UNIVERSITEIT VAN AMSTERDAM
Faculteit der Exacte Wetenschappen

TNO Defensie en veiligheid

Bachelorproject verslag Natuur- en Sterrenkunde, omvang 12 EC, uitgevoerd als stage bij TNO defensie en veiligheid in de periode 26-04-2010 tot 01-07-2010

Solid propellant grain geometry design, a model for the evolution of star shaped interfaces

Auteur:
Arnon Lesage,
5795656

Begeleiders:
Ir. Francois Bouquet
Dr. Rudolf Sprik

August 9, 2010

Contents

Nomenclature

$\dot{m}$	Mass flow
$\dot{r}$	Burn rate
π_K	Temperature sensitivity of pressure
ρ_c	Chamber gas density
ρ_e	Exhaust gas density
ρ_p	Propellant mass density
$\theta(\theta^*)$	The angle from the origin to a point on the off-center circle
θ^*	The angle from the origin of the off-center circle to a point on the off-center circle
θ_i	Angles representing the limits of the angular range over which the piecewise functions of $r(\theta)$ work
$\theta_i(bd)$	Angles representing the limits of the angular range over which the piecewise functions of $r(\theta, bd)$ work
a	angle used in geometry,= $\frac{\pi}{N}$
A_t	Nozzle throat area
a_0	Propellant constant that depends on initial grain temperature
A_b	Burn Area
A_e	Nozzle exit Area
A_t	Throat Area
b	angle used in geometry,= $\frac{k_1\pi}{N}$
bd	Burn Depth
c	Angle used in geometry,= $\frac{k_2\pi}{N}$
F	Thrust
F_r1	The initial Radius of the first fillet
F_r2	The initial Radius of the second fillet
IR	Inner radius
k	Specific heat ratio
k_1	Fraction of angles, = $\frac{b}{a}$

k_2 Fraction of angles, $= \frac{c}{a}$

N Number of star points

n Combustion index

OR Outer radius

P_a Outer pressure

P_c Combustion chamber pressure

P_e Exit pressure

Pa Point angle

R Gas constant, or later on circle radius

$r(\theta)$ Radial function describing the initial interface

R_c The radius function of an off-center circle

$r_i(\theta)$ Radial function describing a single piece of the geometry

$r_i(\theta, bd)$ Radial function describing a single piece of the geometry as it evolves as a function of burn depth

$rf_i(\theta)$ Radial function describing the fillet piece of the geometry

$rf_i(\theta, bd)$ Radial function describing the fillet piece of the geometry as it evolves as a function of burn depth

T_c Combustion temperature

V_c Grain cavity volume

V_e Exit gas velocity

Samenvatting

Vaste stuwstof motoren bieden geen mogelijkheid om zich actief te reguleren. Actief reguleren houdt in dat het mogelijk is om tijdens het vliegen de stuwkracht te veranderen naar wensen. Na het ontsteken, verbrandt de stuwstof en levert het een bepaalde stuwkracht. De stuwkracht is afhankelijk van de hoeveelheid verbrandingsoppervlak van de stuwstof. Door de stuwstof een bepaalde geometrische vorm te geven, is het mogelijk de hoeveelheid stuwkracht toch passief te kunnen reguleren. In tegenstelling tot actief reguleren, houdt passief reguleren in dat de stuwkracht veranderingen vantevoren, dus voor het ontsteken worden bepaald. Tijdens het verbranden van de stuwstof, zet de vaste stuwstof zich om in gas, en verandert de geometrische vorm van de stuwstof massa. De stuwstof geometrie moet dan ook na het veranderen nog steeds aan de stuwkrachteisen voldoen. Hiervoor is een model nodig dat de manier waarop de geometrie verandert kan simuleren. Hiermee kan het model berekenen wat de stuwkracht is van een bepaald geometrie ontwerp tijdens zijn gehele verbranding. Het model dat hier wordt bestudeerd zal de eerste stap zetten naar een optimalisatie model. Het optimalisatiemodel, het einddoel, is een model dat een stuwstof geometrie berekent die aan bepaalde prestatie-eisen voldoet. De stap die in dit onderzoek wordt bestudeerd is niet de optimalisatie, maar het berekenen van de prestaties van cilindrische geometrie, gebaseerd op twee dimensionale stergeometrie. Het model is uitgewerkt in MatLab, in opdracht van TNO Defensie en Veiligheid. Het model berekent de prestaties op een manier die optimalisatie mogelijk maakt. Dit was nog niet mogelijk met de bekende methodes, en daarom werd het huidige model ontwikkeld.

1 Introduction

TNO Defence Security and safety provides many solutions to the overall safety and security as a strategic partner of the Ministry of Defence. TNO has a long history in researching propellants. The research began in the middle of the 1980's, with the design of igniters and starter engines for the Ariane 5 main engine (Vulcain). The Ariane 5 is a civilian rocket used as a carrier of satellites into orbit.

Over the years, development provided detailed methods by which the performance of a given propellant and propellant-geometry can be determined. The currently used software, GDP[1], is able to calculate the performance and burn behavior in detail. Yet, it does not provide fast analysis or a basis for optimizations of the propellant geometry, also known as the grain geometry. Optimization is an essential ingredient for the development of grain geometry. The grain, which is the propellant bulk, is developed when the requirements of the rocket are known. It is therefore needed to have a method that calculates a grain-geometry based on the given performance requirements. Further calculations and fine tuning of the geometry can then be done using the current software models.

The project is split into two separate projects. The first is the development and implementation of a model that provides fast performance calculations of a given propellant and propellant-geometry. Different approaches are available which will be discussed. The chosen approach is implemented in MATLAB. Fast analysis of the grain is the key ingredient of the chosen approach. Speed is important because in the next step, optimization, simulation of many different grains takes place. It is therefore desired to have an approach that minimizes the time of a single analysis to under a second. The second part of the project is the optimization phase. A grain geometry will be calculated that provides the requirements for the mission at hand. The subject discussed is the first part of the project, the analysis of a grain geometry. In (4.8) a discussion follows that provides some points of thought for the second part of the project, the optimization.

The model produced will concentrate on burn area evolution, as a function of burn depth. This means it will not take into account all factors affecting the burn rate, and with that the production of thrust. These factors will be discussed though in order to understand, why these can be examined separately and how future implementation is possible.

2 Solid rocket motors

A solid rocket motor (SRM) is a machine that provides thrust. Every SRM provides a different amount of thrust depending on payload, destination and other factors. But thrust is not the only performance variable of a SRM. For example the mass of the propellant is also a major factor. As the propellant burns, it's mass decreases, allowing higher acceleration. Therefore performance assessment models also look at specific impulse, mass flow and other variables. This model will only examine a few aspects of the performance of a SRM, although many others can also be examined.

The solid rocket motor is one of two major classes of motors, where liquid propellant is the other kind. What makes solid unique versus the liquid propellant, is the simplicity of the motor. The solid propellant contains the fuel and oxidizer. It is therefore enough to ignite the propellant and no other chemical has to be added for combustion to take place. There are different propellant compositions, usually double base or composite, that all have different properties. Burn rate, ignition temperature, flame temperature and other properties that can all be found using experimentation and modeling.

The simplicity of the solid propellant is also it's drawback. Requiring nothing external to burn, the SRM's burning cannot be regulated in-flight by providing more or less fuel, as in engines. A form of active, real time, control of the SRM's thrust is not possible. Once ignited it will continue to burn until combustion stops. Combustion stops when the propellant has depleted or when it is not possible to sustain combustion conditions (temperature and pressure). The only form of active thrust control is a thrust termination system. A thrust termination system allows shutdown of the motor, but this usually destroys the motor, and does not allow re-ignition. These systems are usually used as a failsafe to stop failing rockets, or when a stage rocket is separated from the main rocket.

Active control of an SRM is not possible and therefore interest lies in a form of passive control. It is possible to determine in advance at which phase of the SRM's burning it will provide more or less thrust. A model that calculates when a rocket provides more or less thrust is able to predict a way of passive control of the thrust of an SRM.

2.1 Passive regulation

The ultimate goal of the project is the optimization of a propellant grain. Requirements will be provided in the form of a thrust profile (a gabarit curve), with an allowed margin of error (Fig. 1). The thrust profile is a thrust vs. time diagram. Different missions require different thrust profiles, either rising thrust, neutral or more complex profiles (Fig. 2). The goal is to provide a grain configuration that meets the specified thrust profile requirements, and falls within the allowed margins of thrust. In the first part of the project we need to be able to produce a thrust diagram of a certain grain geometry. We will mostly be able to simulate burn area and burn rate though, so how these exactly provide the

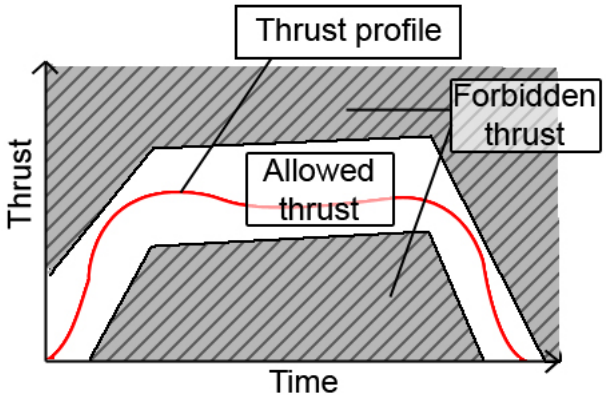


Figure 1: A thrust profile. It shows the way performance requirements are expressed, the white area is the area where within it's margin we wish to have the thrust profile. The red profile fulfills the requirement.

