ALMA MATER STUDIORUM
UNIVERSITA' DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA
Sede di Forlì

Corso di Laurea in
INGEGNERIA AEROSPAZIALE
Classe L-9

ELABORATO FINALE DI LAUREA
in Propulsione Aerospaziale

# SVILUPPO DELL'INTERFACCIA PER IL CARICAMENTO DELLA GEOMETRIA INIZIALE DEL PROPELLENTE SOLIDO DI UN MODELLO BALISTICO

CANDIDATO
Matteo Marcantoni

RELATORE
Prof. Fabrizio Ponti

CORRELATORE
Ing. Filippo Carra

Anno Accademico 2014/2015
Sessione II

# ABSTRACT

Il presente lavoro di tesi, nell'ambito della simulazione balistica di razzi a propellente solido, è volto alla creazione di un'interfaccia che permetta il caricamento delle geometrie dei motori desiderati, in formato SPP, nel codice di simulazione balistica ROOBOST come richiesto da Avio Spa. In particolare saranno descritte le procedure attraverso le quali tale interfaccia è stata realizzata e verranno analizzati i passaggi mediante i quali è possibile ottenere una mesh 3D del motore a propellente solido partendo dalla sua geometria in formato SPP. Si analizzerà inoltre il processo di chiusura di una mesh estratta da una qualsiasi iterazione della simulazione al fine di potervi effettuare delle simulazioni CFD sempre su richiesta della sopracitata azienda. Infine verranno mostrati e discussi i risultati ottenuti per meglio visualizzare il lavoro svolto.

**INDICE**

---

# INDICE DELLE FIGURE

# INTRODUZIONE

Nell'ambito della propulsione per mezzo di endoreattori di tipo chimico, i razzi a stato solido presentano numerose caratteristiche vantaggiose quali la compattezza, la semplicità di utilizzo e la scarsa manutenzione. Tuttavia presentano svantaggi quali l'impossibilità di gestire o modificare la spinta una volta innescata la combustione: l'unico sistema di controllo è quello della movimentazione dell'ugello di espansione dei gas combusti.
Si evince subito quindi, l'importanza delle simulazioni nella fase progettuale di un razzo a propellente solido (SRM, *solid rocket motor*), prima della sua realizzazione.

Il lavoro svolto è incentrato sulla realizzazione di un modulo integrativo per il caricamento delle geometrie in formato SPP del grano dei propellenti solidi, del codice di simulazione balistica *ROBOOST* (*Rocket BOOst Simulation Tool*), sviluppato presso il laboratorio di Propulsione e Macchine della facoltà di Ingegneria Aerospaziale dell'UniBo con sede a Forlì, per l'azienda Avio Spa la quale opera nel campo della propulsione spaziale.

Il modulo sviluppato permette all'utente di scegliere, tramite selezione da interfaccia grafica, la geometria desiderata che, appositamente convertita, viene scritta in un file da importare all'interno del CaD open source *SALOME*. Su tale geometria, sempre all'interno del CaD, viene in seguito effettuata un mesh (in base ai parametri preventivamente scelti dell'utente in un'apposita sezione della stessa interfaccia grafica). Tale mesh è quindi disponibile per essere caricata all'interno del codice *ROOBOST* per avviare la simulazione.

Nei capitoli seguenti, dopo una piccola introduzione alla propulsione a razzo, ai codici *ROBOOST* e *SPP* nonché al CaD *SALOME*, verrà trattato nello specifico il metodo di lettura e conversione delle geometrie, i relativi problemi e le soluzioni attuate, la realizzazione del concernente codice e il lavoro di chiusura delle mesh estratte durante una qualsiasi simulazione per potervi effettuare delle simulazioni CFD.
Infine verranno illustrati e discussi i risultati ottenuti.

# RAZZO A PROPELLENTE SOLIDO

## Schema base e funzionamento

Un razzo a combustibile solido è un razzo che impiega un motore a propellente solido. Questi tipi di razzi differiscono da quelli a propellente liquido per un più ampio *range* di applicazioni ed una minor manutenzione necessaria, tuttavia lo svantaggio principale è rappresentato dal fatto che la spinta di un razzo a stato solido non può essere controllata né regolata: dipende dalla forma impressa alla superficie del grano e dalla velocità di combustione dello stesso; un razzo a stato solido quindi una volta acceso non può essere spento fino ad esaurimento propellente.

Il processo di produzione e caricamento nel motore del grano può essere riassunto nelle seguenti fasi:

- Macinazione degli ingredienti fino al raggiungimento della dimensione media desiderata delle particelle;
- Pre-miscelazione e miscelazione del combustibile con l'ossidante;
- Colaggio nell'involucro del motore contenente la spina con la geometria che verrà impressa una volta solidificato il propellente;
- Vulcanizzazione del grano di propellente e rimozione della spina.

L'involucro del motore può essere riempito o attraverso un colaggio continuo del propellente (*continuos mixing*) o attraverso uno frazionato (*batch mixing*) usato prevalentemente nel caso di motorizzazioni di grandi dimensioni. Un colaggio di tipo frazionato presenta lo svantaggio che le diverse miscele di propellente possono risultare differenti sia dal punto di vista chimico, in quanto possono riscontrarsi diverse percentuali degli ingredienti, sia dal punto di vista particellare in quanto è assai difficile ottenere due macinazioni identiche.

Proprio le fasi di miscelazione e di colaggio risultano essere le più critiche per la determinazione delle proprietà balistiche del propellente, in quanto fortemente dipendenti dalla reologia del grano.

