions

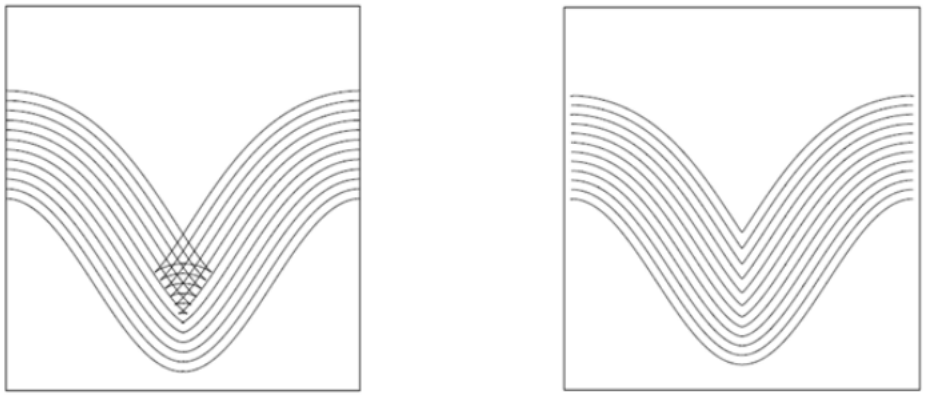
Consider an initial interface represented by a parameterized smooth cosine curve,

$$\gamma(0) = (-s, [1 + \cos(2\pi s)]2), \tag{B.1.1}$$

propagating with speed $V = 1$. From equations (2.2.2) and (2.2.3), an exact solution of the problem at a time t is given by,

$$x(s,t) = \frac{(\partial y/\partial s)(s,0)}{((\partial x/\partial s)^2(s,0) + (\partial y/\partial s)^2(s,0))^{1/2}} t + x(s,0) \tag{B.1.2}$$

$$y(s,t) = \frac{(\partial x/\partial s)(s,0)}{((\partial x/\partial s)^2(s,0) + (\partial y/\partial s)^2(s,0))^{1/2}} t + y(s,0) \tag{B.1.3}$$



(a) (b)

Figure B.1 – Swallow tail and entropy satisfying solutions to an advancing cosine curve interface.

Figure B.1 shows how a shock develops. It becomes unclear how to determine the normal direction at the shock, and therefore, how to continue the propagation. The shape of the propagated interface beyond the formation of the shock depends on the nature of the interface. If the interface is regarded to be a geometric curve, the solution might be the formation of a 'swallowtail' as shown in Figure B.1 (a), where the front passes through itself. This is the solution given by equation (2.2.2) and (2.2.3). If however, the interface is regarded as moving boundary, separating two regions, the front at time t should only consist of points a distance t from the initial interface, as in Figure B.1(b). This is known as the Huygens principle construction and can be said to remove the 'swallowtail' from the solution.

The entropy solution illustrated in Figure B.1 (b) is found by invoking an 'entropy condition' which states: If the interface is seen as a burning flame front, then once a particle is burnt it remains in a burnt state for all time. The connection to the notion of entropy arises from the fact that by removing part of the solution some information of the initial interface is lost and the problem becomes irreversible.

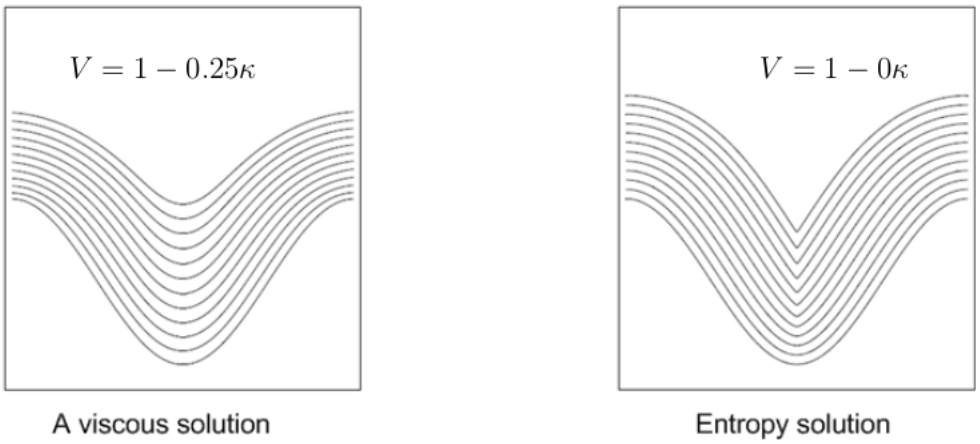


Figure B.2 – A viscous solution to the advancing cosine curve and its limit, the entropy solution.

B.2 Curvature and the viscous limit

Define the curvature κ of a parameterized interface $\gamma(s)$ as,

$$\kappa = \frac{(\partial^2 y/\partial s^2)(\partial x/\partial s) - (\partial^2 x/\partial s^2)(\partial y/\partial s)}{((\partial x/\partial s)^2 + (\partial y/\partial s)^2)^{3/2}} \tag{B.2.1}$$

Consider again the cosine interface, equation (B.1.1), and let the speed with which the interface propagates now be given by $V = 1 - \epsilon\kappa$. It has been proved (Grayson [63]) that for $\epsilon > 0$, a smooth initial curve propagated at a speed $V = 1 - \epsilon\kappa$ will remain smooth for all time. The following observation, illustrated in Figure B.2, is central to the level set approach.

For a smooth initial interface γ, let

- $\gamma^\epsilon(t)$ be the family of the curves obtained by propagating γ with a speed function $V = 1 - \epsilon\kappa$.
- $\gamma_{constant}(t)$ be the family of curves obtained by propogating γ with a speed function $V = 1$.

Then at any time t

$$\lim_{\epsilon \to 0} \gamma^\epsilon(t) = \gamma_{constant}(t). \tag{B.2.2}$$

This is known as the viscous limit. The reason for calling it the viscous limit can be seen by a consideration of the hyperbolic conservation laws, and their link to interface propagation. A hyperbolic conservation law is an equation of the form,

$$\partial\mu/\partial t + \partial G(\mu)/\partial x = 0 \tag{B.2.3}$$

The solution to equation (B.2.3) can develop shock discontinuities. For example Burger's equation, which describes the motion of a compressible fluid. If, however, a viscosity term, $\epsilon(\partial^2\mu/\partial x^2)$, is added to the right hand side of equation (B.2.3) the solution remains smooth for all time.

Now let an initial interface be given by a graph of a function $f(x)$, and suppose the interface remains a function for all time. Let μ now be the height of the propagating interface at time t, so that $\mu(x,0) = f(x)$. The tangent at a point (x,μ) is $(1,(\partial\mu,\partial x))$. The change in height μ is related to the speed V at which the interface propagates in its normal direction by

$$\frac{\mu}{V} = \frac{(1+(\partial\mu/\partial x)^2)^{1/2}}{1}. \tag{B.2.4}$$

The equation of motion becomes

$$\frac{\partial\mu}{\partial t} = V(1+(\frac{\partial\mu}{\partial x})^2)^{1/2}. \tag{B.2.5}$$

Using $V = 1-\epsilon\kappa$ and the formula $\kappa = -(\partial^2\mu/\partial x^2)/(1+(\partial\mu/\partial x)^2)^{3/2}$, equation (B.2.5) becomes

$$\frac{\partial\mu}{\partial t} - (1+(\partial\mu/\partial x)^2)^{1/2} = \epsilon\frac{\partial^2\mu/\partial x^2}{(1+(\partial\mu/\partial x)^2)}. \tag{B.2.6}$$

Differentiating both sides of the equation yields an evolution equation for the gradient $\frac{\partial\mu}{\partial x}$ of the interface,

$$\frac{\partial\mu}{\partial x\partial t} + \frac{-\partial(1+(\partial\mu/\partial x)^2))^{1/2}}{\partial x} = \frac{\partial(\epsilon(\partial^2\mu/\partial x^2)/(1+(\partial\mu/\partial x)^2)}{\partial x} \tag{B.2.7}$$

Now, if $\frac{\partial \mu}{\partial x}$ is substituted with μ, the equation for the change in height of the function f looks like a hyperbolic conservation law,

$$\frac{\partial \mu}{\partial t} + \frac{\partial (G(\mu))}{\partial x} = \epsilon \frac{\partial^2 \mu}{\partial x^2}, \tag{B.2.8}$$

with $G(\mu) = (1+\mu^2)^{1/2}$. This makes it possible to exploit the theory developed for hyperbolic conservation laws to develop accurate schemes for advancing interfaces. All that is required is to describe an interface by means of a graph of a function. This can be done for any interface following the implicit front representation philosophy of the LSM, introduced by Sethian and Osher [11].

Appendix C

Published work

The following papers were presented at the 2013 AIAA Joint Propulsion Conference in San Jose, USA. The contents of the work presented in this thesis is partially included to the papers.

Solid rocket motor grain burn back analysis using level set methods and Monte-Carlo volume integration

Wichard Sullwald[1] and Francois Smit[2]
Stellenbosch University, Stellenbosch, South Africa, 7600

Adriaan Steenkamp[3]
Flamengro, Pretoria, South Africa, 0157

and

Werner Rousseau[4]
Rheinmetal Denel Munitions, Somerset West, 7130

We employ the level set method to perform coupled grain burn back analysis in solid rocket motor simulations. A method of generating a signed distance function from STL files for the initialization of the level set function is given. Monte-Carlo integration techniques are applied to calculate the burning surface and port area parameters from the evolved implicit representation of the burning surface. Multiple timescales are used for improved efficiency of the grain regression coupling with the internal ballistics code.