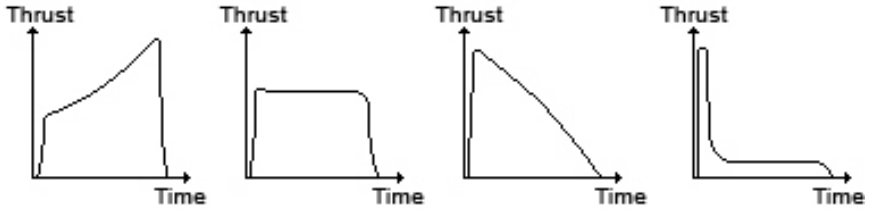


Figure 2: A selection of thrust profiles, showing progressive, neutral, regressive and more exotic burning profiles.

thrust is a subject that will be discussed in a later section (2.2). With passive control as a goal, all factors that affect thrust are examined. Narrowing down the factors to those important variable factors that can be modeled and allow for easy passive control. After ignition of the motor, the pressure first builds up until it stabilizes. The assumption of the model is a quasi steady state where the pressure is stable. The transient states describing ignition and extinction, are then best described by the current, and more detailed software GDP.

The thrust provided by an SRM is a function of several factors. Among them the propellant, the geometry of the propellant and design of the motor and rocket. The rocket and motor design is basically the design of the nozzle and the rocket hull, the materials used and so on. The design is predetermined and does not change flight performance in-flight. Moreover other factors supply the requirements for the rocket hull and nozzle. We are then left with an acceleration that is a function of the composition of the propellant and the geometry of the

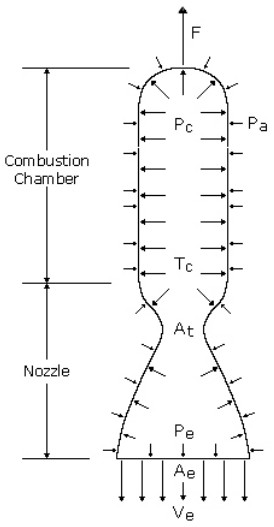


Figure 3: SRM definitions showing: Combustion pressure P_c, Combustion temperature T_c, Outer pressure P_a, Throat Area A_t, Nozzle exit Area A_e, Exit pressure P_e and Exit gas speed V_e providing the thrust F.

propellant. In the following sections these are discussed, narrowing them down to the most important effects.

Although thrust is described as the main performance variable of an SRM, this is not entirely true for igniters. Igniters are special kind of SRM that have to ignite the main rocket motor. This is done by exhausting hot gas into the main motor's combustion chamber. What makes igniters unique is then that thrust is of lesser importance and mass flow, and thermal energy is where interest lies.

2.2 Thrust and mass flow

The model developed specializes in determining the burn area vs. burn depth of a certain propellant geometry. To calculate a thrust diagram, the burn rate, the nozzle, the propellant and other factors are taken into account. I will attempt to explain briefly how these factors come into play. A few definitions can be seen in (Fig. 3), which are at play in a typical SRM.

A few assumptions are made to make matters simple. The first is the assumption of isentropic flow, which translates to an ideal rocket with adiabatic gas expansion, and no thermal losses. This is not entirely true but the effects are small for SRM's. Moreover, the assumption is made of a quasi steady state, where mass flow through the nozzle is constant with $m_{in} = m_{out}$. The last assumption is not wrong, but it is not correct for the entire duration of combustion, at ignition and extiction we are not at a quasi steady state. Therefore these stages will be modeled with other software.

The thrust F contains two terms, The first is called "Momentum thrust", and the second "Pressure thrust".

$$F = \dot{m}V_e + (P_e - P_a)A_e \tag{1}$$

The first term the "Momentum thrust" $\dot{m}V_e$, is a function of the mass flow $\dot{m}$ and V_e the speed of the gas when exiting the nozzle. The second expression of (1), the "Pressure thrust" is a function of the ambient pressure outside of the rocket P_a, the pressure of the gas when it leaves the nozzle P_e, and that difference multiplied with the area of the nozzle-end where these two pressures come in contact A_e.

The mass flow of gas out of the nozzle can be expressed as $\dot{m} = \rho_e V_e A_e$. Where ρ_e is the density of the exhausted gas, V_e it's speed and A_e the surface through which it moves when exiting the nozzle. But the mass flow is assumed to be constant $m_{in} = m_{out}$ and so, at an earlier phase of the combustion, the mass flow converted to gas comes from the solid propellant. The rate at which the propellant mass is converted into gas can be expressed as:

$$\dot{m} = \rho_p \dot{r} A_b \tag{2}$$

Where ρ_p is the mass density of the propellant, $\dot{r}$ it's burning speed and A_b the burning surface.

The thrust is thus a function of many variables, but the last expression for the mass flow (2) is of great interest: it is the most dominant term and because in igniters mass flow is the important performance variable. The mass flow (2) is a function of burn rate and burn area, which are both variable and predictable and therefore provide a way of passive control over mass flow and thrust.

All mass flow variables are now examined to see what affects them. The model specializes in the geometry and therefore a separation is needed of the mass flow (2) variables, that are a function of the modeled geometry, and those that are not. Burn rate, geometry and propellant density are easily described. Geometry is mostly discussed from section (3) onwards, but then the evolution of the grain is examined. Basically A_b is the surface of the geometry. The value of A_b can be found using geometry, it changes with time though and so is best described as a function of burn depth $A_b(r)$ (later on $A_b(bd)$) or time $A_b(t)$. The propellant density ρ_p is a propallant property which is easy to determine, and is constant at all times. Last is the burn rate $\dot{r}$ which is discussed next.

2.3 Burn rate

Burn rate is one of two major variables of the mass flow, yet many factors affect the burn rate itself. Composition of the propellant plays a major role but is predetermined. Moreover the composition is usually the same throughout the entire propellant mass. So by experimentally determining the properties of the propellant composition we can leave out much of it's properties as they will not have an effect on variable performance. Therefore if the other affecting factors are negligible the burn rate is very predictable. The conditions affecting the burn rate are[2][3]:

- First and foremost the pressure in the combustion chamber.
- Initial temperature of the propellant.

- Gas flow along burning surface.
- Motion of the rocket (fast spinning for example).

The research towards the effects of these conditions, is not yet able to provide an analytic prediction. But the effects of each of the conditions separately have been studied, and provides empirical predictions[2][3]. Following is a deeper examination of these conditions as a background to why none are taken into account by the model, and how they could be implemented. These are not the point of focus in this model, as they have already been modeled using other models. Our focus still lies in the geometry evolution, but it is still necessary to understand what their effect might be.

2.3.1 Pressure and burn rate

Experimental testing of propellants provides burn rate's dependence on pressure. Quick examination of such experimental measurements[2] provides the following expression[4] for the relation

$$\dot{r} = a_0 P_c^n + b \tag{3}$$

Where $\dot{r}$ is the burn rate, P_c is the pressure, a_0 and b are functions of the initial temperature of the propellant and n is known as the combustion index. This relation can be simplified though, for the case of rockets where b is usually very small[3]. The simplified result

$$\dot{r} = a_0 P_c^n \tag{4}$$

This is known as the Saint-Robert's or Vieille's law. Where a_0 and n are found empirically for a certain propellant. They usually apply for a certain range of pressures. A set of different values for a_0 and n can provide the needed relations between burning rate and pressure throughout the combustion.

2.3.2 Temperature and burn rate

Temperature affects the rate at which chemical reactions take place. Therefore the initial propellant temperature affects the burning rate. It is therefore common to place the rocket in a temperature controlled space, or at least protect from the sun prior to ignition. Moreover initial temperature requirements are determined, so that if conditions exceed the conditions, launch is delayed. As a rocket in-flight is exposed to extreme temperatures, from 220K up-to 344K, it is important to see how the propellant performs in these extreme variations. A typical composite propellant experiences a variation of up-to 20% to 35% in chamber pressure P_c[2]. Moreover, for such temperature variations the thrust operation time varies with the same percentage. Nonuniform temperature within the grain may have an even more disastrous effect as the pressure may not be symmetric and the thrust vector alters. The effect of grain temperature on pressure is expressed as $\Delta P = P_0^{\pi_k \Delta T}$. Where ΔP is the variation of

pressure from the reference pressure P_0, at a temperature variation of ΔT with π_k an experimentally measured coefficient known as the *temperature sensitivity of pressure* at a constant burning area (K). By determining the a_0 parameter for the correct reference temperature, and giving the entire propellant that uniform temperature, we can neglect the effect of a temperature difference.

2.3.3 Erosive burning

Erosive burning[5][6] is the increase in burning rate because of the fast flow of hot gases along the burning surface. This problem mainly exists near the nozzle where gas flow is fastest. This is primarily a product of the pressure gradient within the combustion chamber. This results in an uneven burning along different sections of the grain, which would make for a very hard to predict thrust profile. Erosive burning is strongest when the flow is fastest. So when the amount of burning propellant is large, we have a large amount of gas. And it flows fast when it has to go through a small opening. We can therefore say that erosive burning is a function of $\frac{A_b}{A_p}$. With A_b burn area, and A_p the port area or the cross section area of the grain cavity. The larger this fraction the more likely is the occurrence of erosive burning. In small igniters this is usually a smaller scale effect than in large SRM's, but it can be of importance in both cases. The model does not take erosive burning into account at the moment, which is one of the subjects the more detailed software can handle.

2.3.4 Other effects

Many other effects take place in a nonideal rocket engine. None will be taken into account, as this would require highly complex computations. Among these effects are erosion of the nozzle throat and movement of the rocket motor. Both of these are assumed to not pose a big problem in our case. When erosion of the nozzle throat takes place, the nozzle throat grows bigger as it erodes because of the the high temperature. Nozzle erosion is basically a variable A_t which we have seen in (2.2). This affects pressure build up in the combustion chamber, among other things. Rocket movement can also have an effect. A spinning rocket will create higher pressures at the outer most area of the combustion chamber. Higher pressures affect burn rate, and therefore movement is not desired. These effects can play a role, but simply complicate matters. The model does not take these into account, although it could be done in future versions.