La principale differenza dai razzi a propellente liquido è data dal fatto che in questi ultimi i serbatoi del carburante (propellente e ossidante) sono separati dalla camera di combustione mentre in quelli a solido coincidono; durante la combustione infatti la superficie combusta del grano regredisce (di circa alcuni mm/s) facendo quindi variare le dimensioni della camera di combustione.

La definizione di spinta si ricava a partire dal principio di conservazione della quantità di moto:

$$F = \dot{m}u_e + (p_e - p_a)A_e$$

in cui con $F$ si indica la spinta, $\dot{m}$ la portata massica di gas che attraversa l'ugello, $u_e,\ p_e$, $A_e$ rispettivamente la velocità, la pressione e l'area nella sezione d'uscita dell'ugello, $p_a$ la pressione ambiente.

Inoltre supponendo che il flusso dei gas combusti sia supersonico a valle della sezione di gola dell'ugello(ipotesi valida esclusi i transitori di accensione e spegnimento) ed imponendo una condizione di isoentropicità a tale flusso possiamo esprimere la velocità e la portata di uscita nel modo seguente:

$$\dot{m} = \rho^* A^* \sqrt{\gamma R T^*} = p_0 A^* \sqrt{\frac{\gamma}{RT_0}} \left(\frac{2}{\gamma+1}\right)^{\frac{\gamma+1}{2(\gamma-1)}}$$

$$u_e = \sqrt{\frac{2\gamma R}{\gamma-1} T_0 \left[1 - \left(\frac{p_e}{p_0}\right)^{\frac{\gamma-1}{\gamma}}\right]}$$

dove con $\gamma$ si intende il rapporto tra i calori specifici a pressione e volume costanti, con $R$ la costante specifica dei gas, con $p,\ \rho,\ T$ rispettivamente pressioni, densità e temperature, con

l’asterisco le grandezze in condizioni di sonicità, con il pedice “0” i valori totali delle rispettive grandezze a cui è posto il pedice stesso e con il pedice “e” ci si riferisce alla sezione di uscita.

Le parti che costituiscono tali tipi di motori sono principalmente il propellente solido, un case schermato da apposite protezioni termiche, un sistema di accensione e un ugello.

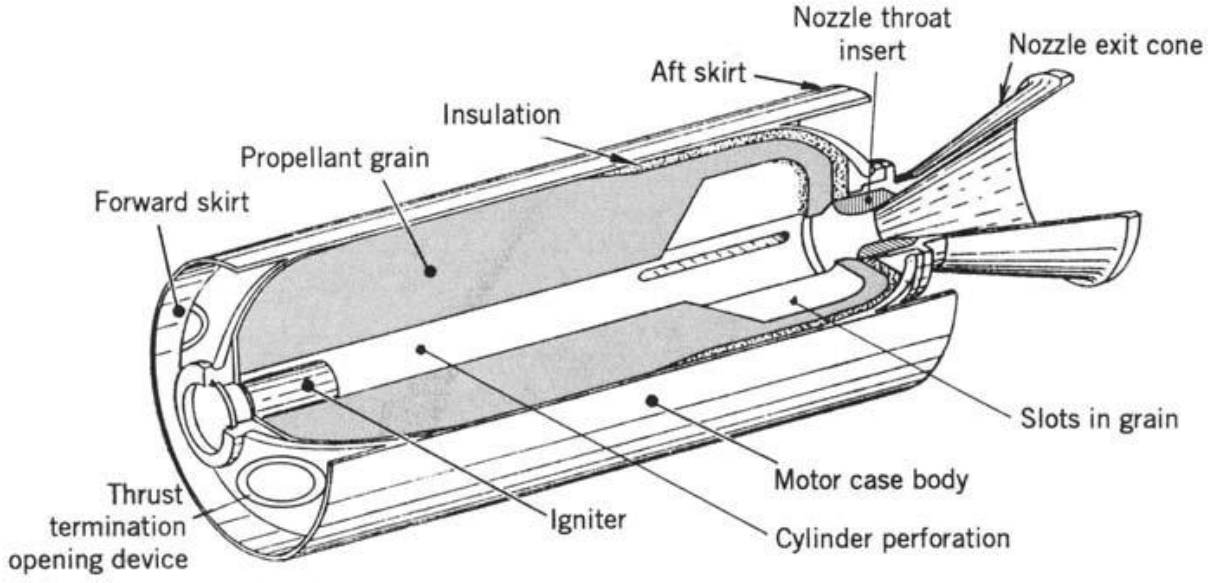



*Figura 1 - Schema base di un razzo a propellente solido.*

Il propellente solido, detto grano, è un composto di polveri (contenente sia il combustibile che il comburente così da determinare una completa combustione autoalimentata) pressate e tenute insieme da un legante quali per esempio alcuni tipi di gomme. Di seguito viene riportato un elenco dei materiali più utilizzati nella realizzazione dei grani di propellente solido.