Nomenclature

γ	=	general interface / grain burning surface
$\boldsymbol{V}$	=	normal velocity of interface
ϕ	=	implicit representation of an interface
Ω	=	3D/2D spatial domain
v	=	vertex / grid point in Ω
p	=	triangular planer patch
e	=	edge of a patch connecting two vertices
$\vec{\boldsymbol{x}}$	=	3D/2D position vector or point
N	=	natural number
A	=	surface area
Ψ	=	volume
dx	=	length increment

Subscripts:

$inside$	=	interior domain of an interface
$outside$	=	exterior domain of an interface
t	=	time
Δt	=	discrete time increment
g	=	discrete rectangular grid domain
n	=	index of MC points
env	=	thin envelope around a interface
c	=	motor casing

[1] Masters student, Division of Applied Mathematics, Stellenbosch University
[2] Associate Professor, Division of Applied Mathematics, Stellenbosch University
[3] Consulting Engineer, Flamengro numerical simulation division of Armscor
[4] Design Engineer, Product Development, Werner.Rousseau@rheinmetall-denelmunition.com, AIAA Member

I. Introduction

Solid Rocket Motor (SRM) grain design has evolved considerably in recent times. Defining the burning area and port area profiles of the grain have traditionally been time consuming tasks. This often prevents the designer from evaluating all possible designs due to time constraints. The traditional methods used for modeling the grain burn back, such as defining the area and port profile geometrically, are not easily accomplished and are difficult to incorporate into CFD models. Additionally any grain that does not fall into a pre-defined grain configuration requires a new geometric model. CAD models have also been extensively used in the industry but with ever changing software techniques devised to automate, such grain regressions rapidly become obsolete.

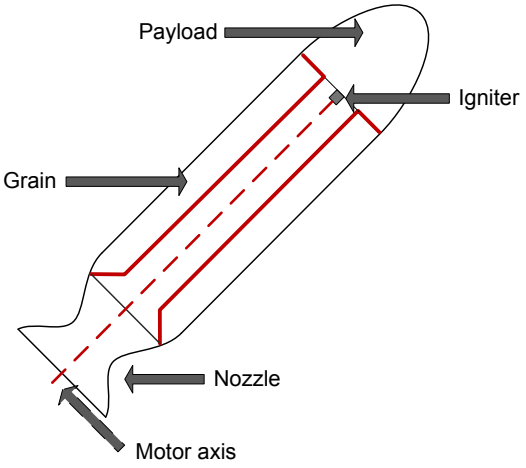


Figure 1: Layout of SRMs

Apart from the challenges involved in the above mentioned methods, there is a limitation in predefining the regressed burning surface since the burn rate is a function of pressure, and simultaneously, the pressure is dependent on the mass flow which again is a function of the burning surface. This creates a feedback loop in which the pressure and burning surface are codependent, which in turn can cause complex burn rate distributions across the grain surface. An implication being that the grain regression at predefined burn rates is, at best, an intelligent guess of the physical model. This codependence has given rise to a need for an interactive grain regression module that can handle both spatially and temporally varying burn rates. Recently, front tracking techniques such as the Volume of Fluids (VOF) method, Fast Marching Method (FMM) and especially the Level Set Method (LSM) have become the preferred techniques. The basic layout of a motor is given in Fig. 1. The grain design refers to the shape of the exposed surface of grain before ignition. The grain forms a hollow core, referred to as the combustion chamber. The axis through the center of the combustion chamber is referred to as the longitudinal axis or motor axis.

We employ the LSM to create a grain burn back module for burn back analysis and coupled with a 1D Internal Ballistics (IB) code to simulate the complete operational phase of motors with arbitrarily complex grain designs. The problem of handling the burnout phase of a motor is also addressed. This is accomplished without explicitly finding the burnt out sections of the motor but instead taking advantage of the ease with which intersections and unions of surfaces can be handled within the LSM representation of interfaces. Multiscale time integration for the LSM and IB solvers are employed as suggested by Steward et al[1]. This is possible since the regression rate of the grain and the velocity of the flow differ with large orders of magnitude and so the regression analysis need not be solved for every time step of the internal flow solver.

The formulation of the LSM will be presented along with the methods of initialization of the SDF and parameter calculation for a general closed convex interface in 3D. A multiple timescale coupling of LSM with an IB code is done and the method of handling the burnout phase of a motor is shown. Finally some results of simulations are compared to static test results for some novel designs.

II. Level Set Method

A. Formulation

The level set method introduced by Osher and Sethian[2], is used to advance an interface γ in its normal direction at a non-uniform velocity $\boldsymbol{V}$. It relies on representing γ as a zero level set of a higher dimensional function ϕ, defined on a domain Ω that spans γ. We can say that γ devides Ω into two sub domains, Ω_{inside} and $\Omega_{outside}$, i.e.

$$\begin{aligned} \gamma &= \{v \in \Omega \mid \phi(v) = 0\}, \\ \phi(v) &< 0 \quad \forall\, v \in \Omega_{inside}, \\ \phi(v) &> 0 \quad \forall\, v \in \Omega_{outside}, \end{aligned} \tag{1}$$

where v is a gridpoint in Ω.

The level set equation,

$$\phi_t + \boldsymbol{V}|\nabla\phi| = 0, \tag{2}$$

describes the evolution of ϕ in time and implicitly advances the interface. Fig. 2 shows the procedure for LSM interface evolution. The ϕ function is initialized as an SDF on a uniform rectangular grid Ω_g, that spans Ω. This results in a smooth continuously differentiable function and allows for evolutions of complex interfaces, including topological changes and sharp corners to be handled effectively. Equation 2 is solved using a discrete first order upwind approximation to the spatial derivative $\nabla\phi$ and an Euler time integration.

Since the velocity function is only defined on the interface itself, extension velocities need to be built across Ω. The most popular method for building extension velocities is based on solving the Eikonol equation with the FMM[3], this ensures that the ϕ function remains an SDF representation of the interface and prevents the isosurfaces of ϕ from grouping or separating as the interface is evolved. In our case the velocity function only varies in a single dimension and thus, the use of the FMM extension velocities is not required. Instead the velocity function is set equal to the burn rate at the given distance along the longitudinal axis of the motor. Equation 2 is then solved on ϕ_t using explicit first order upwind schemes that converge to the physically relevant solution of $\phi_{t+\Delta t}$, which is an implicit representation of $\gamma_{t+\Delta t}$.

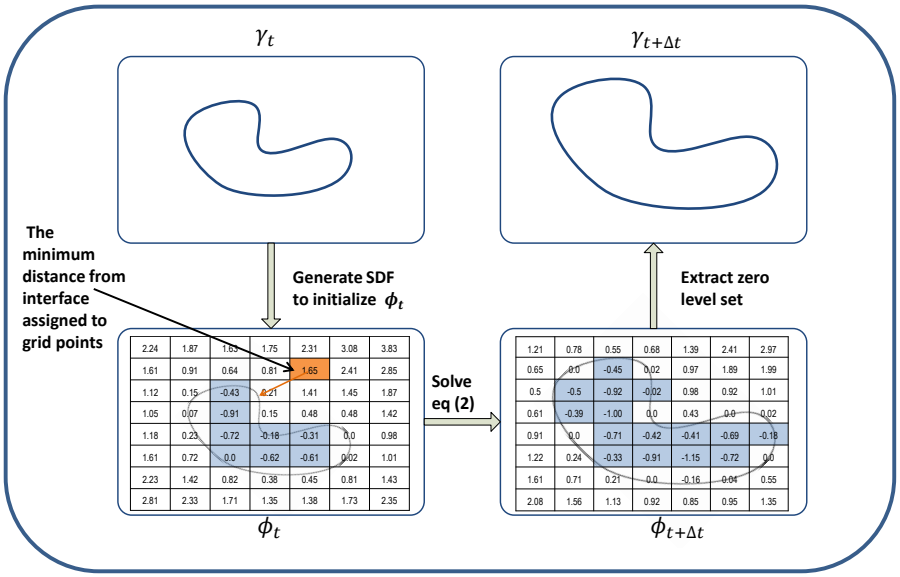


Figure 2: LSM procedure for a single time step evolution of the interface γ.

Once $\phi_{t+\Delta t}$ is found, the interface can be extracted and the necessary parameters calculated from the implicit function. In the case of closed convex interfaces in 3D, the LSM facilitates elegant calculations of interface parameters such as surface area and enclosed volume, making this method attractive for application in burn back analysis.

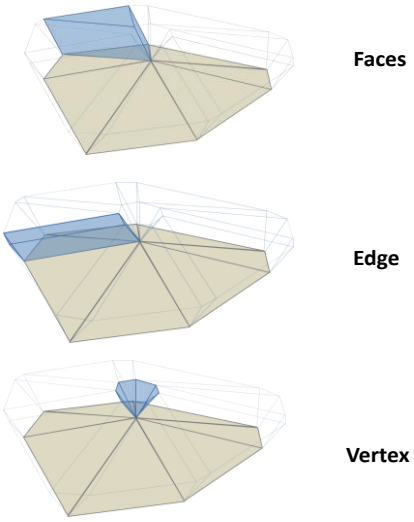


Figure 3: Polyhedra inclosing points possibly closest to a face edge or vertex of a triangular patch.

B. Signed distance function generation

Since most CAD software packages have the option of exporting STL files, the STL triangulation format is chosen as the manner in which the arbitrary grain geometries are represented by the designer. The generation of the SDF as an initial condition for the LSM from an STL file is accomplished by the following method based on the work of Mauch[4].

Note that an STL surface is the union of planer triangular patches, each defined by three vertices. The positive normal direction of each patch is also given. The minimum distance from an arbitrary point in 3D to the surface could be the distance to a vertex, edge or face of a triangular patch, as illustrated in Fig. 3. These are handled separately by doing scan conversions of polyhedra that contain the points

closest to each face, edge or vertex, on the discretized domain Ω. The grid points possibly closest to the face of a patch p will be enclosed by a rectangular prism that can be defined by the three edges and the normal of p. The points possibly closest to an edge e is enclosed by a wedge defined by the respective normals of the two patches that share e. The points possibly closest to a vertex v are enclosed by a polygonal pyramid defined by the normals of the $n \geq 3$ patches that share v. Once the polyhedral is defined, a scan conversion of Ω_g returns the grid points that are included to the distance calculation from each specific segment of the triangular patches. A loop over all the patches is performed and the grid points are assigned the minimum absolute distance that has been assigned during the distance calculations. The sign of the minimum absolute distance is kept for each specific grid point.