2.4 Mass flow and pressure

All variables contributing to the mass flow (2) have been accounted for: the variable area $A_b(r)$, a geometric value, the constant propellant density, ρ_p, and the burn rate, $\dot{r}(P_c)$, a function of the pressure and constants as described by (4). Therefore, the missing piece is the chamber pressure, P_c. Using the principle of conservation of matter, we can derive another relation. The amount of propellant mass burned per unit time (2) is equal to the sum of change in mass

within the combustion chamber, and the mass flowing out of the chamber[2].

$$\dot{m} = \frac{\partial}{\partial t}(\rho_c V_c) + A_t P_c C_d \sqrt{\frac{k}{RT_c}(\frac{2}{k+1})^{\frac{k+1}{k-1}}} \tag{5}$$

Where $\dot{m}$ on the left is the mass flow (2). The first term on the right is the change in mass amount in the chamber grain cavity, a function of the chamber gas density, ρ_c, and grain cavity volume V_c. The second term on the right expresses the mass flowing out of the nozzle. A function of nozzle throat area, A_t, chamber pressure P_c, chamber gas temperature T_c, R the gas constant, and k the specific heat ratio. This expression can be simplified using to[3]:

$$\dot{m} = \frac{\Gamma C_d P_c A_t}{\sqrt{RT_c}} \tag{6}$$

Now all the needed elements are there, combining (6) and (2) the pressure P_c is expressed as a function of two area expressions A_b and A_t. The burn area is a function of burn depth, $A_b(r)$, which is the part modeled by the algorithm and discussed in the following sections. And burn rate is a function of pressure $\dot{r}(P_c)$.

The model produces only burn area and burn depth numbers, and never makes the above transformation to mass flow and thrust. Yet, mass flow and thrust can be expressed with little added complication.

3 Grain geometry

Burn rate determines how fast a propellant burns. The amount of propellant actually available for burning at that burn rate is determined by the shape of the propellant mass, the grain geometry. The burning takes place at the surface, and the amount of surface a certain shape has is determined by it's geometry. The propellant mass is distributed on the inside of the combustion chamber. The mass is usually extruded or molded in a certain shape within the cylinder. In most cases the shape forms a cavity within the propellant mass, which is connected to the nozzle. This space is where hot gases move towards the nozzle. On ignition all exposed surfaces around this space will burn. When burning, the surface recedes as solid propellant turns to gas. The receding takes place in a direction that is normal to the surface.

The configuration of the grain or the shape of it defines how it behaves in time. The grain is in most cases cylindrically symmetric. In that case it is sufficient to examine a two dimensional cross section of the grain. Rocket models with variations along their length axis do exist but these require more complex calculations. While a complexer configuration does not necessarily provide a performance that is not achievable in a simpler constant 2D cross section model. This study will only focus on grain geometries that can be described by a two dimensional cross section. Another symmetry often found in the grain is a rotational symmetry. This is often the case because it is important to get a rotationally symmetric exhaust out of the nozzle, or else the rocket net thrust vector will not be straight.

Performance of a grain is usually measured in a thrust vs time diagram. Alternatively when neglecting burn rate effects, burn area vs burn depth, is used instead of the thrust/time. These diagrams can easily display if a grain is progressive or regressive, which translates to growing or diminishing thrust(Fig. 2). But these can be easily described with cylindrical grains burning in or out. More complex grain are required for more complex performances. A combination of progressive followed by regressive, or constant burn, usually require more exotic shapes, like a stars or fynocyls. (Fig. 2)

The performance of a grain the model looks at is ultimately the burning surface as a function of burn depth. In a two dimensional grain it is therefore necessary to calculate the circumference of the grain along the burning interface, as a function of burn depth. For a circular interface(Fig. 4) this is simple enough. The burning interface recedes radially so burn depth translates linearly to a greater circle radius. The circumference unlike the sphere area example mentioned above, is of a circle $C = 2\pi R$, and thus the thrust profile increases linearly $S = 2\pi RL$ with L cylinder length.

Grain geometry evolution is a problem that is best described by an interface. The interface is the front of the propellant that is burning. As the the interface propagates the it's geometry changes, and the the amount of burning surface changes. The way to describe a propagating interface is not so simple. The interface propagation problem at hand has similarities with other physical problems. The burning front in forest fires, paper, ocean waves and crystal

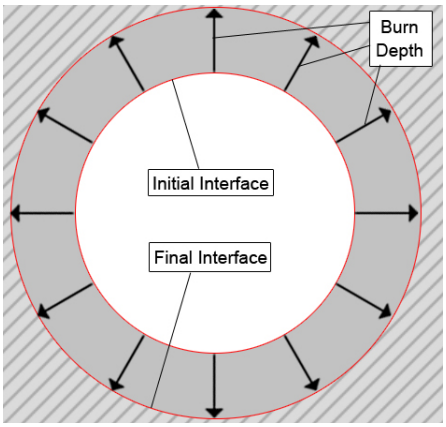


Figure 4: Circle or cylinder receding outward as it burns. The red lines show the initial and final burn interfaces. As the interface moves outward, its length grows translating to more combustion taking place.

formation are some of the examples where a propagating interface plays a role. Most problems are different than ours. The interface propagation speed is usually not consistent. When looking at burning of normal fuels, the burn rate is dependent on the amount of available oxygen. Convex surface are exposed to more oxygen than concave shape, and therefore propagation speed in many problems is curvature dependent. Propellant are different though, because the oxidizer is within the propellant the burn rate is consistent over the entire burn surface.

The property that sets the grain geometry problem apart, rises from the fact that the fuel and oxidizer are present within the propellant. This causes the burn rate to be constant over the entire interface.

3.1 Propagating interface

In this section the problems are examined that arise when trying to solve a propagating interface problem. A closed curve is considered, in a two dimensional plane. An exact approach fails and raises many problems, while a numeric approach can be relatively easy to achieve.

We begin with an initial closed curve $C(0)$ at time $t = 0$. Where $C(t)$ is the curve as it propagates in time. Propagation takes place in a direction normal to the curve, at a speed V. Parameterizing the curve as a function of θ seems logical, and for the sake of simplicity we take a curve that is best described in polar coordinates, and is similar to star shapes. $\vec{r}(\theta, t)$ is then the position vector that parametrizes the curve as it propagates. With the the boundary condition that $\vec{r}(2\pi, t) = \vec{r}(0, t)$. The normal $\vec{n}(\theta, t)$ at any point along the

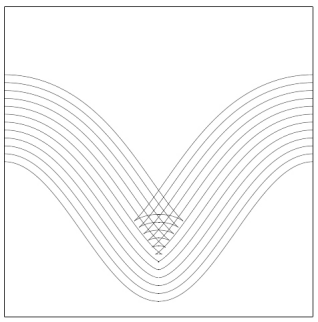


Figure 5: Interface evolution fails.

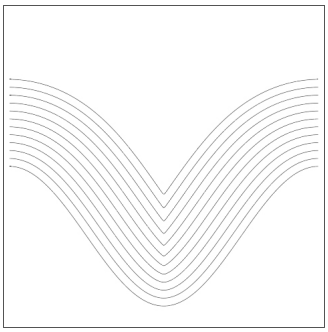


Figure 6: Interface evolution succesful.

curve is used for the propagation direction. From this follows that

$$\vec{n}(\theta,t)\cdot\frac{\partial\vec{r}(\theta,t)}{\partial t}=V(\theta,t) \tag{7}$$

with $V(\theta,t)=V_0$ being constant. This is also known as Eikonal's equation, and the solution is quite straight forward. This can be rewritten in terms of $\vec{r}(\theta,t)=(x(\theta,t),y(\theta,t))$ coordinates, which provides [7]

$$\frac{\partial x(\theta,t)}{\partial t}=V_0\frac{\partial_\theta y}{\sqrt{(\partial_\theta x)^2+(\partial_\theta y)^2}} \tag{8}$$

$$\frac{\partial y(\theta,t)}{\partial t}=-V_0\frac{\partial_\theta x}{\sqrt{(\partial_\theta x)^2+(\partial_\theta y)^2}} \tag{9}$$

Using these an attempt is made at the evolution of a shape. It can be seen in (Fig. 5) how this method fails. Round corners should form cusps but fail to do so. The solution used is indeed the correct solution that shows evolution for a propagating interface that evolves perpendicular to itself. But it is not physically the correct evolution, as the interface cannot cross the same point twice. The propellant is burnt after the first time the interface moves over it.

3.2 Interface propagation methods

It is obvious that a better approach is needed in order to solve this problem correctly. Either by approximation or exact. The requirements we have from the method or model is that it can primarily handle star like shapes, and that it paves the way to optimizations. We will now examine three approaches to see how they cope with those requirements, and what their weaknesses and strengths are.

3.3 Cellular automaton

This approach is very simple and thus very easy to implement. It does have a few major drawbacks. The idea is that the combustion chamber or cylinder

is converted into a grid of cells, which visually are pixels. Each cell is then assigned a value describing it's state. The initial values are initially determined by the grain configuration we examine. The states are for example "Burning", "Propellant" and "No propellant". Then we define rules that help evolve the grid. Every iteration the algorithm scans all cells and changes states according to the rules.

The strength of this method is that any shape or grain can be examined. Two methods were examined of inputting the initial grain configuration. One option is the use of an image file where the grain is visually sketched. This can be converted into a matrix where each pixel in the image is a cell in the matrix. The color of the pixel defines the value or state of the cell. Using images allows for the greatest possible versatility in grain design. Another option is based on parameters that define functions. These functions are then used to input the values within the matrix. This is not very versatile and not fast.

So we start with a grid of N by N cells. Using one of the methods above the initial state of each cell is determined. The possible cell states are as described above, "Burning", "Non-burning, Contains propellant" and "Non-burning, No propellant". After the construction of the initial grid or matrix, a loop code is used to evolve the grain. Every iteration the rules are used and the states of certain cells change. The rules used to evolve the grain are:

- If cell state is "Burning" and neighboring cell state is "Non-burning, Contains propellant", then change neighboring cell to "Burning".
- If cell state is "Burning", then change to "Non-burning, No propellant".
- Count the amount of burning cell to determine circumference or burning area.