| Functional category | Common examples | Comments |
|---|---|---|
| Solid oxidizers | Ammonium perchlorate (AP), other perchlorates, | AP is used in all but minimum smoke propellants. |
| | ammonium nitrate (AN), other nitrates, | AN is low cost but has numerous drawbacks, |
| | ammonium dinitramide (ADN), hydrazinium | including moisture sensitivity, little ballistic |
| | nitroformate (HNF) | tailorability, phase transitions, and aging concerns. ADN, HNF are still immature in U.S. |
| Energetic | Nitramines: Cyclotrimethylenetrinitramine (RDX), | Nitramines are used in most Class 1.1 and some |
| monopropellants | cyclotetramethylenetetramintramine (HMX), | Class 1.3 propellants. Provide increased 7sp. |
| | hexanitrohexaazaisowurtzitane (CL-20) | These three nitramines provide similar Isp in aluminized propellants. CL-20 is the densest and most oxygen-rich, but least mature. |
| Binders | Hydroxyl-terminated polybutadiene (HTPB), | HTPB, CTPB, PBAN are the most common Class 1.3 |
| | carboxyl-terminated polybutadiene (CTPB), | propellant binders. PEG, PPG, NC are generally |
| | polybutadiene acrylonitrile (PBAN), polyethylene | plasticized with nitrate esters. NC and GAP are |
| | glycol (PEG), polypropylene glycol (PPG), | energetic. |
| | nitrocellulose (NC), glycidyl azide polymer (GAP) | |
| Curatives | Isocyanates, epoxides | Isocyanates are used to cure hydroxyl-terminated polymers; epoxides cure PBAN, CTPB |
| Fuels | Beryllium, aluminum, magnesium | Beryllium gives the highest Isp, but is toxic. Aluminum has better Isp and density than magnesium but is not as easily ignited. |
| Plasticizers | Dioctyl adipate (DOA), dioctyl phthalate (DOP), | DOA, DOP, and similar esters are used with |
| | triacetin, other inert esters, nitroglycerin (NG), | nonpolar binders such as HTPB. Triacetin is used |
| | butanetriol trinitrate (BTTN), trimethylolethane | as a desensitizer for nitrate ester propellants. |
| | trinitrate (TMETN), other nitrate esters | Nitrate esters provide increased 7sp. NG is highest density and highest performance nitrate ester. Other nitrate esters are used to decrease detonability (compared with NG) or to give better low-temperature properties. |
| Stabilizers | A02246, p-rc-methyl nitroanaline (MNA), nitrodiphenylamine (NDPA) | A02246 prevents oxidative cross-linking of HTPB; MNA, and NDPA stabilize nitrate esters |
| Ballistic modifiers | Iron oxide, aluminum oxide, oxamide | Iron oxide; aluminum oxide accelerates burn rate; oxamide and other coolants slow burn rate. |

*Figura 2 - Materiali per propellenti solidi.*

Nel grano di propellente si possono avere diverse forme impresse sulla superficie. In base alla forma impressa ed alla sua evoluzione nel tempo si può definire una maggiore o minore superficie di combustione. Nelle figure seguenti vengono riportate alcune forme caratteristiche del grano ed una possibile evoluzione nel tempo di una geometria a stella.

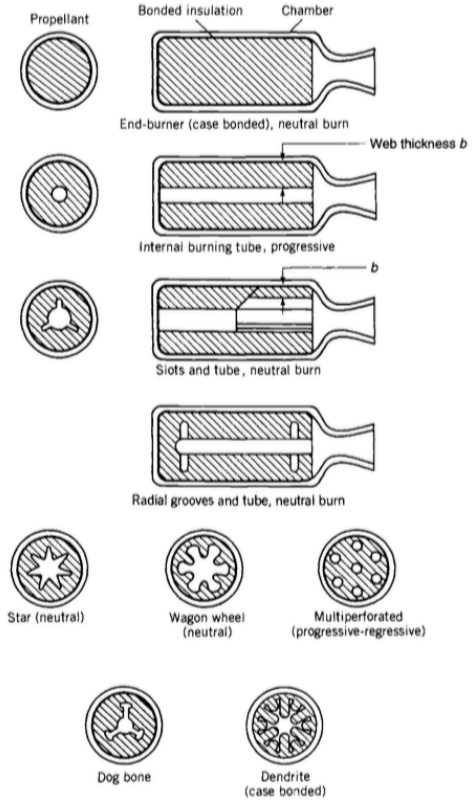



*Figura 3 - Sagome interne del grano di propellente.*

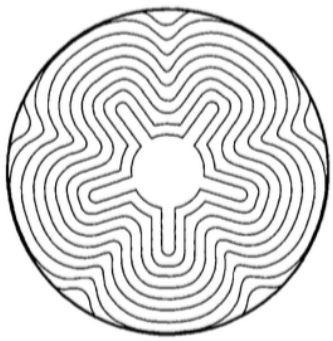


*Figura 4 - Evoluzione temporale di una geometria a stella.*

In base all'evoluzione della superficie in combustione, quindi alla geometria impressa sulla superficie del grano di propellente solido, si possono ottenere diversi tipi di spinta: una geometria con un'ampia area di combustione a pochi istanti dall'accensione determina un notevole incremento di spinta iniziale. Proprio grazie alle forme impresse sul grano quindi si riescono ad ottenere le spinte e le variazioni di spinta desiderate durante la combustione del propellente. I grani possono quindi essere classificati secondo la loro evoluzione temporale come riportato in figura.