C. Parameter calculation by Monte-Carlo integration

The burning surface and chamber volume are the only two parameters that are significant for the coupling of a burn back module with a 1D internal ballistics code for SRM simulation. The parameters are calculated by means of Monte-Carlo (MC) volume integration, and is based on the Monte-Carlo method of Metropolis and Ulam[5]. The burning area is calculated by performing a thin envelope approximation. MC integration relies on scattering a large number N, of uniformly distributed points $\vec{x}_n, n = 1,2,\ldots,N$, across a domain Ω with a known volume Ψ_Ω. Calculating the volume of interest, Ψ, is done by finding the ratio of points inside the interface to the total number of points and multiplying with the domain volume,

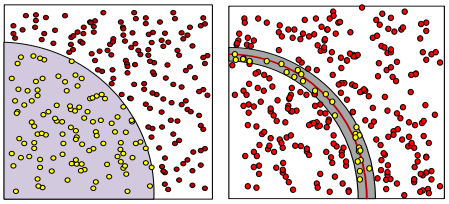


Figure 4: Monte Carlo integration for area and contour length.

$$\Psi = \frac{N_{inside}}{N} \times \Psi_\Omega. \tag{3}$$

These points are referred to as the MC points. For ease of illustration, the 2D analogue where area and length, rather than volume and area are calculated is illustrated in Fig. 4. The area A is calculated by finding the volume of a thin envelope Ψ_{env} of width dx as follows:,

$$A = \Psi_{env}/dx. \tag{4}$$

In the case of the implicit representations through SDFs, the MC points, $\vec{x}_n$ are simply trilinearly interpolated to find $\phi(\vec{x}_n)$ and then selected by their sign, negative being inside γ, i.e. $\vec{x}_n \in \Omega_{inside}$, and vice versa. For the contour calculation, the thin envelope area is found by selecting the points that interpolate to a value between the negative and positive half width values, $dx/2$ and $-dx/2$.

III. Internal Ballistics Coupling

A. Domain discretization

The LSM solver operates on a 3D discretization of a domain that spans the entire physical motor, whereas the IB solver operates on a 1D discretization of the motor axis. These 2 discrete grids are co-located such that each 1D element of the IB grid can be represented by a given number of 2D slices of the LSM grid, see Fig. 5. The number of slices of the LSM grid that fall within the range of a single element of the IB grid does not need to be uniform and can be chosen to suit the simulation of a particular grain design, depending on the complexity of the geometry of each segment. The 2D slices are recompiled to form a smaller subdomain on which the IB segment is defined and this is used to perform the parameter calculation of each segment.

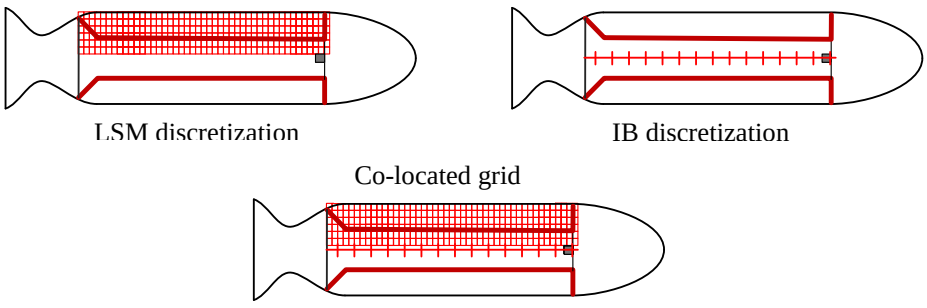


Figure 5: Domain discretization.

B. Multi scale time integration.

The coupling of a grain regression module and an IB solver couples two physical processes that occur with vastly varying velocities, the IB solves a flow field with velocities in the order of $O(10^3)$ m/s whereas the grain regression rate is typically in the order of $O(10^{-2})$ m/s. The two processes can therefore be solved at different time scales. The grain burn back is solved at a coarse time scale and the calculated parameters are interpolated to a finer timescale of the IB solver. The time scale used for the grain burn back is dependent on the rate of temporal change of the localized burn rates. As soon as the burn rates at any specific location along the longitudinal axis vary by a given percentage from the burn rates used during the previous grain burn back calculation, the grain burn back should be performed again. The problem of varying burn rates is encountered again since the burn rate progression is not specifically known. A retroactive procedure is employed where the burn back is performed to a temporary time step and linearly interpolated until the IB time scale reaches that of the burn back analysis. The burn back then proceeds at the new burn rates for another temporary time step. The temporary time step for the burn back analysis is chosen to satisfy the CFL condition on the LSM procedure. The assumption that the parameters can be linearly interpolated is weak, however sufficient, since the time step restrictions bound the introduced assumption error. A layout of the procedure is illustrated in Fig 6.

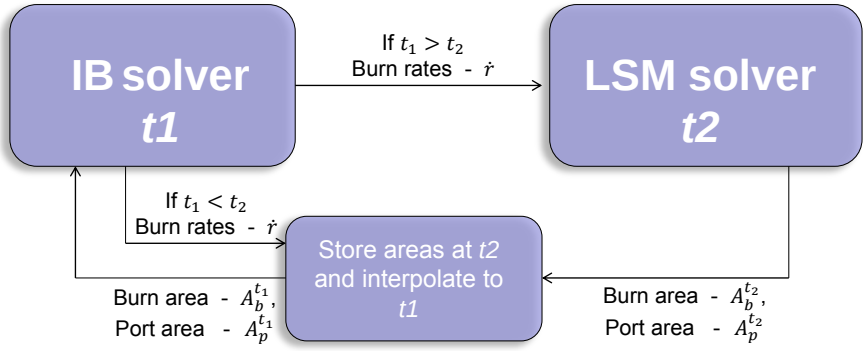


Figure 6: Multi time scale coupling of the IB and LSM solvers.

C. Burnout phase area and volume calculation

During the burnout phase of SRMs the grain burning surface reaches the motor casing and the grain becomes burnt out. The burn out phase stretches over the time from the first instance that the burning surface reaches a section of the casing until the entire motor is burnt out and there is no more propellant left within the motor. Let the motor casing be given by the interface γ_c. Now the grain burns back and eventually reaches the motor casing, starting to expose some segment thereof to the combustion cavity. The burnt out areas of the grain are disregarded in the burnt area calculation by performing a secondary interpolation of the MC points that contribute to the area, A_{env}, of the thin envelope approximation of the burning surface contour. The points are interpolated to the implicit representation of the casing, ϕ_c and deleted from the contributing points if found to be outside the casing, i.e. $\phi_c(\vec{x}_n) > 0$ as illustrated in Fig. 7.

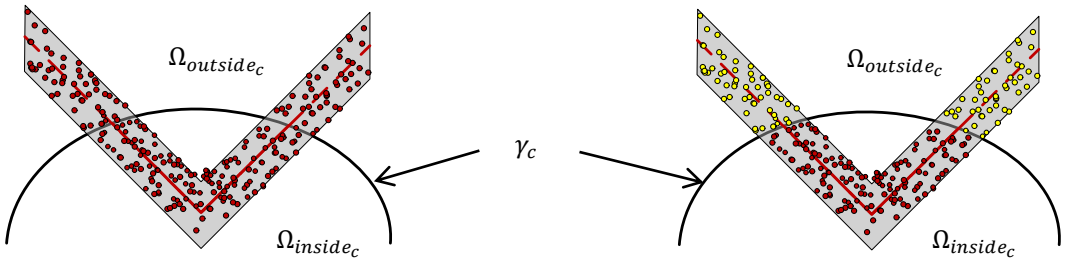


Figure 7: MC integration, disregarding burnt out grain segments.

The volume calculation during the burnout phase is found by finding the volume inside the intersected interfaces γ and γ_c. The intersection can be handled elegantly with the implicit representations of the interfaces,

$$\gamma \cap \gamma_c \rightarrow \max(\phi, \phi_c).$$

IV. Results

A. Merging spheres

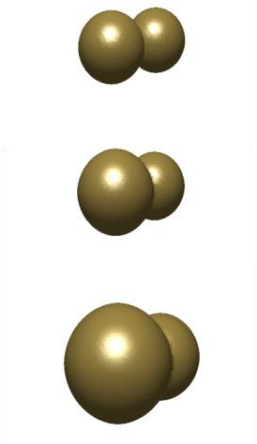


Figure 8: Two spheres growing and merging at non uniform speed.

First an analytical case to verify the calculation accuracy of the area and volume parameters was performed. Two spheres are placed a distance apart and then evolved at different speeds. The case at different time stages is illustrated in Fig. 8. The results are given for various grid resolutions and number of MC points used for the integration.

The case was run at a grid resolution of $40 \times 40 \times 60$ and the total residuals with respect to the analytical area are given for an increasing numbers of MC points used to perform the integration, as shown in Fig 9. From the residuals it can be seen that an increase in the number of points used improves the accuracy of the integration. The curve however does appear to resemble asymptotic behavior, which is possibly due to the discrete representation of a continuous geometry and the fact that the surface is calculated by a thin envelope approximation. Next the case was run for increased grid resolutions with a fixed number of MC points, as shown in Fig. 10.

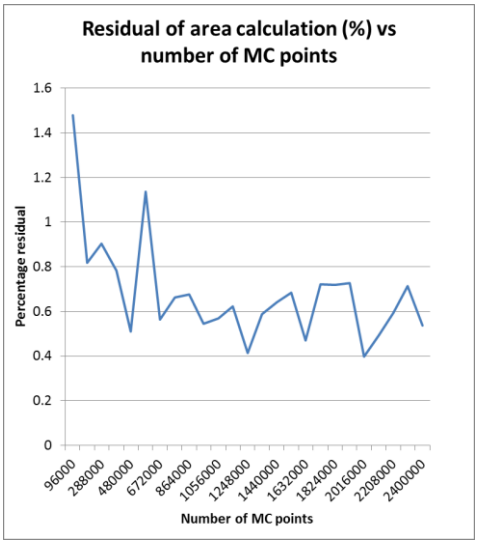


Figure 9: Residuals of the Monte-Carlo area integration for varying numbers of MC points.

From the results shown it is seen that the residuals decrease as the grid resolution is increased. The gain in accuracy quickly becomes small w.r.t the increased computational cost. The residuals for all the cases, apart from the grid resolution of $20 \times 20 \times 30$, included in the results are within the region of 1% and below, which is an acceptable error for design purposes since the experimental data from SRM static tests are typically not of higher accuracy and variation in manufactured motors often reach the 1% range. The same case was also setup with the interface evolved by means of the LSM and no significant changes in the parameter residuals were observed when compared to the analytically evolved interface. This gave a good initial confidence in the accuracy of the LSM for interface evolution, since the case contains both sharp corners and changes in topology. Both these properties typically prove problematic for numerical interface evolution techniques.