The definition of a neighboring cell to a burning cell has a strong influence on the evolution. A neighboring cell is a cell that is a maximum distance from the burning cell. Lets call this maximum distance "neighbor distance". We then have a situation where a circle or radius "neighbor distance" of cells around the burning cells should start burning. But in a simple model this maximum distance of neighboring cells is small, approximately 1 or 2 cells. In that case the circle will not appear as circle but appear very "pixelated" (Fig. 7). The consequences of this for small values of neighbor distance can be seen in (Fig. 8). To make this work better the "neighbor distance" must be large, to form a nice circle. But then the grid would also need to be large, so N has to be large. This raises the biggest drawback of this method. The algorithm scans every cell of the N^2 cells, at every iteration. This makes for a calculation heavy algorithm. Moreover to know the state of the grid after a certain time all iteration steps before that have to take place as well. Both of these factors do not work well with future optimization steps.

This method is good if one wants to examine an eccentric shape that doesn't have any symmetries, or a shape that cannot be easily described, and therefore has to be sketched. Nevertheless this method is not suitable for optimization mainly because of the long calculation times.

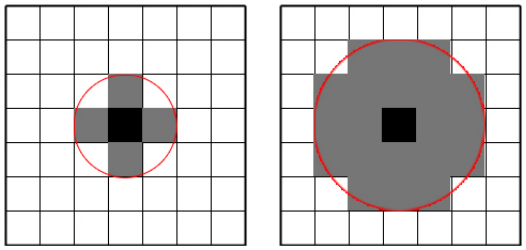


Figure 7: Two examples showing some variations in neighbor distance, the first with neighbor distance set to 1, the second set to 2.

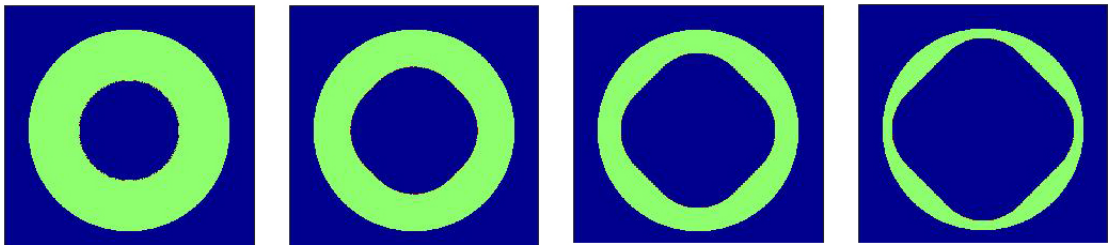


Figure 8: An example showing the incorrect evolution of a circular interface using neighbor distance set to 1.

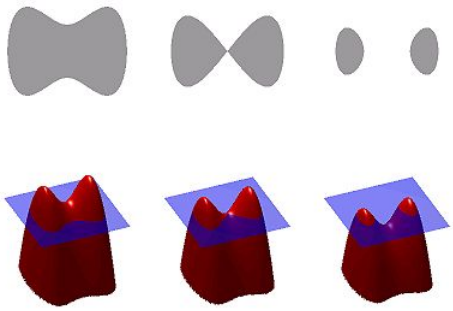


Figure 9: An example showing the way the fast marching algorithm translates a moving two dimensional interface into a stationary three dimensional surface.

3.4 Fast marching algorithm

Developed by Sethian J.A. in many articles[7][8][9] over many years. The fast marching algorithm doesn't try to follow and move the interface. Instead, a 2D interface is turned into a 3D surface. The cross section of the surface describes the interface at a specific time(Fig. 9). The surface has to be constructed first though. This is done by evolving it at every level, and contructing the next level. This is time consuming, and for a simple geometry with an acceptable resolution 10's of seconds are normal calculation time. The fast marching algorithms strength is it's ability to handle variable and inconsistent propagation speeds. Aslo it handles the formation of cusps and islands and other complexities well. This ability makes it a prime candidate for many other interface propagation problems. Yet it's weakness make unsuitable for optimization. Requiring long calculation times is one of the weaknesses, another is that to construct the surface all levels are needed until the level required is modeled. It is not possible to simply look at the interface evolution after a certain time without calculating all levels before. Algorithms based on fast marching algorithms are available for MatLab and Mathematica. These were tested and it was established that these method's strengths are not needed at the moment. We are not faced with complex speed functions, difficult geometries. Moreover it simply does not provide an algorithm that works well with optimization. This method would be very suitable for more complex and more versatile interface propagation problems.

3.5 Geometric solution of a parametric star

Using this method concentration is put on star-like shapes made out of straight lines and arcs. The star shape is defined by parameters, where the set of chosen parameters defines a unique star shape. After producing an initial shape the star evolution is modeled, using certain reference points within the shape that do

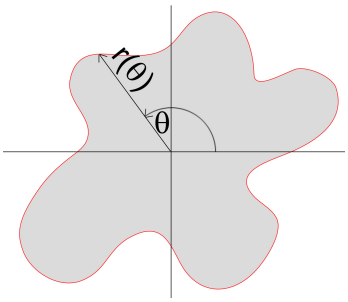


Figure 10: The radial function, $r(\theta)$, defined as the distance from the origin to the interface. The function $r(\theta)$ plotted over the range $\theta : 0 \rightarrow 2\pi$ describes the entire interface.

not change when evolving. As mentioned the star is made of straight sections or circle sections, and these sections are defined as functions. Polar coordinates are used and the function describing the interface is the radial distance as a function of the angle $r(\theta)$ (Fig. 10). Points of intersection between the different sections are determined. From these points of intersection it is possible to extract the order at which certain sections burn off or disappear. Using all these points, functions and disappearance time of certain sections, it is possible to construct the shape as a function of burn depth. This method is the only method that gives an exact solution to the problem. This is due to the fact that the function describing the shape is not continuous but a piecewise-defined function. Every piece of the piecewise function, is easily evolved as a function of burn depth. A straight line evolves by moving in a direction perpendicular to itself. The distance the line is moved is equal to the burn depth. A concave circle section as described earlier in the circle example (Fig. 4), evolves by growing. The radius will grow linearly with the burn depth(Fig. 11). Likewise a convex circle section diminishes (Fig. 12). It's radius shrinks linearly with burn depth, until the burn depth is equal to it's initial radius. In that case the circle's radius has reached zero. A circle section with radius zero translates to a sharp corner. In that case the circle section is omitted altogether from further steps of calculation and the bordering function pieces intersect each other. Using all different intersection points and function pieces the ranges of every piece of the piecewise function is determined. Using the function describing the shape information is gathered: the shape is plotted, the circumference of the shape is calculated and so are the remaining area/volume of propellant and port area.

3.6 The chosen method

The last method, the geometric evolution of a parametric star (3.5), was chosen as the favorite method that answers requirements. Built into it is the use of parameters as a definition of the geometry. While the other numeric methods

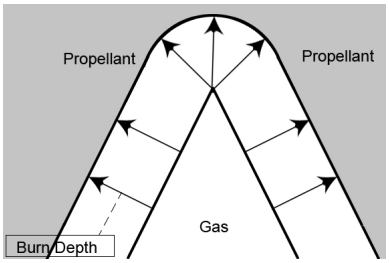


Figure 11: Interface evolution of a concave shape, a sharp cusp disappears.

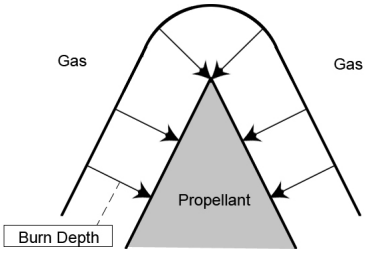


Figure 12: Interface evolution of a convex shape, forming a sharp cusp.

do not yet provide a simple implementation of the geometry. They can handle any geometry even if it is defined as an image, and provide it's evolution. But providing an image out of the performance requirements would be impossible. Trying out parameters on the other hand is already a simple form of optimization, and would be very easy to implement if the geometry is defined parametrically. The parametric definition of the geometry could also be implemented using the numerical methods, so the other reasons to make the choice was that a geometric evolution is analytic and therefore fast. A fast algorithm is very suitable for optimization.

4 Geometric evolution of a parametric star - Building the model

Examination of several models for interface propagation, has resulted in the chosen geometric evolution model. It is the fastest and provides the best performance for optimization. An examination of the inner workings of the model will now follow. First defining how the geometry's initial shape is constructed and then it's evolution.

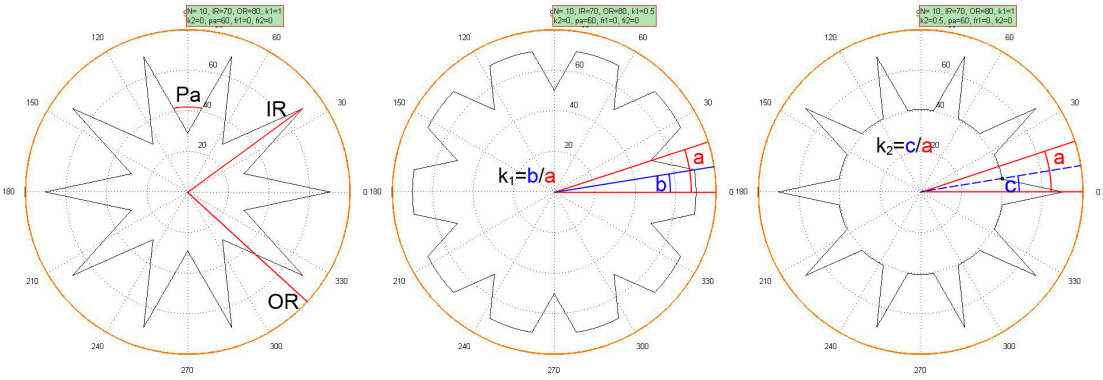
A parametric star is defined by a number of parameters. Each parameters defines a key aspect of the star geometry, it could be a length of a line, or an angle. After the definition of the parameters, the initial shape and therefore it's evolution are determined. It is important to keep in mind that the model we are developing should be able to provide initial optimization steps. The grain is thoroughly examined and fine tuned using the current grain modeling software GDP. It is therefore recommended to use parameters that are compatible with the parameters used by the current software. After constructing the initial shape, we can continue and see how this shape evolves as a function of burn depth.