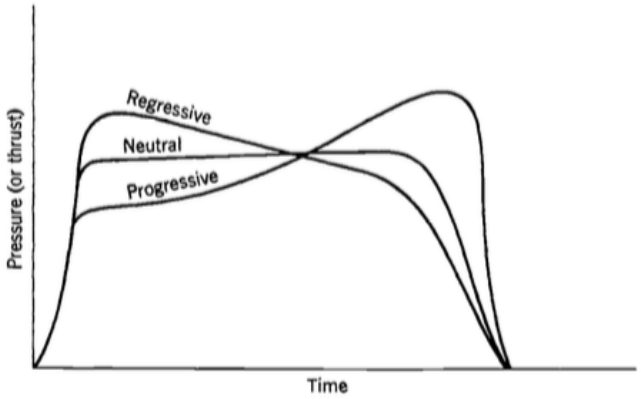



*Figura 5 - Classificazione dei grani secondo l'evoluzione temporale.*

L'impulso specifico dei razzi a propellente solido (massimo ca. 280-300s) è meno pregiato rispetto a quelli a propellente liquido, tuttavia vengono comunque utilizzati per motivi di spazio in quanto per ottenere la medesima spinta di un motore a solido con uno a liquido occorrerebbe molto più volume dove stoccare il carburante. Essendo di estrema importanza in campo aeronautico e aerospaziale contenere al minimo i volumi e quindi le superfici esposte per motivi di resistenza aerodinamica, i razzi a stato solido vengono utilizzati come booster di accelerazione per i primi istanti di volo, come ad esempio i "Solid rocket boosters" nello Space Shuttle. Esistono però alcuni casi di lanciatori, come ad esempio il vettore italo-europeo Vega, realizzati interamente con propellente solido. Il Vega ha appunto i primi tre stadi a stato solido e solo l'ultimo, il quarto, a liquido principalmente utilizzato per la movimentazione in orbita del *payload*.

Come già accennato, a causa dell'impossibilità di controllare la spinta di un razzo a propellente solido una volta innescata la combustione, è di fondamentale importanza fare, precedentemente alla realizzazione del razzo stesso, delle stime il più possibile accurate sulla spinta erogata, sulla pressione totale in camera di combustione, la durata della combustione del propellente ecc…

La fase di progettazione e sviluppo di una nuova motorizzazione è caratterizzata da un'ampia molteplicità di studi parametrici al fine di ottimizzarne la caratteristica propulsiva e valutarne le eventuali condizioni operative critiche. Le simulazioni balistiche quindi sono di fondamentale importanza non solo per una predizione il più accurata possibile delle prestazioni del futuro grano di propellente ma anche per ridurne il tempo di sviluppo, i test sperimentali necessari e quindi i costi ad esso connessi.

Per quanto riguarda le varie configurazioni di propellente si possono utilizzare approcci 1D o 2D nel caso di geometrie assial-simmetriche semplici o di dimensioni ridotte; nel caso in cui ci siano particolari configurazioni quali ad esempio *finocyl* (Figura 8) è necessario un approccio numerico di tipo 3D. Tale metodo è appunto quello utilizzato dal vecchio codice SPP che utilizza solidi semplici come sfere, prismi e cilindri per rappresentare geometrie più complesse.

Oltre ad essere versatili dal punto di vista delle geometrie delle varie motorizzazioni i codici di simulazione come ROBOOST, o precedentemente SPP, devono essere in grado di implementare e valutare non solo condizioni operative nominali come descritte dalla teoria, ma anche quelle non ideali andando a riprodurre in maniera il più possibile esatta tutti i fenomeni fisici e le instabilità che si potrebbero verificare nei *firing-tests* reali. Allo stesso tempo però questi codici devono garantire tempi computazionali accettabili in maniera tale che, grazie ai risultati ottenuti, si possano effettuare modifiche al progetto della motorizzazione per analizzarne nuovamente gli effetti sul profilo di missione.

Tale necessità deriva dal fatto che come brevemente accennato nell'introduzione durante la produzione di razzi a stato solido di grandi dimensioni spesso il grano di propellente viene

suddiviso in più parti colate separatamente ed unite insieme in un secondo momento. La leggera diversità di composizione chimica e di grandezza particellare delle diverse colate, quindi dei singoli segmenti costituenti l'intero grano del propellente, comporta un diverso comportamento dello stesso durante la combustione e quindi differenti prestazioni.
Inoltre durante la fase di colaggio all'interno del grano di propellente possono formarsi piccole sacche d'aria o generarsi dei difetti quali l'occlusione di particelle estranee al propellente. In entrambe i casi si avrà una variazione della regressione della superficie combusta del propellente rispetto a quella che si avrebbe con una superficie liscia.
Si nota subito l'importanza nel tenere conto di tali "anomalie" all'interno del grano di propellente durante una simulazione balistica in quanto una variazione della regressione della superficie comporta un diverso quantitativo di gas prodotti quindi una variazione nella spinta nominale.

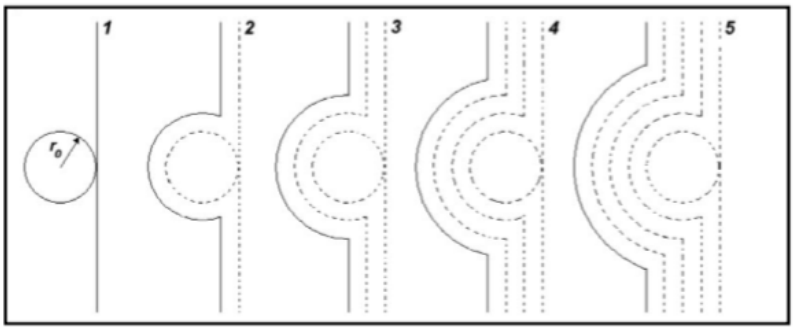



*Figura 6 - Regressione superficiale in presenza di cavità.*

Altra situazione che porterebbe ad un allontanamento dalle condizioni nominali di funzionamento potrebbe essere dovuta ad una disomogeneità degli ingredienti in una zona del grano. Ciò porterebbe ad una variazione della velocità e della direzione di combustione in quel punto ovvero ad una anisotropia del propellente.

Anche in questo caso la superficie in regressione seguirebbe un andamento diverso da quello nominale di progetto alterando le prestazioni balistiche del razzo.

Per questi motivi vengono sviluppati codici balistici il più possibile robusti e con la possibilità di tener conto di un numero sempre crescente di parametri di input che possano descrivere in maniera il più accurata possibile il reale comportamento del grano di propellente durante la combustione per poter effettuare delle simulazioni che riproducano istante per istante la combustione di un razzo a stato solido fornendo come output i principali valori prestazionali.