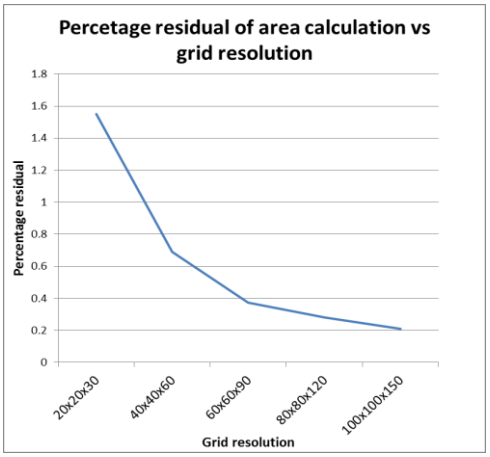


Figure 10: Total residuals resolutions of MC integrated area's for varying grid.

B. The anchor grain

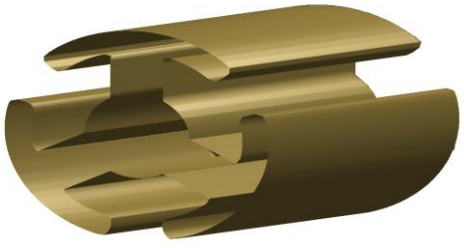


Figure 11: An anchor grain configuration.

In his paper, Mathew Umbel[6] gave an exact analytical calculation for the burn surface of a generalized anchor grain configuration, illustrated in Fig. 11, as a function of distance burned. A CAD model of a specific case was created and the SDF generation technique described in section 2B was used to initialize the LSM and perform a burn back at constant uniform velocities. The results from the MC integration with burnout calculation are compared to Umbel's analytical results. The effect of the width of the thin envelope approximation can be observed in the form of smoothing during the burn out phase as shown in Fig. 12.

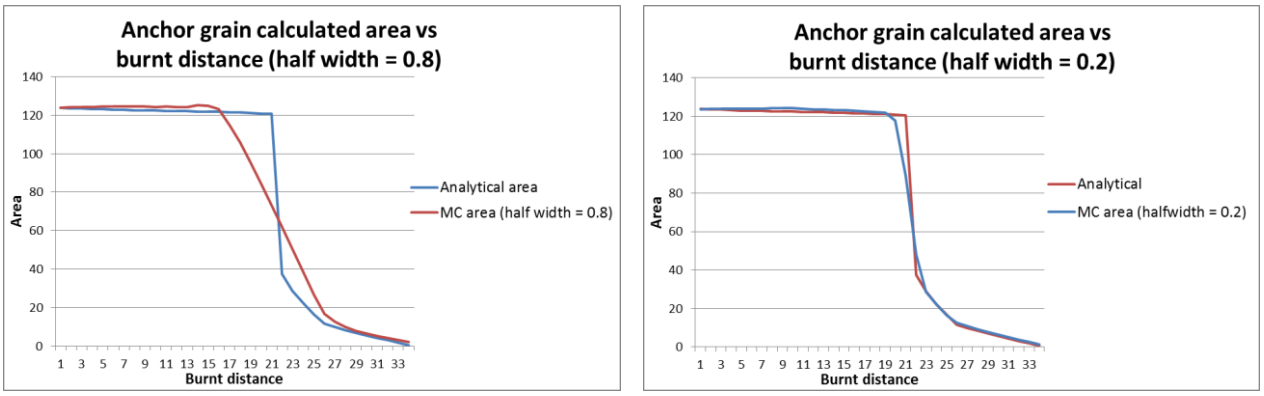


Figure 12: MC integrated area for thin envelope half widths of 0.8 and 0.2 respectively.

The results for a thin envelope of half width $dx/2 = 0.8$ units show a significant deviation from the analytical results during the burnout phase of the geometry. The thin envelope approximation of half width $dx/2 = 0.2$ units, follows the analytical solution more closely. Care should however be taken, since a half width, which is significantly smaller than the grid spacing, could lead to unstable calculations of the burn surface area and make the area integration dependent on the grid orientation with respect to the interface location. The results have so far pointed to a half width of at least more than half the minimum grid spacing in any particular dimension. The grid spacing of the above given results was set to a uniform 0.25 units in all 3 dimension. Fig. 13 shows the results for a half width of $dx/2 = 0.05$ units. Note the oscillations in the calculated area as the interface moves through the grid. This is due to the small width of the envelope.

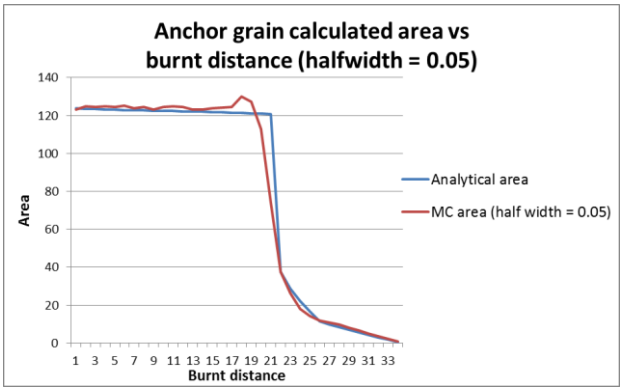


Figure 13: MC integrated area for thin envelope half width of 0.05.

C. Full motor simulation

A full motor simulation of a novel design was conducted, coupling the LSM to an IB code via the multi timescale procedure of section IIB, and the results were compared to actual test bed data. First a short discussion of the IB code is appropriate.

The internal ballistics solver as described by Lamberty[7] developed for the Solid Performance Program (SPP) is the chosen basis for the coupled solver IB code. The code converges on the Mach number of the internal flow rather than the internal pressure, which is perhaps more conventional.

The motor grain design at various stages of the operational phase is illustrated in Fig. 14. Since the design is still under the design phase and the test data confidential, the results for the tests and simulations are given in dimensionless units. The results of the simulation are given in Fig.15. The motor simulation was done without any nozzle erosion models being implemented. This might explain the deviation from the experimental results, which seems to grow with time.

Figure 14: Novel Grain design at various stages of the motor operational phase.

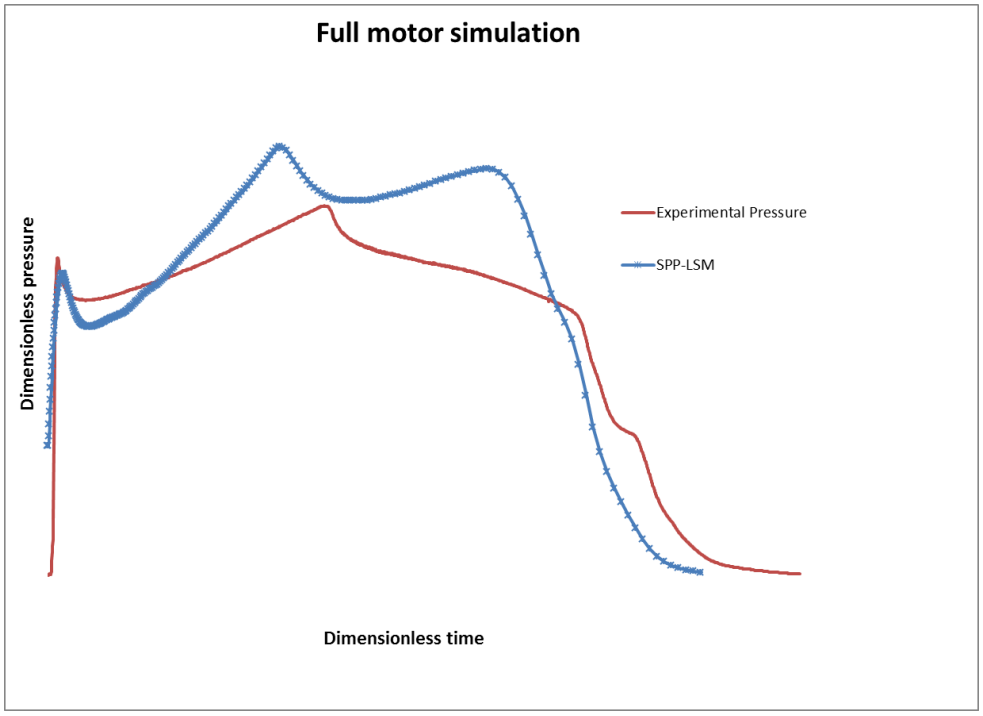


Figure 15: A comparison of static test bed experimental data and the full motor simulation utilizing the SPP and coupled LSM.

V. Conclusions and future work

The LSM burn back simulation provides an accurate numerical method of calculating the necessary geometrical properties to perform an SRM internal ballistics simulation. With techniques such as multi time scale approaches and efficient geometric calculation through the MC integration techniques, it is possible to perform coupled IB and grain burn back simulation of SRMs in a practical time frame. The advantage for design engineers is that a new design can be generated and simulated without any laborious geometric modeling as a function of burnt distance, which could dramatically reduce the time from design sheet to test bed. Certain geometrical effects caused by the non-uniform rate at which the burning surface might regress are also better captured by the LSM.

The next step in the development of LSM grain burn back methods is to increase the numerical efficiency of the algorithms. The methods described above are well suited for parallelization to enhance computational efficiency. Most grain designs are also symmetric in nature and this could be exploited to further reduce the computational cost. A further area of possible improvement is the MC integration techniques. Stratifying the domain as well as optimizing the allocation of MC point density might improve the speed of the algorithm.

Since the LSM has the ability to handle a spatially varying regression rate, the LSM might also provide some insight into the more subtle eccentricities of erosive burning models and how certain grain designs react, looking at possible areas where shockwaves might form and for instance, possibly pose a risk of motor failure. The future plans are to couple the LSM to a full 3D internal ballistics code in an OpenFOAM® environment. An area that requires some development in order to achieve this goal, is the fast and effective 3D grid generation for a 3D IB solver. The implicit representation of the burning surface interface could possibly be effectively exploited for the purpose of fast robust grid generation.

References

[1] Stewart, D. S., Tang, K.-C., Yoo, S., Brewster, M. Q., and Kuznetsov, I. R., "Multi-scale modeling of solid rocket motors: Time integration methods from computational aerodynamics applied to stable quasi-steady motor burning," 2004.

[2] Osher, S., and Sethian, J. A., "Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations," *Journal of computational physics*, vol. 79, 1988, pp. 12–49.

[3] Sethian, J. A., "Advancing interfaces: level set and fast marching methods," 1999.