4.1 The parameters

Defining a star can be done relatively simple, with only a few parameters. In this case a star like shape extruded out of a cylinder. The parameters defining a simple star would be(Fig. 15): the star point opening angle or Point Angle(Pa), the number of star points(N), and the outer and inner radius (OR) and (IR) respectively(Fig. 13a). Adding more complexity to the star is done by defining that the star points are spaced apart (Fig. 13b). The spacing is achieved by defining two angles a and b, that specify at what angle θ a few key points are placed. The angle a is deduced using the N parameter, $a = \frac{\pi}{N}$. The parameter k_1 specifies an angular fraction $k_1 \equiv \frac{b}{a}$. With the parameter k_1 and the angle a, the angle b is expressed. This is easier to work with than defining b explicitly. A b angle defined explicitly would mean that any time we change N we have to redefine b while in this way b is derived from other parameters N and k_1. As a function of parameters only the definition is $b = k_1 a = \frac{k_1 \pi}{N}$.

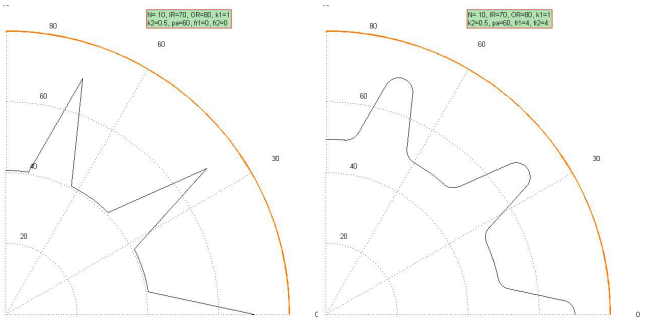
A higher order of complexity within the star is achieved by defining a star whose point are "cut off". It was chosen that the cut is a circular arc with it's center at the center of the star (Fig. 13c). The fraction of the star point that is cut off is defined using another angle c, which is deduced from the parameter k_2, defined as $k_2 \equiv \frac{c}{a}$. This definition is similar to the fraction we saw before as k_1, it defines the angular fraction of the cut off section c compared to the entire star point section a. As a function of parameters only the definition is $c = k_2 a = \frac{k_2 \pi}{N}$.

Another adjustment made to the simple star geometry is the introduction of rounding fillets. The fillets make the grain stronger, allowing it to withstand the vibrations and forces it is exposed to. Two fillets are defined, one concave,



(a) Simple Star (b) Added spacing between star points (c) Added point cut-off

Figure 13: How the parameters affect a star geometry



(a) Without the rounding fillet (b) With the rounding fillet

Figure 14: The effect of a rounding fillet.

and one convex. For each of these a radius is defined which is the fillet radius, named F_r1 and F_r2. The radius expresses the radius of the circle section that is the fillet. The fillet is tangential to both sections that it connects, which is how the center of the fillet circle is found.

Figure 15 shows the way the parameters of the star geometry are defined. In total eight parameter values have to be specified. The parameters shown in red in figure 15 are not directly defined. Rather indirectly by another parameter. The eight parameters are:

- N is the number of star points, which corresponds to $a = \frac{\pi}{N}$.
- k_1 is the fraction of angles, $k_1 = \frac{b}{a}$.
- k_2 is the fraction of angles, $k_2 = \frac{c}{a}$.

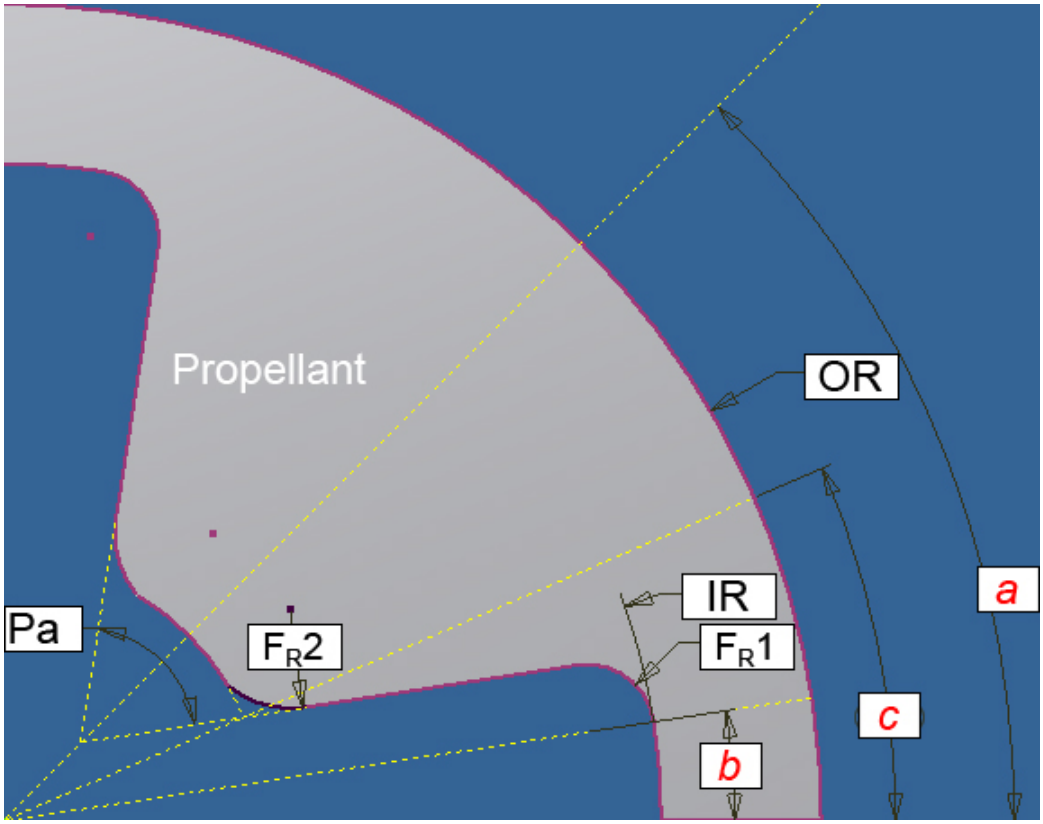


Figure 15: The parameters defining the star geometry.

- Pa is the point angle.
- OR is the outer radius where the propellant reaches the outer casing.
- IR is the inner radius defining at what outer radius the star points begin.
- F_r1 is the radius of the rounding fillet section $rf_1(\theta, bd)$.
- F_r2 is the radius of the rounding fillet section $rf_2(\theta, bd)$.

4.2 Initial shape

Well chosen parameters are very important for an easy calculation of the initial shape. Having just defined all the necessary parameters, it is now necessary to be able to construct the shape right. Visually it is easy to confirm that the initial grain description is right. When the function describing it is without errors the model can go on and calculate the circumference.

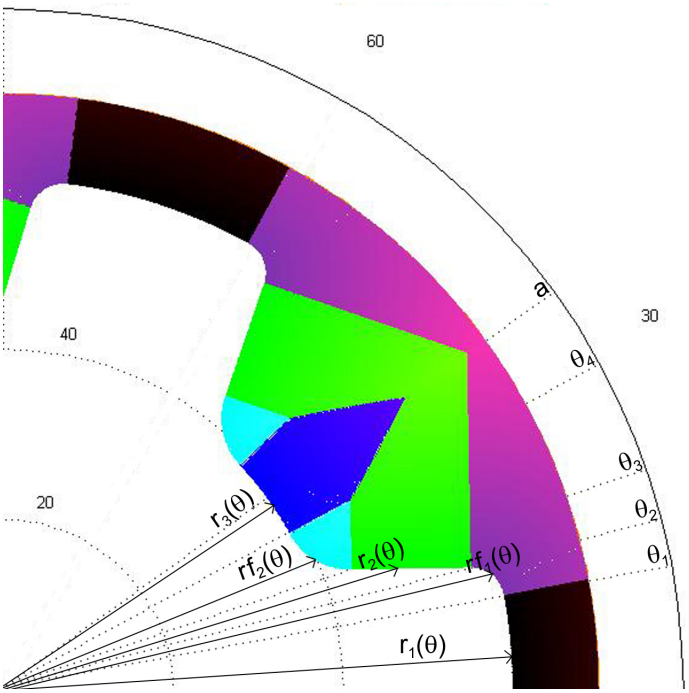


Figure 16: The 5 different piecewise functions, that describe the entire shape. $r_1(\theta)$ in black, $rf_1(\theta)$ in purple, $r_2(\theta)$ in green, $rf_2(\theta)$ in cyan and $r_3(\theta)$ in blue.

The function describing the shape is a polar function $r(\theta)$. For simplicity we would like to describe the smallest part of the star possible. It is obvious that symmetries are present, rotational symmetry means that we only need to describe an angular fraction of the star circular cross section. An angular fraction of $\frac{2\pi}{N} = 2a$, which translates to a cyclic permutation of $r(\theta) = r(\theta+2a)$. But an angular fraction of $2a$ of the star is also describing too much. An angular fraction a of the star is sufficient. The shape is mirrored along the a angle so that $r(\theta) = r(2a - \theta)$. So based on symmetry of the shape alone, we have narrowed down the part we have to describe to the function $r(\theta)$ in the range $0 \to a$.

The polar function $r(\theta)$ is thus our main interest. Because $r(\theta)$ contains sharp corners it's derivative cannot be continuous. It is therefore necessary to define it by way of piecewise functions. I use $r_1(\theta)$, $r_2(\theta)$, $r_3(\theta)$ for the different sections(Fig. 16). Where $r_1(\theta)$ and $r_3(\theta)$ are circle sections and $r_2(\theta)$ is a straight section. $rf_1(\theta)$ and $rf_2(\theta)$ describe $r(\theta)$ at the fillets. Eventually five different functions contribute to $r(\theta)$.

$$r(\theta) = \begin{cases} r_1(\theta) & \text{for } 0 \leq \theta < \theta_1 \\ rf_1(\theta) & \text{for } \theta_1 \leq \theta < \theta_2 \\ r_2(\theta) & \text{for } \theta_2 \leq \theta < \theta_3 \\ rf_2(\theta) & \text{for } \theta_3 \leq \theta < \theta_4 \\ r_3(\theta) & \text{for } \theta_4 \leq \theta < a \end{cases} \tag{10}$$

Here the angular range borders θ_i are the angles to the intersection points between the bordering sections. These angles define the range of angles over which each of the section functions is used.