Vengono effettuate inoltre delle prove in scala ridotta per testare il tipo di propellente del nuovo motore che si vuole realizzare. Il campione in scala ridotta non è sagomato internamente con la stessa geometria del motore di dimensioni normali ma serve unicamente ad estrarre due parametri balistici *a* ed *n*, dipendenti dalla tipologia del propellente e dalle sue caratteristiche reologiche, da inserire all'interno della formula di *Vieille*.

Una volta ricavati tali coefficienti riguardanti la tipologia di propellente utilizzato vengono inclusi negli input della simulazione balistica al calcolatore insieme ad altri parametri che descrivono le caratteristiche del grano sopracitate.
I risultati ottenuti moltiplicati per un apposito fattore di scala (o *Scale Factor – SF*) dovrebbero essere il più possibile simili ai risultati ottenuti nel test con le motorizzazioni in scala 1:1.

# CODICE SPP

Il codice *SPP* acronimo di *Solid Propellant Rocket Motor Performance Computer Program*, rilasciato per la prima volta nel 1975 dalla NASA e divenuto il software di riferimento in tutti gli Stati uniti per la previsione delle prestazioni dei razzi a propellente solido, è un codice di calcolo che, partendo dalla geometria iniziale del grano fornisce i valori di portata massica, di pressione, di spinta, e di impulso in funzione del tempo. L'obiettivo del codice *SPP* è inoltre quello di prevedere per un'assegnata geometria, l'impulso specifico con un errore massimo del +0.5%, la spinta e impulso totale entro il + 3% .

La richiesta di Avio dell'implementazione all'interno del codice ROBOOST di un'interfaccia grafica per il caricamento delle geometrie in questo formato, deriva dal fatto che il software SPP fu venduto ad alcune aziende operanti nel campo della propulsione spaziale le quali tutt'ora posseggono alcune motorizzazioni, ancora oggi utilizzate, descritte in formato SPP.

Il software SPP era composto da una struttura modulare; ogni modulo (ad eccezione di quello di controllo) era basato su un programma pre-esistente. I seguenti programmi erano alla base del software SPP:

- One Dimensional Equilibrium Program (ODE)
- One Dimensional Kinetics Program (ODK)
- Two Dimensional Two Phase Program (TD2P)
- Turbulent Boundary Layer Program (TBL)
- Three Dimensional Grain Design Program
- Motor Ballistics Program

Ognuno dei primi quattro programmi sopra elencati costituiva un modulo a sé stante nel software SPP ad eccezione degli ultimi due che vennero uniti in un unico modulo. Questi programmi vennero aggiunti ad un ulteriore modulo di controllo che ne permetteva l'esecuzione simultanea. Nella figura seguente vengono rappresentati in maniera schematica i sei moduli che componevano il software SPP e per ognuno vengono elencate le principali funzioni.

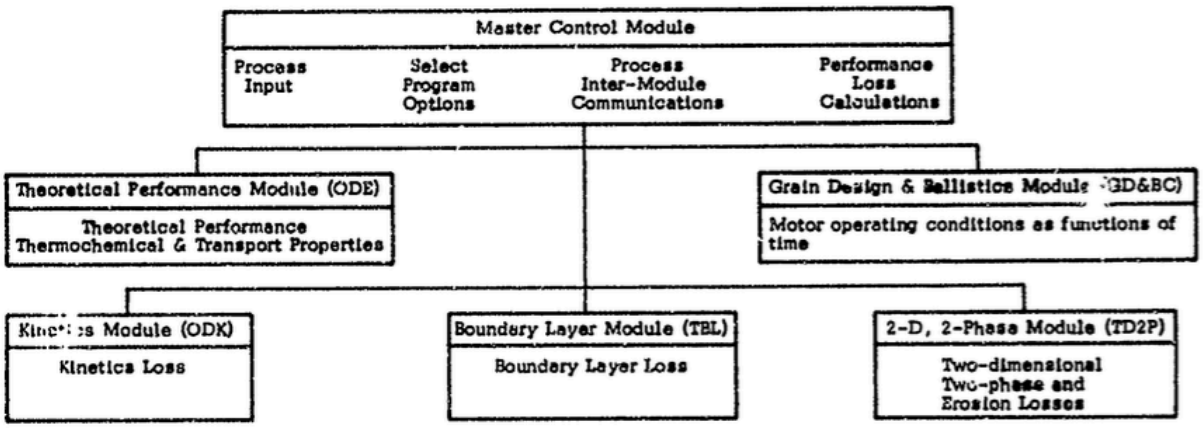



*Figura 7 - Schema base SPP.*

Tutti i programmi già esistenti che furono utilizzati per la creazione del codice SPP vennero inoltre leggermente modificati in maniera tale che il loro funzionamento si adattasse il più possibile allo scopo principale del codice.
Il software SPP fu progettato in maniera tale da fornire all'utente la flessibilità nella scelta dei moduli da eseguire in una qualsiasi sessione di calcolo delle prestazioni.

Il modulo di controllo principale permetteva di eseguire tutti i moduli, o gli interi programmi costituenti il codice, in maniera sequenziale in una sola sessione anche se tale procedimento molto spesso non garantiva la massima efficienza computazionale. Permetteva inoltre di scegliere tra i moduli riguardanti la stima delle prestazioni quali e in che ordine dovessero essere eseguiti.

Una simulazione completa delle prestazioni del motore poteva essere ottenuta in maniera efficiente, dal punto di vista del tempo di esecuzione del codice, se tutti i moduli utilizzati venivano chiamati in un'unica esecuzione. Ciò era reso possibile grazie alla struttura a gruppi con cui era stato progettato il software SPP.