[4] Mauch, S., "A fast algorithm for computing the closest point and distance transform," *Go online to http://www. acm. caltech. edu/seanm/software/cpt/cpt. pdf*, 2000.

[5] N. Metropolis, and S. Ulam, "The Monte Carlo method," *Journal the American Statistical Assosiation*, vol. 44, Sep. 1949, pp. 335–342.

[6] MAtthew R. Umbel, "An Exact Geometric Analysis of the Generalized Anchor Grian Configuration," *AIAA*, Hartford, CT: 2008.

[7] J.T. Lamberty, *A report on the grain design and internal ballistic module of the improved solids performance program*, California: United Technologies Chemical Systems Devision, 1981.

49th AIAA/ASME/SAE/ASEE Joint PropulsionConference
July 14 - 17, 2013, San Jose, CA

AIAA 2013-3789

Rapid Solid Rocket Motor Design

C.W. Rousseau[1]& S.F. Steyn[2]
Rheinmetall Denel Munition, Firgrove, Western Cape, South Africa, 7130

W. Sullwald[3], E.R. De Kock[4], G.J.F. Smit[5] J.H. Knoetze[6]
University of Stellenbosch, Private bag X1, Matieland, Stellenbosch, Western Cape, South Africa, 7602

Internal ballistic design of solid rocket motors (SRMs) is a well-established field. Most grain designs have been well characterized throughout the industry. However, generating and evaluating different grain design options can be quite tedious and time-consuming. Thus, it was endeavored to create a preliminary design tool which can be used in a workshop with a client to promptly establish the most likely and suitable grain and performance design for the particular missile application, starting with a system definition. This tool was developed in the MATLAB® environment.

This tool also serves as an input for the more detailed design. As part of the larger program it was endeavored to create an internal ballistic tool that allows for more detailed analyses. It was endeavored to use the rapidly expanding open source tools available to develop a fully coupled CFD internal ballistic tool. The grain regression and CFD modules have been developed to date.

Nomenclature

a	=	burning rate coefficient
A_t	=	nozzle throat area [mm^2]
C_D	=	drag coefficient
C^*	=	characteristic velocity of the propellant combustion gasses [m/s]
d	=	diameter [mm]
eff	=	nozzle efficiency
F	=	thrust [kN]
I_{SP}	=	specific impulse [s^{-1}]
l_c	=	length of cylinder
$\dot{m}$	=	mass flow [kg/s]
n	=	burning rate pressure exponent
S	=	propellant grain burning surface area [mm^2]
p	=	pressure [MPa]
V_c	=	propellant burning rate [mm/s]
S	=	burning surface area
ρ	=	propellant density [kg/m^3]
γ	=	3D surface interface
ϕ	=	implicit burning surface representation
Subscripts		
base	=	base drag
c	=	cylinder parameters
friction	=	skin friction drag
fin	=	fin parameters

[1] Design Engineer, Product Development, Werner.Rousseau@rheinmetall-denelmunition.com, AIAA Member
[2] Senior Project Manager RDM
[3] Master student, Department of Applied Mathematics, University of Stellenbosch
[4] Master student, Department of Applied Mathematics, University of Stellenbosch
[5] Associate Professor, Department of Applied Mathematics, University of Stellenbosch
[6] Professor, Department of Process Engineering, University of Stellenbosch

on = motor operating
off = motor not operating
wave = wave drag
Φ = implicit burning surface representation
1 = propellant 1
2 = propellant 2

I. Introduction

The initial design of a new solid rocket motor (SRM) can often be time-consuming. For a given system, several different designs can be implemented. It is often required that a representative design be done of all the possible propulsion concepts that could be employed. For example, a boost-sustain motor could have a dual propellant-layer grain, two propellant blocks, finocyl or combinations of these designs. For each design, a geometric model is required. Depending on the tools that are available to the designer, this step can be both tedious and time-consuming.

The aim is to produce a design tool that will allow for rapid initial designs of SRM grains for comparative evaluation. This evaluation should be of such a nature that this can be done in a relatively small amount of time, in some cases within the time of one workshop with a prospective client. The designs from this analysis then serve as input to the next phase of detailed design.

This tool is based on combination of the conservation laws, physics-based analyses, and, when available, empirical and experimental information from similar systems in a similar manner, as proposed by Fleeman[1]. Several steps must be followed to get to the final design. The first step is to define the required motor performance, based on the user's requirements. Once the system envelope is defined, a basic aerodynamic model is created for the system; the SRM envelope is then defined and the potential impulse that can be delivered is calculated. From the potential impulse, thrust curves are optimized to meet the primary mission requirements by performing trajectory simulations. The final step is to use these thrust curves to generate pressure profiles and, accordingly, the corresponding grain designs.

II. System Definition

The first step is to define the hypothetical system's airframe envelope. This is done by using the methodology described in *Tactical Missile Design* by E.U. Fleeman[1]. The analysis is limited to the system mass and estimate of the drag coefficients. Other subsystems such as the warhead, seeker, and guidance are not evaluated. The primary goal is to speedily define an SRM that is capable of delivering the required performance.

The set of equations as presented by Fleeman[1] produces realistic drag coefficients for the subsonic and supersonic region. The coefficients calculated for the transonic regions are much higher than would be expected. Thus instead of the blunt-nose wave drag term used[1], an empirical model is used for the nose wave drag based on data from Waliskog and Hart[2] is also a selectable option. However, it should be noted that since most of the flight time will be predominantly either supersonic or subsonic, the approximation presented by Fleeman is sufficient as a first order approximation for most systems.

The total airframe drag is then the sum of the skin friction and base drag calculated by Fleeman's method and the empirically determined wave drag. For power on (P_{On})[1] the total drag coefficient is:

$$C_{D_{total\_P_{on}}} = C_{D_{friction}} + C_{D_{base\_P_{on}}} + C_{D_{wave}}, \quad \textbf{(1)}$$

and for power off (P_{Off})[1] the total drag coefficient is:

$$C_{D_{total\_P_{off}}} = C_{D_{friction}} + C_{D_{base\_P_{off}}} + C_{D_{wave}}. \quad \textbf{(2)}$$

For the empirical formulas of each of the terms in Eq. (1) and Eq. (2), the reader is referred to *Tactical Missile Design*[1], pg. 23. These equations are used to calculate the drag coefficients to be used in the 3 degree of freedom (DOF) trajectory simulation.

III. Internal Ballistic Equations

From the 3DOF trajectory simulations, an idealized thrust profile is generated taking into account the volume available for the SRM, and the potential volumetric loading. The known required thrust profile and the following equations, found in any good reference on solid rocket motor design and internal ballistics such as Sutton[3] and

Davenas[4], are then utilized by the rapid design code to conduct and evaluate various grain design options that would best meet the idealized thrust profile. The following equations can be found in any good reference on solid rocket motors internal ballistics and design such as Sutton[3] and Davenas[4]. Once the required thrust profile is known, the motor grain design can commence. The thrust, F, can be related to mass flow rate through the nozzle, $\dot{m}$, in the following manner

$$F = \dot{m} \cdot I_{sp} \cdot eff. \tag{3}$$

The specific impulse (*Isp*) of the propellant composition can be found using a thermodynamics package. The nozzle efficiency, *eff*, can be inferred from experience. Therefore, the mass flow rate is the only remaining unknown in Eq. (3). The chamber pressure can be related to the mass flow through the nozzle and the mass flow from the grain when a steady operating point is reached:

$$\dot{m} = \frac{PA_t}{C^*} = \rho \cdot S \cdot V_c. \tag{4}$$

The pressure within the motor chamber can be calculated as follows:

$$P = \frac{\dot{m}C^*}{A_t}. \tag{5}$$

The burn rate (V_c) is a function of pressure. If all other parameters are known or can be estimated with relative certainty, the burning surface area, S, required to generate the required mass flow can be calculated by rearranging Eq. (4), yielding

$$S = \frac{PA_t}{\rho V_c C^*}. \tag{6}$$

This allows the generation of a required burning surface area profile. In the case of a finocyl design the cylindrical section's burning surface area as a function of distance burnt is known given that a certain cylinder length is chosen, and thus it is possible to calculate the required burning surface area for the fins:

$$S_{fin} = S - \pi d_c l_c. \tag{7}$$

Several fin geometries have been geometrically described within the computer code. The required fin burning perimeter is compared to the analytically generated burning perimeter, and the fin parameters are adjusted to obtain the best fit.

In the case of a radially slotted motor design, the surface area is set once the number of slots and slot sizes are chosen, and the required burn rate needs to be calculated. This can be done by scaling a known propellant's burn rate i.e. keeping the burn rate exponent, n, value constant and adjusting the burn rate coefficient, a. The following equation can be solved iteratively to find the best burn rate match to the required thrust profile:

$$V_c = \frac{P A_t}{\rho S C^*}. \tag{8}$$

It should be noted that this can be done for both two layers of propellants and two different propellants along the length of the motor. For two propellants the following equation must be solved numerically:

$$P = \frac{(\rho S_1 V_{C1} + \rho S_2 V_{C2}) C^*}{A_t}. \tag{9}$$

A. Design Decision Path

The basic decision path is shown in Figure 1. Once the thrust curve has been determined, the required mass flow can be obtained from Eq. (3) and the required pressure can be calculated for a constant throat area using Eq. (5).

If the desired thrust profile is boost and sustain, and a lower volumetrically efficient design is sufficient, the finocyl design can be employed. Since the cylinder section's geometry is known for a chosen cylinder length and port diameter, it is possible to solve for the fin area using Eq. (7).

If a boost-sustain profile and a higher performance motor with greater volumetric loading is required, a radial slot grain could be considered using two propellants with different burn rates. Additionally, two configurations can be considered; namely concentric annular layers of propellant or two propellants in series (dual block). Since the geometry is essentially set with only the number of radial slots and port diameter that can be varied, the only unknown is the required burn rate. For the dual block design, however, the length of the propellant sections can be varied. Once the preferable design has been identified, a more detailed design and optimization process commences.