Using the different parameter definitions an in-depth look is made into the function $r(\theta)$. To show the calculations involved more detailed focus is put $r_2(\theta)$ and later as a function of the burn depth, $r_2(\theta, bd)$.
The function $r_1(\theta)$ is the simplest function expression, it describes a circle arc with circle center at the star center. Therefore $r_1(\theta)$ is constant, which creates a circle section. Initially it's value is $r_1(\theta) = IR$, as defined by the parameters. The functions $r_2(\theta)$ and $r_3(\theta)$ are more complicated. We now first assume that fillet radius 1 (F_r1) and fillet radius 2 (F_r2) parameters are equal to 0, so the corners are not rounded. In that case we know that r_2 passes through the points (r, θ): $(r_1(b), b)$ and $(r_3(c), c)$.
$r_3(\theta)$ can be found geometrically to be

$$r_3(\theta) = \mathrm{IR}\frac{\sin(\frac{\mathrm{Pa}}{2} - b)}{\sin(\frac{\mathrm{Pa}}{2} - c)} \text{ for } F_r1 = F_r2 = 0 \tag{11}$$

$r_3(\theta)$ is shown to be constant as $r_3(\theta)$ is a circle section around the center of the star, similar to $r_1(\theta)$. So now we have two points, where through the straight line $r_2(\theta)$ goes (r, θ):

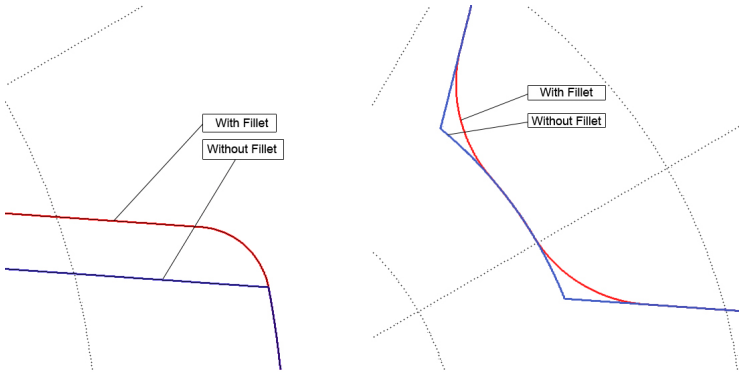
$$(\mathrm{IR}, b) \text{ and } (\mathrm{IR}\frac{\sin(\frac{\mathrm{pa}}{2} - b)}{\sin(\frac{\mathrm{pa}}{2} - c)}, c) \tag{12}$$

Using two points it is easy to find the line gradient being equal to

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{\frac{\mathrm{IR}\sin(\frac{\mathrm{pa}}{2} - b)}{\sin(-c + \frac{\mathrm{pa}}{2})}\sin(c) - \mathrm{IR}\sin(b)}{\frac{\mathrm{IR}\sin(\frac{\mathrm{pa}}{2} - b)}{\sin(-c + \frac{\mathrm{pa}}{2})}\cos(c) - \mathrm{IR}\cos(b)} \tag{13}$$

The gradient in use is a Cartesian gradient, so the two polar points are converted to Cartesian coordinate points where after the line is determined. The line function which is defined in Cartesian coordinates is then converted again to polar coordinates.

When the fillet radii are greater than zero we are faced with a slightly more complex situation. Each of the fillets is also defined differently. F_r2 is purely a rounding fillet and adapts itself to the corner it rounds. On the other hand, F_r1 whose center point is defined by the angle b. So when F_r1 is formed other sections have to move. First we examine the effect of F_r1, which are seen in



(a) F_r1 makes sections move (b) F_r2 just makes the corner round

Figure 17: The effect of a rounding fillet.

(Fig. 17a). The fillet 1 forms between r_1 and r_2, in a way that makes r_2 shift. The gradient of $r_2(\theta)$ remains fixed, but the line itself is moved. The fillet is then formed in between in a way that it is tangential to both sections it borders.

The way r_2 shifts because of F_r1 can also be seen in (Fig. 17a) as the straight section that moves. Each of the two points representing r_2 is shifted to it's proper location. Geometrically it can be found that the shift is equal to:

$$x_{i,correct} = x_i - F_r1(\cos(\theta_1) - \cos(\arctan(-1/m))) \tag{14}$$

$$y_{i,correct} = y_i - F_r1(\sin(\theta_1) - \sin(\arctan(-1/m))) \tag{15}$$

Where $x_{i,correct}$ and $y_{i,correct}$ representing the correct position of two points that form the straight line r_2. The expression $\arctan(-1/m)$ is the angle of the normal to r_2. Using these points the function that represents the straight line can be expressed.

The second fillet expressed by F_r2, doesn't affect the position of any of the bordering sections, it only makes the corner round. (Fig. 17b) The algorithm makes use of the given parameter F_r2 which is the radius of a circle. Using that information the algorithm looks for a circle with the required radius that is tangent to the straight line r_2 and to the circle section r_3.

The way the functions $rf_1(\theta)$ and $rf_2(\theta)$ are expressed also deserves some special attention. The problem is finding an expression for an off-center circle in polar coordinates, where the center point of the off-center circle is specified in Cartesian coordinates (x_c, y_c) (Fig. 18). Starting with a centered circle $r(\theta) = R_c$ with $\theta : 0 \Rightarrow 2\pi$. If I now take the Cartesian components $X_{circle}(\theta) = R_c \cos\theta$ and $Y_{circle}(\theta) = R_c \sin\theta$ and shift them so they are centered around (x_c, y_c), so that $X^*(\theta^*) = R_c \cos\theta(\theta^*) + x_c$ and $Y^*(\theta^*) = R_c \sin\theta(\theta^*) + y_c$. Here $\theta(\theta^*)$ is the angle from the origin to a point on the off-center circle, as a function of θ^* the angle from the off-center circle center to a point on the circle.

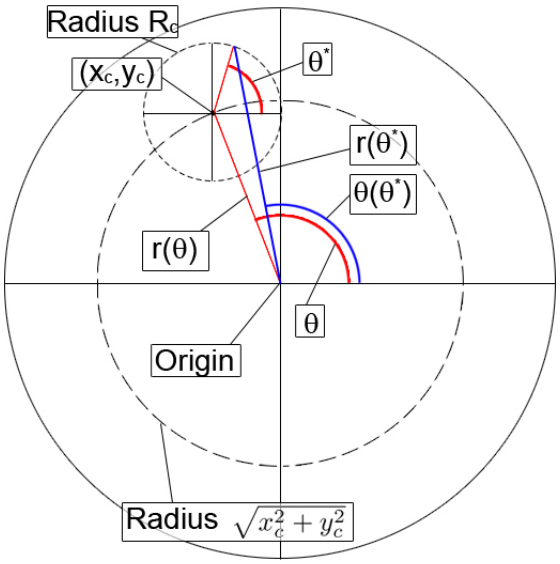


Figure 18: An off-center circle. The blue line shows the final product of interest the function $r(\theta^*)$ or $r(\theta)$.

Transforming back to polar coordinates gives:

$$r(\theta^*) = \sqrt{(R_c\cos\theta^* + x_c)^2 + (R_c\sin\theta^* + y_c)^2} \tag{16}$$

$$\theta(\theta^*) = \arctan(\frac{`rr_c\cos\theta^* + x_c}{R_c\sin\theta^* + y_c}) \tag{17}$$

Using these functions the circular sections of the fillets can be expressed. The functions run over a certain range of angles, but these angles are then specified as θ^* or as the angle when viewed from the fillet center point, and not the star center point. $R_c(\theta^*)$ is the radius function of the fillets. In later stages when the radius of the fillet is variable, this is the radius function that will be altered.

4.3 Evolution of the shape

Until now we have examined the static initial grain. Now the evolution of the grain is examined as a function of the burn Depth (Variable name *bd*). I take the initial shape function and make the necessary adjustments to be able to see the burn depth dependence. Now the functions vary with burn depth as do the intersection points that form the section ranges. $r(\theta) \Rightarrow r(\theta, bd)$

$$r(\theta, bd) = \begin{cases} r_1(\theta, bd) & \text{for } 0 \leq \theta < \theta_1(bd) \\ rf_1(\theta, bd) & \text{for } \theta_1(bd) \leq \theta < \theta_2(bd) \\ r_2(\theta, bd) & \text{for } \theta_2(bd) \leq \theta < \theta_3(bd) \\ rf_2(\theta, bd) & \text{for } \theta_3(bd) \leq \theta < \theta_4(bd) \\ r_3(\theta, bd) & \text{for } \theta_4(bd) \leq \theta < a \end{cases} \tag{18}$$

First participating shapes' evolution in general is examined. The participating shapes are (1) a concave circle section,(2) a convex circle section and (3) a straight line. (1) When burned a concave circle's radius will grow with burn depth. Any concave circle section with initial radius R_i will have a burn depth dependent radius $R_i(bd) = R_i + bd$. (2) Likewise to the concave case a convex circle section will have it's radius affected, unlike the concave case it's radius will shrink. An initial convex circle section with initial radius R_i will have a burn depth dependent radius $R_i(bd) = R_i - bd$. (3) Straight lines move in a direction perpendicular to themselves. So the straight line $(x) = mx + y_0$ will move perpendicular to itself in the direction $-\frac{1}{m}$. It will move a distance of bd in that direction.