Tutti i dati che dovevano essere scambiati tra i vari moduli venivano scritti in un'unità logica esterna oltre ad essere trasferiti internamente in maniera automatica.
A fine sessione questi dati tra loro collegati potevano essere salvati in altri modi differenti ed erano inoltre disponibili per successive sessioni in modo tale da non dover rieseguire nuovamente moduli già utilizzati.
Questo tipo di "connessione" tra i dati di output aveva due principali vantaggi: il primo in fatto di risparmio di tempo e costo computazionale, il secondo costituito dalla possibilità di "ingannare" il codice facendo ricevere come input salvataggi di altre sessioni o di altri programmi balistici compatibili con la struttura ed il formato dei file di output del codice SPP.

In conclusione il software SPP può essere descritto come un codice modulare e flessibile che fornisce predizioni accurate delle prestazioni di un razzo a propellente solido. Le tecniche analitiche e le correlazioni che venivano utilizzate sono descritte nella figura seguente.

SUMMARY OF SPP PROGRAM CAPABILITIES

| ITEM | ANALYTICAL COMPUTATION | CORRELATION | INPUT |
|---|---|---|---|
| Theoretical Performance | ODE | | X |
| Kinetics Loss | ODK | X | X |
| Two Dimensional Loss | TD2P | X | X |
| Two Phase Loss | TD2P | X | X |
| Erosion Loss | TD2P | | X |
| Boundary Layer Loss | TBL | X | X |
| Nozzle Erosion | | X | X |
| Grain Geometry & Ballistics | BAL | | |
| Submergence Loss | | X | X |
| Combustion Efficiency | Modification of Correlation | X | X |
| Particle Size | | X | TD2P(only) |
| Burning Rate | BAL (motor) | | (Strand) |
| Insulation Dilution | | (BAL only) | |

*Figura 8 - Sommario potenzialità SPP.*

Il codice SPP nel corso degli anni, con l'aumentare della potenza dei calcolatori è stato successivamente sviluppato ed ampliato rendendo disponibili più opzioni per la simulazione, migliorando la velocità di esecuzione e la precisione del calcolo delle prestazioni pur mantenendo in generale la stessa struttura modulare.

Ad esempio la versione del codice *SPP97*, può essere suddivisa in tre sezioni di analisi distinte:

1. Prestazione ugelli.
2. Progettazione del grano e balistica.
3. Analisi delle instabilità.

La progettazione del grano può ulteriormente essere suddivisa in tre moduli in base al tipo di motore che viene rappresentato:

1. Bidimensionale
2. Assialsimmetrico
3. Completamente tridimensionale (Three-Dimesional Grain Design Module – GDM)

Le operazioni geometriche eseguite dal modulo GDM erano basate sul calcolo di volumi e delle loro variazioni nel tempo.

Il codice *SPP* riceveva in input un file (in formato *.txt*) nel quale veniva descritto il motore sotto forma di geometrie base (o solidi semplici) come cilindri, coni e prismi, i quali uniti e fusi insieme forniscono la forma completa del motore.

Inoltre all'interno delle descrizioni dei motori in formato *SPP* si può trovare una parte riguardante il *case*, ovvero il rivestimento esterno, del motore.
Per alcuni tipi di motori, quali ad esempio lo Z9, sono descritte anche geometrie particolari quali i FINOCYL (FIN On CYLinder) ovvero particolari configurazioni tridimensionali del grano di propellente.

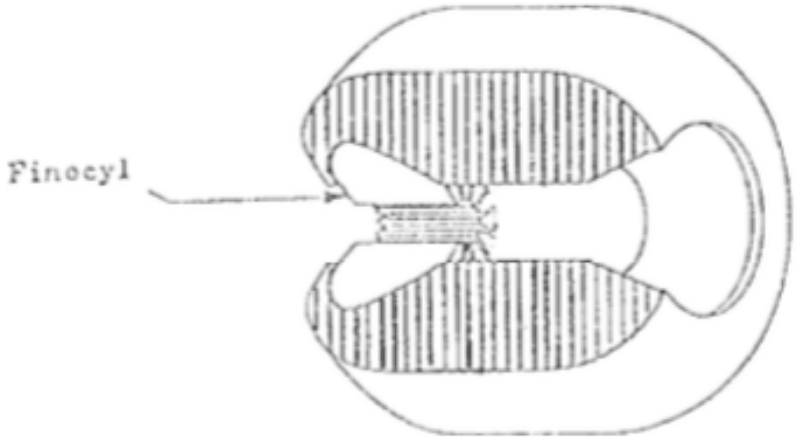



*Figura 9 - FINOCYL: FIN On CYLinder.*

L'interfaccia grafica, sviluppata per Avio Spa durante questo lavoro di tesi, permette che tutte queste informazioni, scritte nelle geometrie in formato SPP e riguardanti l'intera struttura del motore, vengano lette tramite il codice sviluppato in MATLAB® e che successivamente vengano scritte in un nuovo file contenente la stessa geometria in un formato compatibile con il CaD, insieme alle istruzioni che permettono a *SALOME* di ricostruire in maniera corretta il grano di propellente al fine di renderlo disponibile al modulo di mesh.

Nel capitolo successivo dopo una breve introduzione al CaD SALOME viene descritto tramite uno schema a blocchi il funzionamento di tale codice al fine di comprendere in maniera più chiara ed esaustiva la procedura sopra accennata.