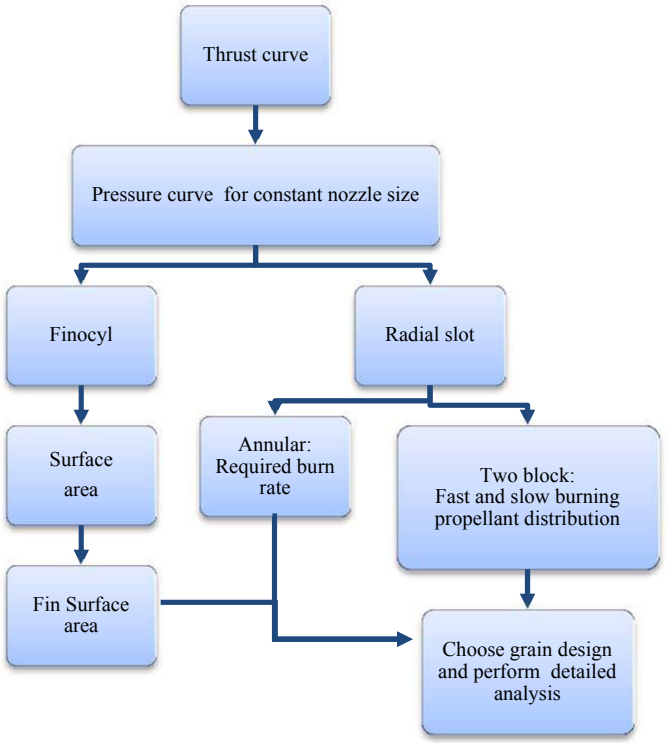


Figure 1: Basic design decision path.

IV. Design of Dual Role Missile

A. System Aerodynamics

A 200 mm caliber missile will be used as an example, to illustrate the design methodology. This system will be required to fulfill both a ground-to-air and air-to-air capability. The system is not mass constrained but has an available length of 4 m. It is assumed that a canard-wing configuration will be required for aero control. Figure 2 shows the aero output and the system mass is calculated to be 226 kg. Several templates for classes of tactical missile aero structures are available in to simplify and speed up user input. The code operator is then only required to modify a few of the parameters to match the user's requirement. The drag and the missile mass are exported to the 3DOF trajectory code.

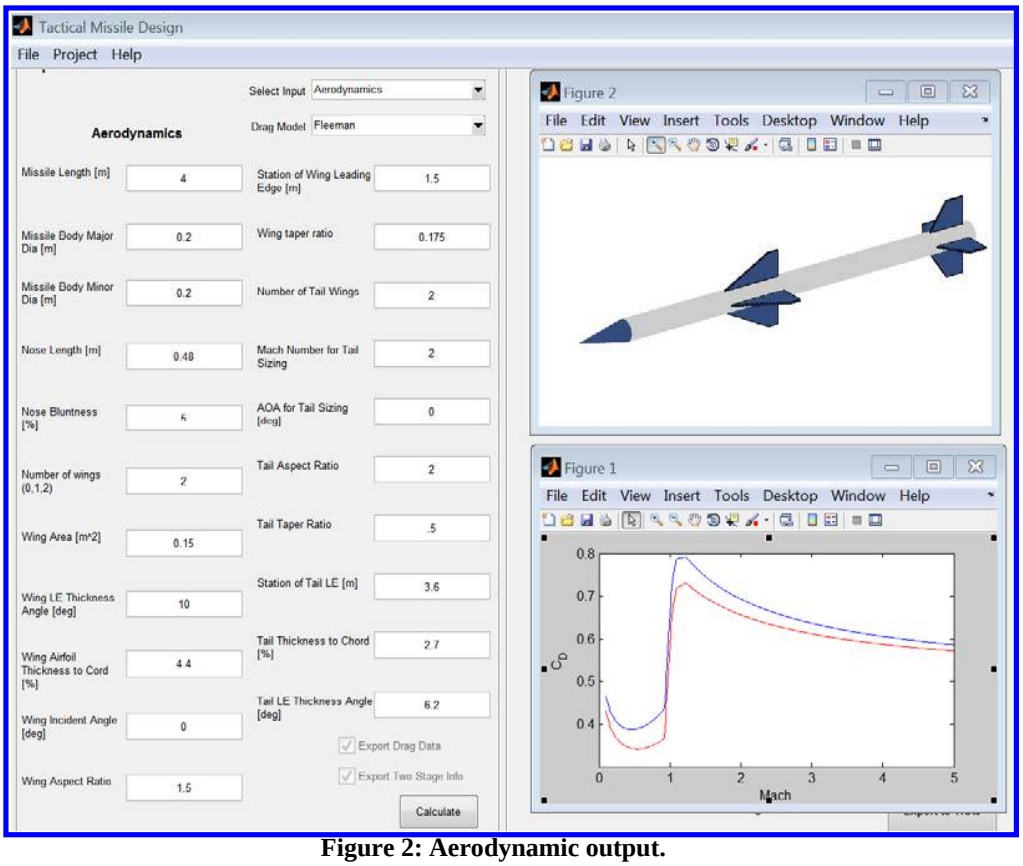


Figure 2: Aerodynamic output.

B. Obtain Thrust Profile

The primary flight profiles are presented in Table 1. Since the maximum Mach number has been limited to 3.5, the thrust profile must be of such a nature that the maximum Mach number will not be exceeded. The air-to-air profile is the most likely to result in the maximum Mach number being exceeded. This is due to the higher launch velocity and the lower drag at higher altitudes.

Table 1: Flight profiles.

Primary flight profile	Launch altitude [km]	Launch Mach number	Intercept altitude [km]	Horizontal range [km]	Intercept Mach	Maximum Mach
Air-to-air	10	0.8	10	40	>1.2	3.5
Ground-to-air	0	0	10	20	>1.2	3.5

The code allows for the boost sustain profile to be specified by two methods, the first one being to specify the maximum Mach number as the goal after the boost phase, and then setting the sustain thrust equal to the drag. The code solves iteratively for the boost phase, the first guess being the impulse required to accelerate the missile to the required velocity; the thrust level is set by the minimum acceleration level required. The remaining impulse is allocated to the sustain section and the burn time calculated from the thrust level required to maintain the maximum Mach number. The second method is to specify a Mach goal below the maximum Mach number after the boost phase and then solve iteratively for a sustain thrust that will accelerate the motor to the maximum Mach value, assuming there is enough propellant available to do so. Once this has been done, the 3DOF trajectory code is allowed to complete the flight profiles to ensure that the requirement is met.

These two thrust profiles (see Figure 3) can then be traded off against each other. If the goal is to boost the motor to its maximum velocity, then more of the energy is expended during this particular phase. This requires that the boost phase operates for a longer period of time, as well as, in the case of a finocyl, becoming less volumetrically efficient, since the fin section may have to be extended. A second possible constraint for this design is the propellant burn rate and the available web. Thus if the web of the fin section is insufficient to provide the required burn time at

a reasonable propellant burning rate, this design may not be executable. Additionally, the boost to sustain ratio may not be achievable.

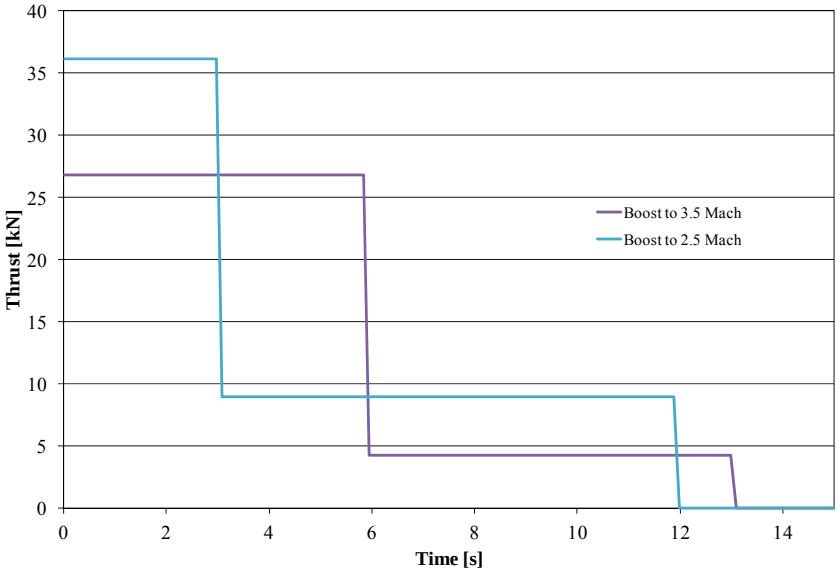


Figure 3: Thrust profiles.

If the maximum velocity is only to be achieved after the sustain phase, the boost duration as well as the propellant mass consumed during this phase can be reduced. This allows for a shorted fin section increasing the overall volumetric efficiency.

When considering radial slot grain designs, the main limitation is the initial thrust level required to safely separate from the launch platform. Since the initial grain surface area is relatively low, it is difficult to generate the required mass flow without having to resort to an excessively high burn rate or operating the motor at too high a pressure as the surface area increases. This can also be compensated for, to some extent, by geometric modifications, but it may reduce the volumetric efficiency of the grain, which is the primary advantage of this kind of grain design.

C. Finocyl Design

The thrust profile required to boost the system to 2.5 Mach is selected, as the thrust profile generated to boost the system to 3.5 Mach is not practically achievable. This thrust profile is now passed to the next module in the code that calculates the required operating pressure and area profile, to generate the thrust as described in Section III. The booster and sustained outer diameters are specified independently to allow for the different rubber layups to be accounted for. A typical propellant's thermodynamic and ballistic data can be imported. The required burn time is achieved by scaling Vielle's burn rate constant, a, for a particular nozzle throat size. The nozzle throat size may have to be adjusted iteratively if the predicted pressure to generate the thrust level exceeds the desired maximum operating pressure. If the required burn rate does not fall within the applicable range of the propellant type, the best possible solution is returned.

Figure 4 gives the required burning surface area for the motor. It is important to note that the booster surface area profile is for the required burn surface area augmentation to achieve a neutral profile for both boost and sustain. An approach to achieve this profile could be to include a tapered transition between the cylinder and the fin section. However, it would not be possible to achieve a truly neutral profile in practice.