In more detail it can be seen how the front described by $r_2(\theta, bd)$ evolves. It was already discussed how it shifts due to rounding of fillet radius F_r1. And it was just remarked that straight lines move a distance of bd in the direction perpendicular to themselves. So as before when $r_2(\theta)$ was handled, actually not the line is handled but two points on the line. The points are moved and through them the line described by $r_2(\theta, bd)$ is formed. The amount of shift is bd and the direction is perpendicular to the line. As before the gradient of the line described by $r_2(\theta, bd)$ is m, and the normal gradient is $-\frac{1}{m}$. The angle of that normal $\arctan(-\frac{1}{m})$. Taking the x and y component the points can be altered easily.

$$x_{i,evo} = x_{i,correct} + bd\cos(\arctan(-\frac{1}{m})) \tag{19}$$

$$y_{i,evo} = y_{i,correct} + bd\sin(\arctan(-\frac{1}{m})) \tag{20}$$

The rest of the shapes are circular with a certain radius. For the concave sections $r_1(\theta, bd)$, $r_3(\theta, bd)$ and $fr_1(\theta, bd)$. The radius is adjusted in the same way as described before to $R_i(bd) = R_i + bd$. For $r_1(\theta, bd)$ and $r_3(\theta, bd)$ this is simple as their defining function is their radius. So their adjustment are:

$$r_1(\theta, bd) = r_1(\theta) + bd \tag{21}$$

$$r_3(\theta, bd) = r_3(\theta) + bd \tag{22}$$

The fillet F_r1 is not centered around the star center, therefore the position function $rf_1(\theta, bd)$ does not represent it's radius, but it's distance from the center of the star (Fig. 16). The radius function has to be adjusted to have the bd addition. For the second fillet F_r2, the same applies. The burn depth adjustment applies to the radius function and not to the position function $rf_2(\theta, bd)$.

The adjustment in this case is negative. The radius function of $rf_2(\theta, bd)$ diminishes as $R_i(bd) = R_i - bd$. So in equations 21 and 22 the radius constant R would be exchanged with a radius function $R(bd) = R \pm bd$. Using all these calculation methods the expressions for all different sections are found. The goal now is to see at what range of angles each of these functions is used.

4.4 Intersection points

The functions $r_1(\theta, bd)$, $r_2(\theta, bd)$, $r_3(\theta, bd)$, $rf_1(\theta, bd)$ and $rf_2(\theta, bd)$ are known. But it is important to find the range over which each of these is used. The angles $\theta_{1-4}(bd)$ express the boundaries of the piecewise function ranges. Some require more complex calculations than others, which rely on the intersection points between certain sections. Not only bordering sections have intersection points. Sometimes sections burn off and their neighbors start to intersect each other. An analytic solution was used in the calculation of these intersections, and so it requires some algorithms. Two different algorithms are being used in all these cases, which describe the intersection between all different sections. (1) The intersection between two circles, (2) the intersection between a circle and a straight line.

At this point of the evolution care has to be taken in how the grain evolves. For almost any initial geometry design, some sections disappear during it's evolution. When a section disappears the bordering sections start to intersect each other. It is necessary to find a set of rules that know at what point which sections disappear. Five different cases have been identified, and separated using a set of rules. Each of these cases is like a burn phase where the total burning surface stays linear within the phase, the only factor breaking the linearity is if the grain is starting to reach it's outside border.

1. This phase takes place in most grains. It happens when both fillet radii are non zero. The phase describes a shape where all sections are present and burning. Usually this will end with the fillet section described by $rf_2(\theta, bd)$ disappearing. The interface in this phase is described by all five functions.

2. In many cases the second burn phase, after grain section fillet 2 has disappeared. all the other sections are still burning and present. The shape is then described using the four remaining section functions $r_1(\theta, bd)$, $rf_1(\theta, bd)$, $r_2(\theta, bd)$ and $r_3(\theta, bd)$. In this case the intersection between $r_2(\theta, bd)$ and $r_3(\theta, bd)$ provides the boundary to both of these functions.

3. The grain section described by $r_2(\theta, bd)$ has disappeared, yet the section described by $r_3(\theta, bd)$ has not. Therefore the fillet section $rf_1(\theta, bd)$ is intersecting $r_3(\theta, bd)$. And the intersection point provides the boundary to these functions.

4. The grain section described by $r_3(\theta, bd)$ has disappeared, yet the section described by $r_2(\theta, bd)$ has not. $r_2(\theta, bd)$ goes up-to the boundary of our shape, a.

5. The grain sections described by $r_2(\theta, bd)$ and $r_3(\theta, bd)$ have both disappeared. Now the section of the first fillet $rf_1(\theta, bd)$ continues up-to the boundary of our shape, a.

A grain always loses sections, never develops new, so a shape evolves only into a higher number phase never to a lower number. Also, a shape can only undergo one of the phases (3) or (4). The algorithm does not climb or drop down the ladder. For every value of bd it determines which phase the grain is in. This means that the history of the grain is not important, the algorithm knows immediately what the grain geometry is at a specific burn depth, without it having to know what it was in earlier burn steps.

Using this set of rules coupled with the intersection points, and combining all this with functions describing each section, the model is able to describe the grain evolution.

4.5 Burn area

Describing the shape of the grain is a great achievement of the model. Once the grain has been modeled it is a matter of simple calculation to find the circumference of the grain. And the circumference of the 2D cross section is converted to a burn area when multiplied by the length of the grain. The model doesn't do this step though, it only calculates the circumference. To calculate the circumference two different methods are used. (1) For straight lines like the one described by $r_2(\theta, bd)$, The distance between the two border points is determined. This is the distance between two polar points (r_1, θ_1) and (r_2, θ_2):

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} = \sqrt{r_2^2 + r_1^2 - 2r_2 r_1 \cos(\theta_1 - \theta_2)} \tag{23}$$

(2) The circular sections are defined by a radius and two border angles. The circumference of a circle section with radius R and beginning and ending angle θ_1, θ_2 can be easily determined using:

$$c = R(\theta_2 - \theta_1) \tag{24}$$

These two simple calculations on all five sections provide the circumference of the shape when multiplied by the $2N$ permutations that describe the entire geometry.

An extra complication is added when looking at the burning at later stages. As the interface approaches the casing the propellant at that section is depleted. The algorithm continues to evolve the interface, also outside the casing. Yet, the algorithm calculates all intersection points of the interface with the casing. Using the intersection points with the casing, the algorithm finds which sections or within the casing, which are partially in partially out, and which are entirely outside of the casing. Those that are entirely outside are ignored from circumference calculation and plots. Those that are within the casing are handled as usual. And those that are found to be partially outside, are treated that way. Their length are calculated using their intersection point with the casing, and

they are plotted up to the casing. The circumference only calculates burning propellant interface, and although the casing closes the interface shape in the late stages, it is not counted as burn area.

4.6 Achievements and Limitations of the algorithm

The algorithm is able to provide an analytical description of the grain. Following this the algorithm provides an analytic description of the grain's evolution. It is able to do so for a given set of parameters, for any given burn depth. The correctness of the evolution is examined visually, an incorrect evolution can be easily observed. An error in the evolution shows up as sections that overlap, or parts of the shape that are missing.

Many examples exist of when the algorithm fails, but the main limitation is probably when the Pa angle is too small. When the angle Pa reaches smaller values the star point goes beyond the origin into negative $r(\theta)$ values. In this case the entire algorithm breaks down, and we are left with odd results. Odd is what is troublesome because a grain is formed, but not a correct one, and one that also evolves incorrectly. If the algorithm is called to provide only the burning area(like intended) then there is no clear indication that the algorithm failed. So this leaves us with a two options. Either one has to always examine the grain's evolution visually, or we need to determine the parameter limits at which the algorithm fails.

The second option would be most ideal, and would fit well with an optimization algorithm, providing boundaries for parameters search. A third solution also exists and it is to alter the algorithm so it can handle the case of star points going beyond the origin. This would solve the problem but unlike the former solution it would not solve further problems. All the parameters used have certain limitations as a function of the other parameters, and the point going beyond the origin, is only one example of these failures. The fillet radii may not be too large, N cannot be zero, OR has to be set larger than IR, and in the same way there are many more limits that arise when the parameter definitions are examined. At no point does the algorithm check for any of these limits.

I will now summarize the problems that might make an invalid grain.

1. The number of star points, N, has to be larger than 1.
2. The outer radius, OR, has to be larger than the inner radius, IR.
3. The fillets 1 and 2 may not be too big so that they do not intersect.
4. Fillet 2 may not be too big so that it cannot go beyond the shape border, the line at an angle $\theta = a$.
5. Star point may not go beyond the origin into a negative radius range.

4.7 Results and comparison with the current software, GDP[1]

The developed software is able to construct star geometries based on parameters, evolve them and calculate burn area profiles. Eventually two different programs developed use the developed algorithm. The first is a function that accepts geometry parameters and a single burn depth value, as input. The output produced is the burn area, at the particular stage in the geometry evolution, defined by the given burn depth. This function could prove to be very useful in future optimization steps. The second program that uses the algorithm also accepts the parameters as input, as well as a range of burn depths, and a number of calculation steps. It produces two figures(Fig. 19), a burn area vs. depth profile for a geometry, and figure showing the evolution of the geometry at different burn stages or burn depths.

The current software, GDP[1], allows for grain modeling and combines numerical and analytic methods depending on grain geometry. A star like shape like ours is done analytically. But the software also allows for more complex grains, which are analyzed with a numeric algorithm which is not fast, and could take a several minutes to complete a geometry evolution.

A comparison of the grains and their evolution is shown in (Fig. 20). It can be clearly seen that the currently developed algorithm provides the exact same results. moreover the way the parameters in our algorithm were defined the parameters can be plugged into the old software package instantly. The differences one should keep in mind are that in the old software package a diameter is used while in ours a radius. And that we have two extra parameters (k_2, F_r2)that need to be set to zero in order to compare the results.

4.8 Improvements and beyond

The algorithm provides exactly what it was meant to do. It can provide an analytic evolution of a star like grain geometry. Optimization of the grain and implementation of limits is the following step. But other improvements that can be implemented are also possible. Some of these are discussed next.