# CAD SALOME

*SALOME* è un software open-source che fornisce una generica piattaforma per simulazioni numeriche con tool di Pre- e Post-elaborazione. Si basa su un'architettura semplice e flessibile formata da componenti riutilizzabili.

*SALOME* si basa sulla tecnica di modellazione solida CSG - *Constructive Solid Geometry*. Questa tecnica, precedentemente chiamata *Computational Binary Solid Geometry,* permette di creare superfici o solidi complessi utilizzando operatori Booleani partendo da geometrie più semplici.
In figura viene visualizzato un esempio del funzionamento della tecnica CSG.

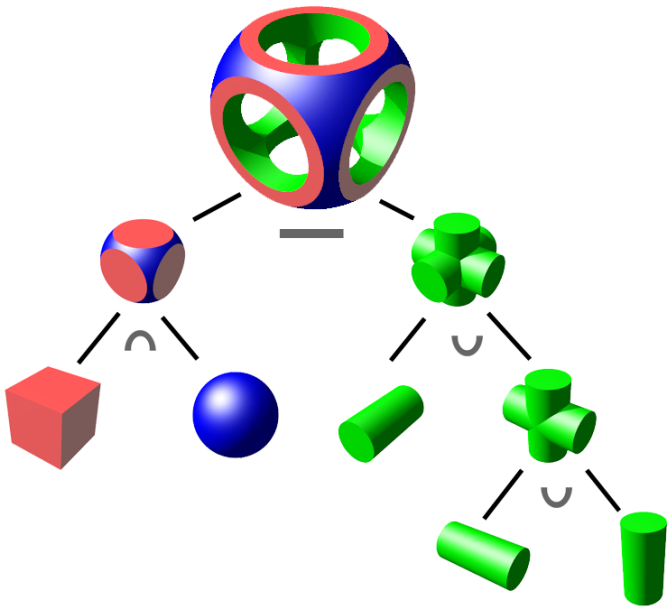


*Figura 10 - Constructive Solid Geometry.*

Questa tecnica è utilizzata nel processo di realizzazione delle geometrie interne del grano di propellente in quanto, essendo descritte da forme semplici come coni e cilindri nel vecchio formato SPP, il CaD *SALOME* deve ricostruire la geometria completa "assemblando" solidi semplici.

*SALOME* può essere utilizzato come un software autonomo per la generazione di modelli CaD, la loro preparazione al calcolo numerico e alla post-elaborazione dei risultati ottenuti, ma può anche essere utilizzato come una piattaforma integrativa per codici numerici esterni al fine di realizzare un più ampio strumento per la gestione di modeli CaD.
In questo lavoro di tesi infatti le geometrie del grano di propellente non vengono realizzate direttamente dal CaD *SALOME* per via grafica ma secondo le istruzioni riportate nel file "tradotto" dal codice SPP.

Viene quindi riportato in *Figura 11* uno schema base riassuntivo del funzionamento del CaD all'interno del processo di costruzione delle geometrie ed in *Figura 12* lo stesso schema in maniera più dettagliata evidenziando le funzioni chiamate e le loro finalità.