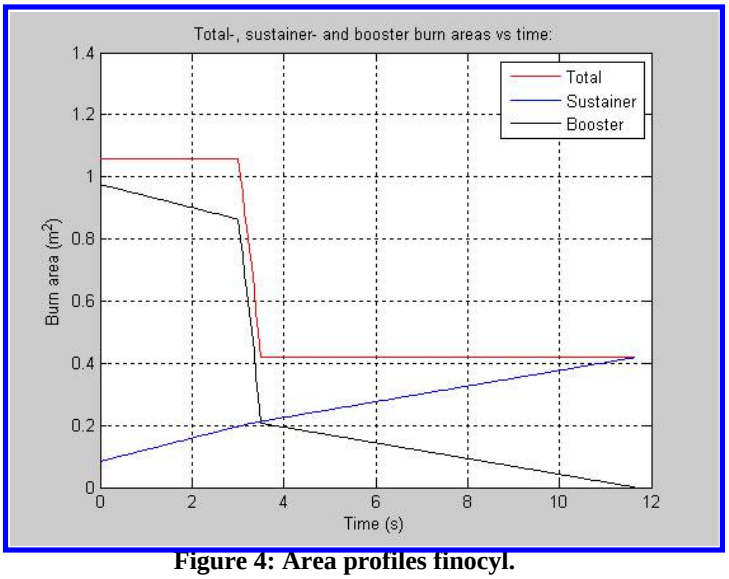


Figure 4: Area profiles finocyl.

Figure 5 shows the burning perimeter required from the booster. The booster section's dimensions are sent to the fin design module. The fin parameters can be modified to best suit the required burning perimeter. Figure 6 shows the fin design and the burning perimeter. For this particular example, the fin design was chosen to be initially progressive to compensate for anticipated erosive burning effects at ignition which elevates the initial burning surface area profile to be regressive, as shown in Figure 4. Once this has been done, it is possible to create a CAD model of the grain. To avoid having to parameterize the entire model to perform the grain burn-back surface analysis, the grain can be sent to a module that uses level set methods (LSM) to obtain the area profile. Several iterations can be performed to fully evaluate the effects of the transition between the cylinder and fin sections. This will be discussed in Section V.A.

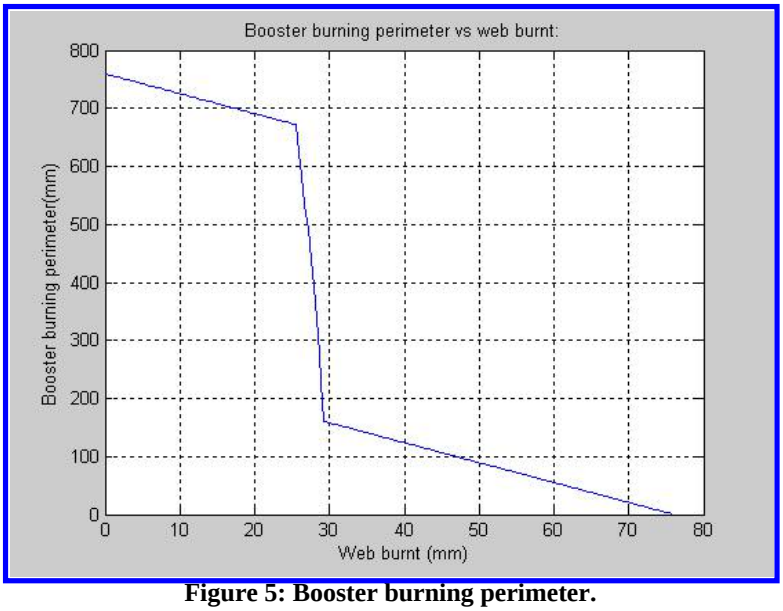


Figure 5: Booster burning perimeter.

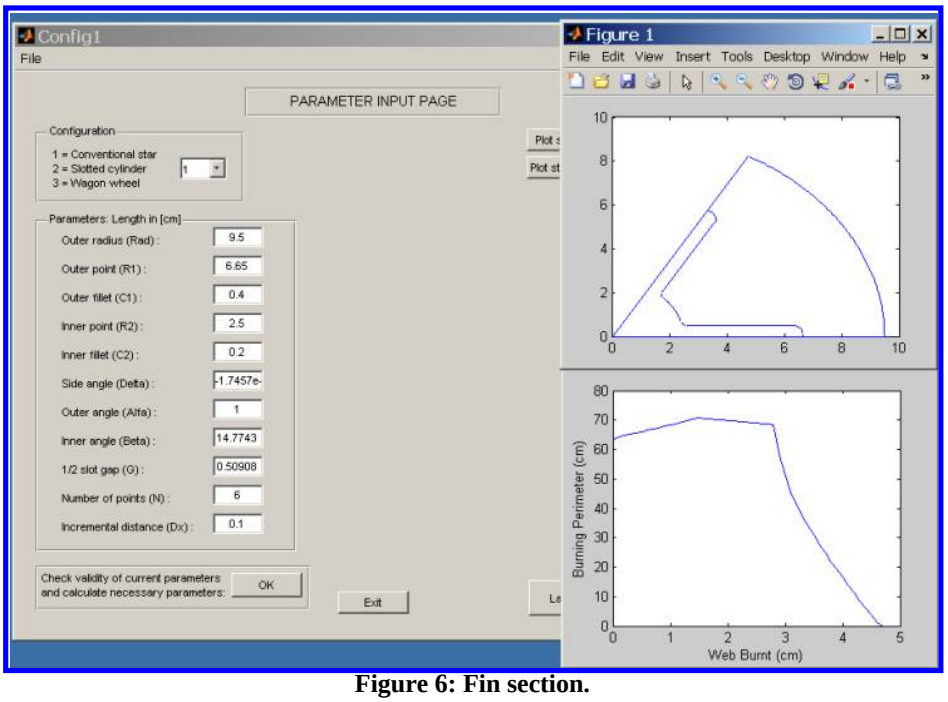


Figure 6: Fin section.

D. Slotted design

Boost-sustain grain designs can be achieved with high volumetrically loaded, radially slotted grain designs utilizing two propellant formulations: a fast-burning and a slow-burning formulation. The two propellant formulations can also be configured in two geometric layouts. The first concept is that of two concentric layers, with the fast-burning propellant on the inside and the slow-burning propellant on the outside. Figure 7 shows the potential profile for such a design given all the other motor design and ballistic parameters. This design is not suitable, as it does not allow enough mass flow during the boost phase, whilst the much larger burning surface area over the full length of the second propellant leads to much higher thrust levels than desired for the given motor design and ballistic parameters, as mentioned above.

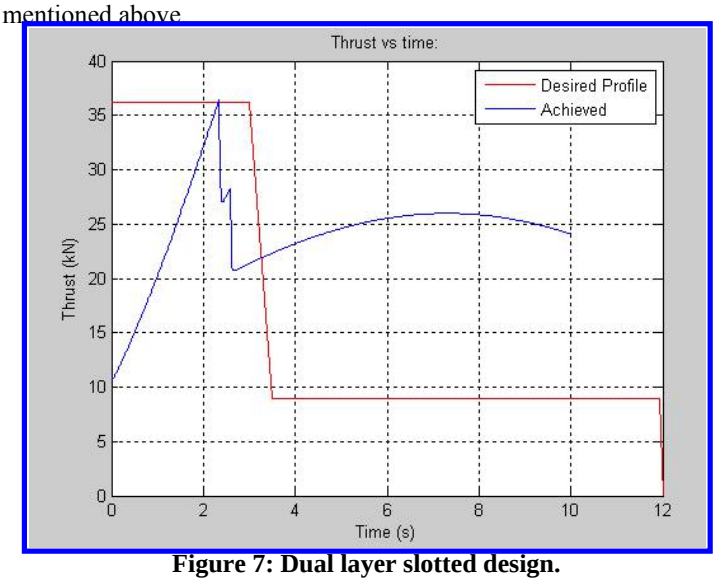


Figure 7: Dual layer slotted design.

The second configuration represents that of two propellant formulations cast in tandem along the length of the grain, with the fast burning propellant at the rear of the motor. This design makes the lower sustain thrust levels easier to obtain, since the total grain length is reduced when the boost phase is burnt out. There is a penalty in such a design due to the additional insulation requirement in the boost area. However, it is shown in Figure 8 that such a

design, even without any geometric modification in the fast burning propellant area, the requirement is nearly met. If there is no initial launch thrust requirement, such a design could be considered. This design however, depending on the client, may not be considered due to increased cost and increased manufacturing complexity.

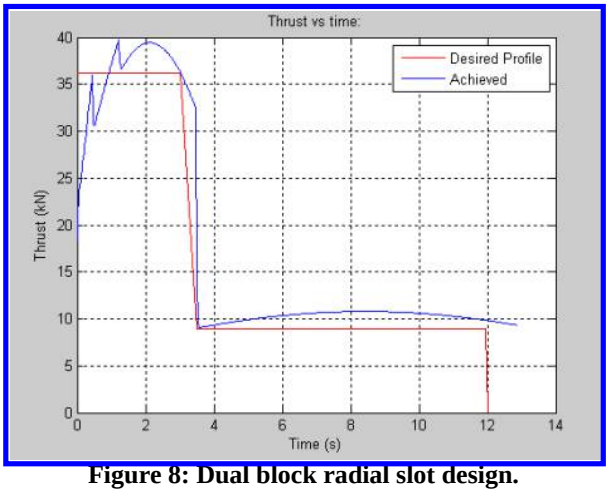


Figure 8: Dual block radial slot design.

V. Advanced Modules

Once the most suitable boost-sustain grain design has been achieved using the algorithm as presented above, the next step is to do a detail design of the most suitable grain design which takes factors such as casing geometry, rubber layup, etc., into account. Several geometries are generated for this design, taking the relevant constraints into account. The geometries are then exported to the LSM module where the area profiles are generated. Once the detail design of desired grain geometry has been completed, full CFD simulations can commence for final design validation.

A. Level Set Method

The level set methods of Osher and Sethian[5] is employed to create a grain burn back module for the burn-back analysis of complex grains. It relies on implicitly representing an interface $\boldsymbol{\gamma}$, as a Signed Distance Function (SDF) $\boldsymbol{\phi}$.The initial $\boldsymbol{\phi}$ function is generated from an STL file of the grain design by the method of Mauch[6], The interface/burning surface is advanced in its natural direction at a speed V by solving the Level Set equation,

$$\phi_t + V|\nabla\phi| = 0. \quad \textbf{(10)}$$

Figure 9 illustrates a single time step evolution of a general interface $\boldsymbol{\gamma}$, using the LSM.