The algorithm can model the evolution of a star like shape, and produce the burn area profile, yet that is not the thrust or mass flow the rocket provides. To be able to provide other performance variables we will need to take the burn rate into account. Overlaying the burn area vs. burn depth with the burn rate is basically making the burn depth a function of time. This means stretching out or compressing the profile at different times. In section (2.3) the burn rate was examined, and it was found to be a function of pressure or burn area, depending on the expression used. And pressure is a function of burn area, port area and other factors. A model that takes into account all these factors is complicated and is probably done numerically. It was chosen that our model will concentrate on the burn area vs. burn depth, while the complicated calculations are done using GDP. But in order to optimize a grain for specific thrust or mass flow requirements and not for specific burn area requirements, this will have to be

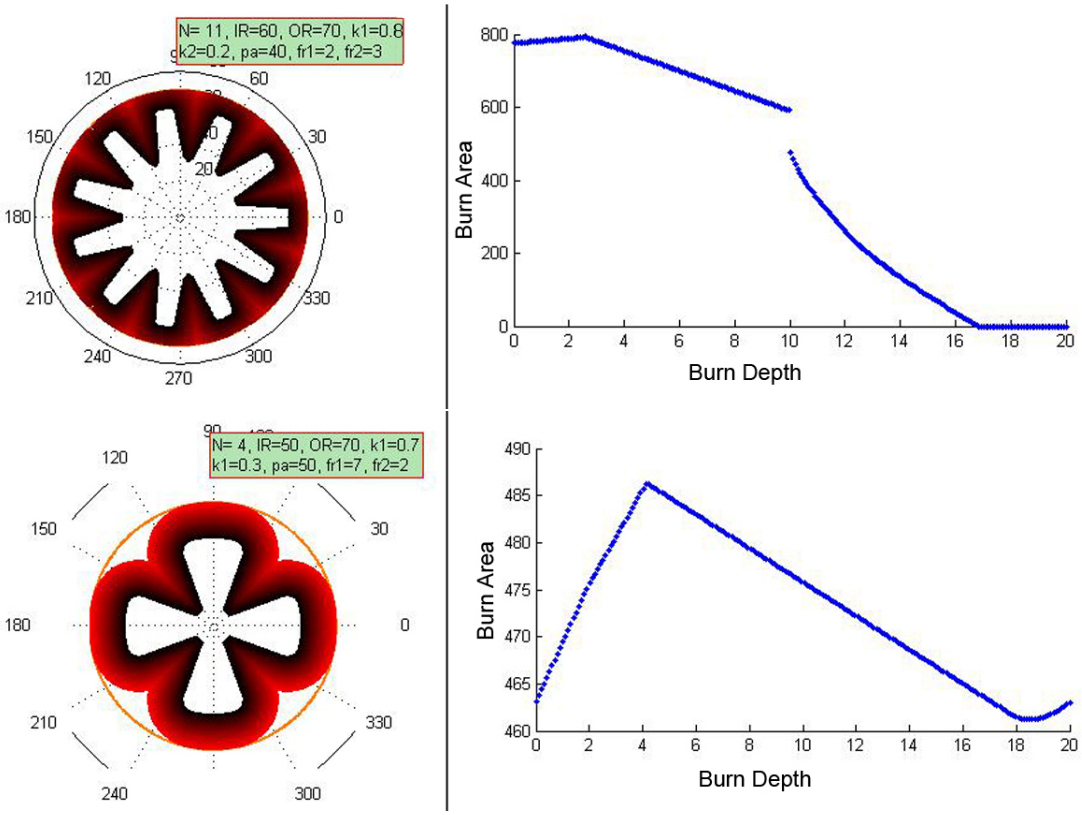


Figure 19: Two examples of algorithm output: The left part shows the evolution of the grain. The figure is made of many lines that show the geometry interface at different burn depths. The redder the interface line the later in the evolution we are looking. The right figure shows the burn area vs. burn depth profile of the grain.

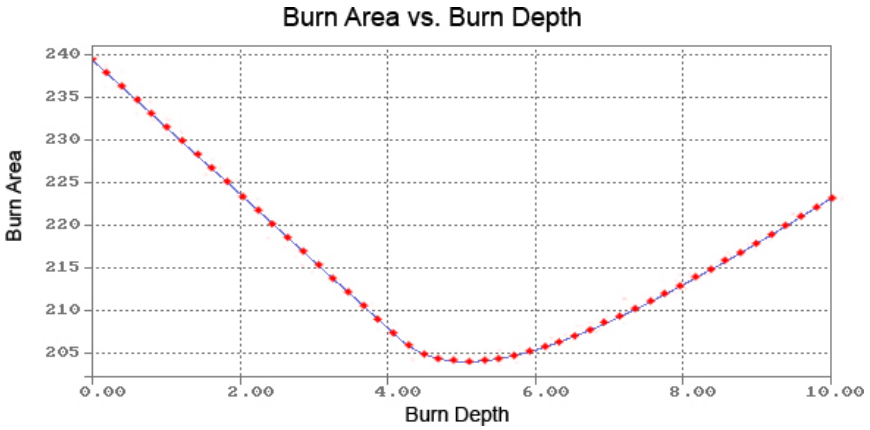


Figure 20: A comparison between the old software package (Blue line), and our algorithm(Red points). Both presenting the burn area vs. burn depth profile for a geometry with parameters: $OR = 35, IR = 25, k_1 = 0.8, k_2 = 0, Pa = 50, F_r1 = 4, F_r2 = 0$ and $N = 7$.

added in some way to the algorithm.

If and when burn rate is taken into account into the model, one can also examine the possibility of including effects like erosive burning(see 2.3.3), rocket movement and other effects(see 2.3.4) that usually affect different kinds of rockets. This would complicate matters, and the analytic strengths of this algorithm could be lost. That would again leave us with a model that does not lend it self well to optimization steps. Moreover the currently available software already takes these effects into account, and therefore I would advice against it.

As discussed before there should be some kind of verification of the parameters. Nowhere does the algorithm check if the parameters actually construct a shape that is correct. This is a good first step in the direction of optimization. The optimization needs to know which parameter values are allowed, therefore it needs to know the range of values each of the parameters can have as a function of the other parameters. This is done using geometry providing maximum and minimum limits to the parameter values. Only a few limit equations have to be set-up, as each equation is a function of a few parameters. Then the algorithm checks if the parameters obey the limits. These limits can then be used as the range of values to be attempted in the optimization steps.

A few methods are available for future optimizations steps. Our main goal once again is to have a number of points defining the thrust as a function of time or burn area/burn depth that are the algorithm's input, and a grain geometry expressed as parameters as the output. One method is by "intelligent" trial and error: the algorithm scans different parameter values until one of the parameter fulfills one of the required points, afterwards the algorithm tries to fulfill the next requirement point. Some parameters have more effect on earlier stages of the evolution while others more on the later stages, that is when use is made

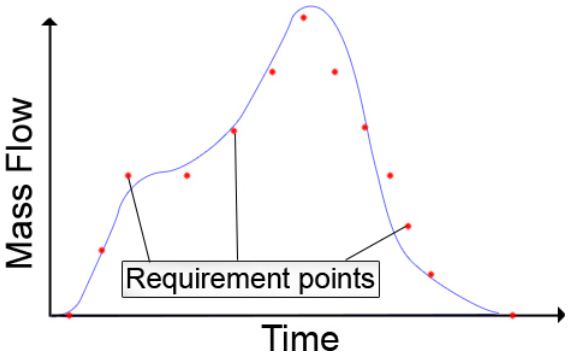


Figure 21: An example showing the way requirements are expressed using points.

of intelligent trial and error. Knowing which parameter to fine tune to affect a certain phase in evolution, reduces the amount of values that are attempted. This could be improved by using some knowledge about what kind of geometries provide certain profiles, and fine tuning the values.

There is one more option I would like to propose, which is more elegant, and makes better use of our analytic solution. Again we have a certain amount of points which are set as the requirements(Fig. 21). Our parameters are the variables. And the algorithm produces the function. By providing as many points as we have parameters we can solve the function for each point. This is easier said than done, and I will try to explain why. I described before that while evolving the grain undergoes several phases (Section (4.4)). Each of these phases is basically a different compilation of piecewise functions, with different boundaries. So each point in the requirement can be in a different phase of the same geometry. The phase or function we need to use at each point is only known once we know the parameters. Therefore this approach is not very easy to solve, and requires some additional steps. Perhaps as a variation of the least squares method this could provide a good solution.

5 Conclusion

The problem examined is the burning and thrust function of star-like grain geometries. Several different methods have been examined that can simulate the burning of the grain geometry. Eventually the method we have based it on is a geometric parametric approach. It is based initially on defining a star using parameters. The star is built out of sections, which all change and burn independently. This model is able to describe the shape of the propellant grain at all different burn steps, and does so at great speed and accuracy. A single step calculation time is approximately 0.1 to 0.01 sec. Moreover the model is very suitable for future steps that optimize the geometry according to performance requirements. The performance delivered is the burn area vs. burn depth, while ultimately it should deliver the mass flow vs. time profile. It was examined how the burn area profile should be translated into a thrust profile. The model algorithm is compatible with the current software, GDP.

References

[1] Grain Design Program, AED Electronics, http://www.aedelectronics.nl/gdp/gdp.htm.

[2] Sutton G.P., Wiley Interscience, Rocket Propulsion Elements, 1992.

[3] Barrere M. and Jaumotte A. and de Veubeke B. F. and Vandenkerckhove J., Elsevier, Rocket Propulsion, 1960.

[4] Geckler R., Butterworths scientific publications, The mechanism of combustion of solid propellants, 1954.

[5] J. C. Godon, J. Duterque, and G. Lengellet, JOURNAL OF PROPULSION AND POWER, Vol. 9, No. 6, Erosive Burning in Solid Propellant Motors, 1993.

[6] H. S. Mukunda and P. J. Paul, COMBUSTIONAND FLAME, 109:224-236, Universal Behaviour in Erosive Burning of Solid Propellants, 1997.

[7] Sethian J.A., Comm. in Math. Phys., 101, 487-499, Curvature and the Evolution of Fronts, 1985.

[8] Sethian J.A., J. Differential Geometry, 31, 131-161, Numerical Algorithm for propagating Interfaces: Hamilton-Jacobi Equations and Conservation Laws, 1990.

[9] Sethian J.A., Cambridge Monograph on Applied and Computational Mathematics, Level Set Methods and Fast Marching Methods Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science, Cambridge University Press, 1999.