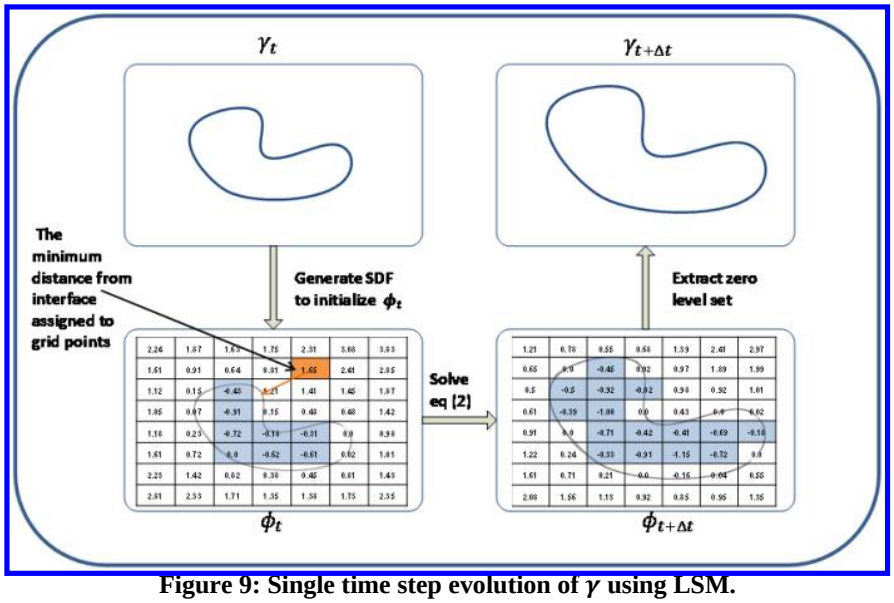
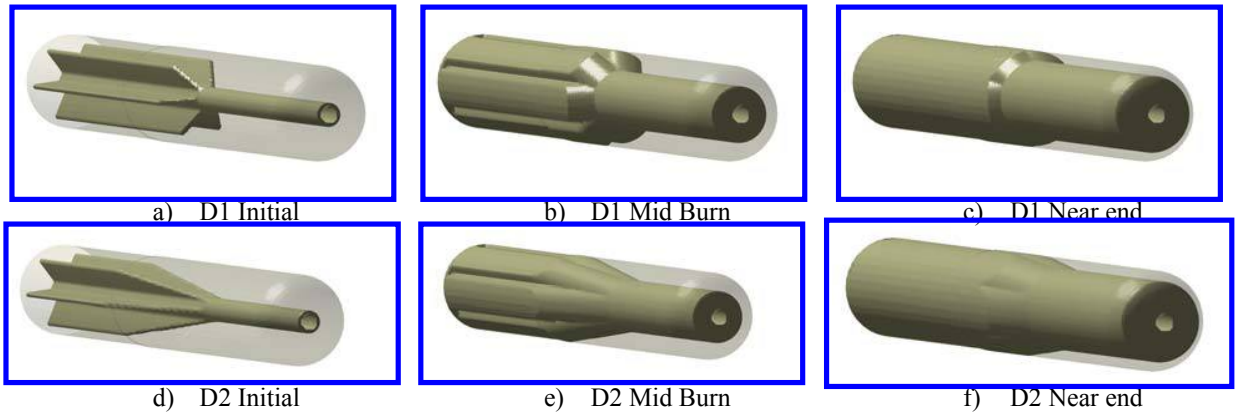


Figure 9: Single time step evolution of γ using LSM.

Instead of explicitly finding the interface at the next time step, the burn area parameter $\boldsymbol{S}$ is calculated using Monte-Carlo type integration which relies on the interpolation of a number of random points to $\boldsymbol{\phi}$. The detail of the LSM burnback module is found in Sullwald *et al*[7]**.**

Figure 10 shows the two initial designs generated. The first design (D1) has a very steep transition between the fin and cylindrical sections. The second design has a less steep transition between the fin and the cylindrical section to create a more neutral burn profile. Being able to visualize the grain in this way allows for design modifications to be made with more certainty.



a) D1 Initial b) D1 Mid Burn c) D1 Near end

d) D2 Initial e) D2 Mid Burn f) D2 Near end

Figure 10: Burn back using the LSM method.

Figure 11 shows the area profiles for the grains as predicted using the LSM method. Due to the progressive nature of the star section it is difficult to reach the desired initial surface area. However, this grain will experience erosive burning and thus the initial thrust level will be lifted. D1 is the closest to the desired performance and is therefore optimized further. D2 with tapered section has a longer transition between boost and sustain as well a regressive sustain profile. The integral of the surface area is compared to the design goal to ensure that the required impulse will be delivered. The grain can then be lengthened to ensure surface area integrals match.

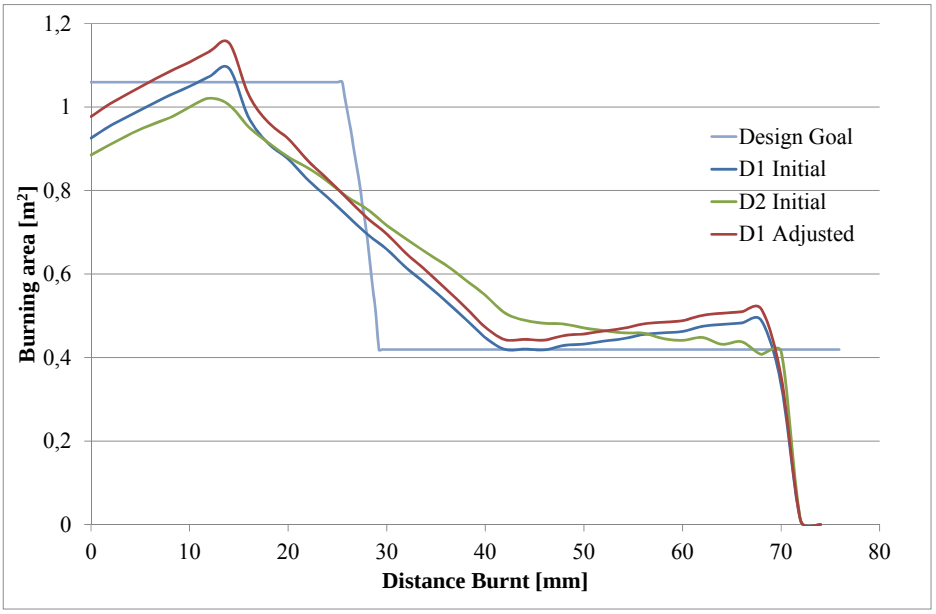


Figure 11: Predicted burning area LSM.

B. Advanced CFD Simulation

The final step in the design is to perform internal ballistic simulations for the grain[8]. A CFD solver is in development for this purpose using OpenFOAM. The first step has been to develop a flow solver that can capture complex phenomena associated with SRM internal flows, by utilizing a high order reconstruction method to approximate the average cell values. This reconstruction method presented by Jiangand and Osher [9] calculates the flux on the cell boundaries via a polynomial. The polynomial consists of a combination of adaptive weights, which are calculated by utilizing stencils obtained from the solution. Stencils containing discontinuities in the solution carry smaller weights to avoid spurious oscillations and a larger weight in smooth regions of the solution. The weights adapt to the solution to achieve a higher order of accuracy. This has been successfully done in OpenFOAM by de Kock *et al*[10]. Utilizing the built-in functions of OpenFOAM, the time and mesh configurations are controlled. The solution is approximated via the weighted essentially non-oscillatory method (WENO) which is the high order reconstruction method mentioned before.

The CFD model is independent of the grain design, allowing the user freedom to do a variety of grain designs. SRMs receive mass flow from the propellant and therefore sufficient source terms can be added to the governing equations to simulate this effect. By ensuring all physical effects are accounted for and can be captured within this simulation, it should be possible to predict effects such as erosive burning, effects of ignition transients, and eventually even combustion instability related phenomena.

The next phase of development will be to integrate the LSM method developed with the OpenFOAM solver.

VI. Conclusion

Rapid detailed surface regression and fully coupled CFD internal ballistic analyses of the most feasible grain design of numerous grain designs evaluated with a rapid grain design tool utilizing simple steady state internal ballistic equations has been made possible by utilizing the level set methods of Osher and Sethian[5] to create detailed grain burn back surface analyses of complex grain designs along with rapidly expanding open source tools available to develop a fully coupled CFD internal ballistic analysis. These advanced tools has made it possible to very soon in the design process evaluate and tailor important design drivers such as for example erosive burning for high volumetrically loaded grain designs which is not possible with only a simplistic rapid grain design tool utilizing simple steady state internal ballistic equations.

References

[1]Fleeman, E., *Tactical Missile Design*, No. ISBN-10: 1-56347-782-3, AIAA, 2nd ed., 2006.

[2]Waliskog, H. A. and Hart, R. G., Investigation of the Drag of Blunt-Nosed Bodies of Revolution in Free Flight at Mach Numbers from 0.6 to 2.3, Research Memorandum L53D14a, NACA, Langley Aeronautical Laboratory, Langley Field, Virginia, June 1953.

[3]G P Sutton. Rocket Propulsion Elements. 4th ed., 1975.

[4]Davenas, A., *Solid Rocket Propulsion Technology*, Pergamon, Oxford, England, 1993.

[5]Osher, S. & Sethian, J. A. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *Journal of computational physics* **79,** 12–49 (1988).

[6]Mauch, S. A fast algorithm for computing the closest point and distance transform. *Go online to http://www. acm. caltech. edu/seanm/software/cpt/cpt. pdf* (2000). at <http://csdrm.caltech.edu/publications/cit-asci-tr/cit-asci-tr077.pdf>.

[7]Sullwald W., Smit G.J.F, Steenkamp A. and Rousseau C.W. "Grain burn back analysis using level set methods", *49th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, 15 - 17 July 2013, San Jose, California.

[8]Nessyahu H. and Tadmor E., "Non-oscillatory central differencing for hyperbolic conservation laws", *Journal of Computational physics,* 1990, pp. 408-463

[9]Jiangand G. and Osher S. "Efficient implementation of weighted ENO schemes", *Computational physics* , 1996, pp. 202-228.

[10]De Kock E.R., Smit G.J.F., Knoetze J.H and Rousseau C.W. "Implementation of an internal ballistics model in OpenFoam", *49th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, *15 - 17 July 2013, San Jose, California*.