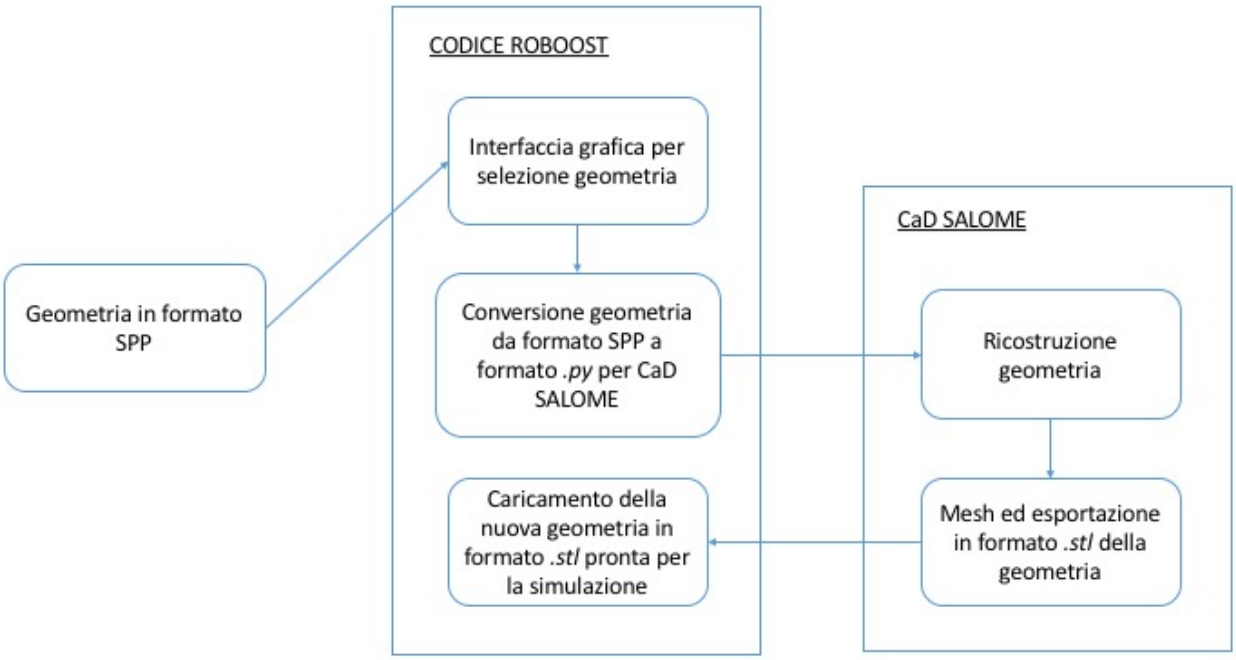



*Figura 11 - Integrazione SALOME - ROBOOST.*

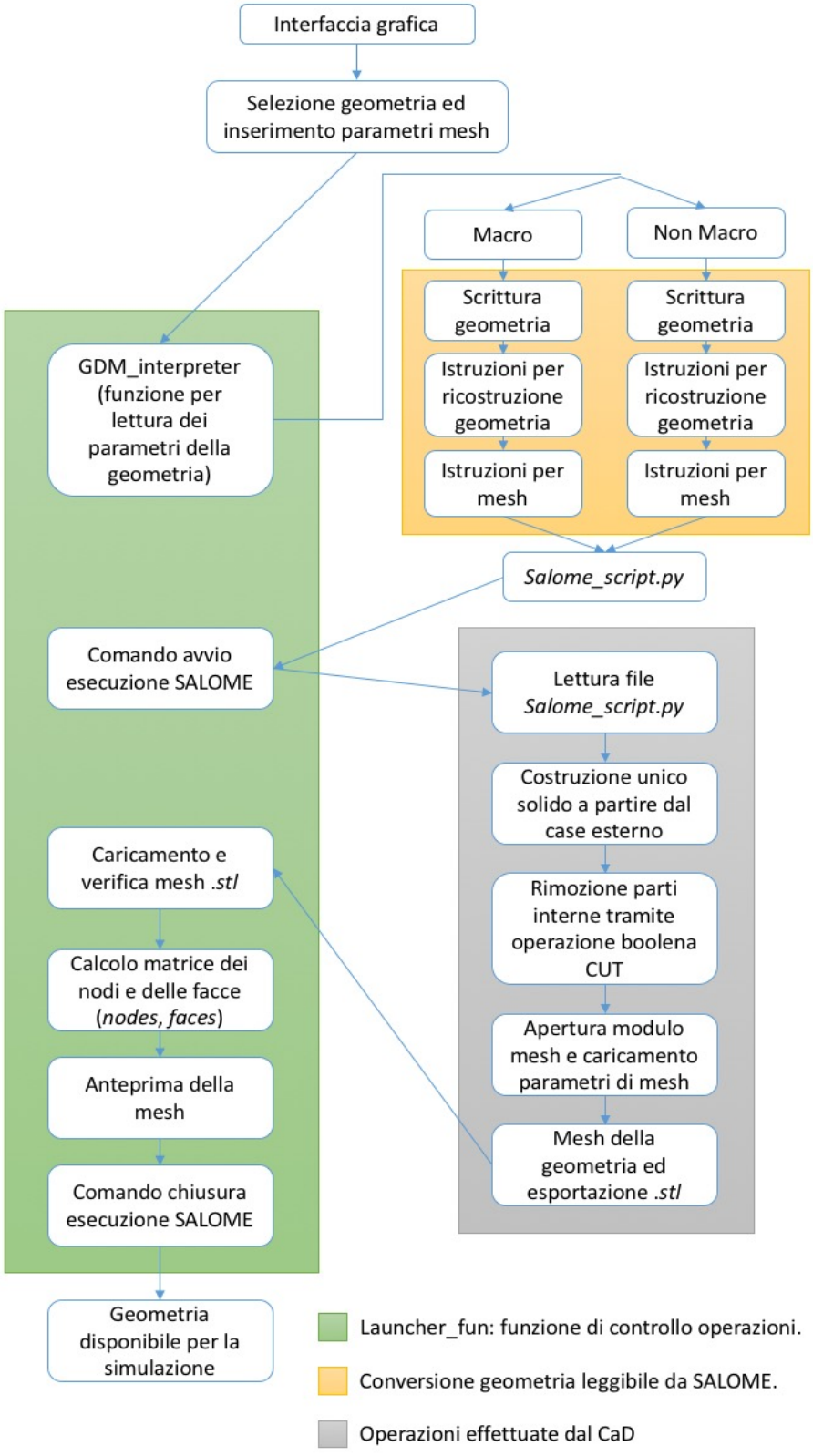



*Figura 12 - Schema dettagliato integrazione SALOME - ROBOOST.*

# CODICE ROBOOST

## Generalità

Il codice ROBOOST (ROcket BOOst Simulation Tool) costituisce un algoritmo numerico, sviluppato in ambiente MATLAB®, su piattaforma Windows®, per la simulazione del processo di combustione e regressione della superficie del propellente solido mediante approccio geometrico 3D.

Il processo di regressione superficiale ed esaurimento virtuale del volume di propellente avviene sfruttando l'evoluzione nel tempo di un dominio discreto triangolare 2D (mesh triangolare) mediante tecnica di *off-setting* applicata ai singoli nodi della griglia.
Mediante tale approccio risulta quindi possibile valutare evoluzioni della superficie in combustione totalmente non-omogenee e non-isotrope.

Un modello di balistica interna 0D non-stazionario provvede alla stima delle principali grandezze termodinamiche in camera e nella valutazione del rateo di combustione del propellente mediante tradizionale *formula di Vieille*.

$$r_b = SF \cdot HUMP \cdot a \cdot p_0^n$$

Nell'equazione $r_b$ rappresenta il rateo di combusitone, $SF$ (scaling factor) è un coefficiente correttivo che tiene conto del fatto che vengono effettuate prove sperimentali su motori di differenti dimensioni, $HUMP$ è un fattore che considera le anisotropie del propellente, $p_0$ è la pressione totale in camera di combustione e $a$ e $n$ sono coefficienti balistici tipici della relazione di Vieille.
In figura viene schematicamente riportato il layout complessivo e funzionale del codice.

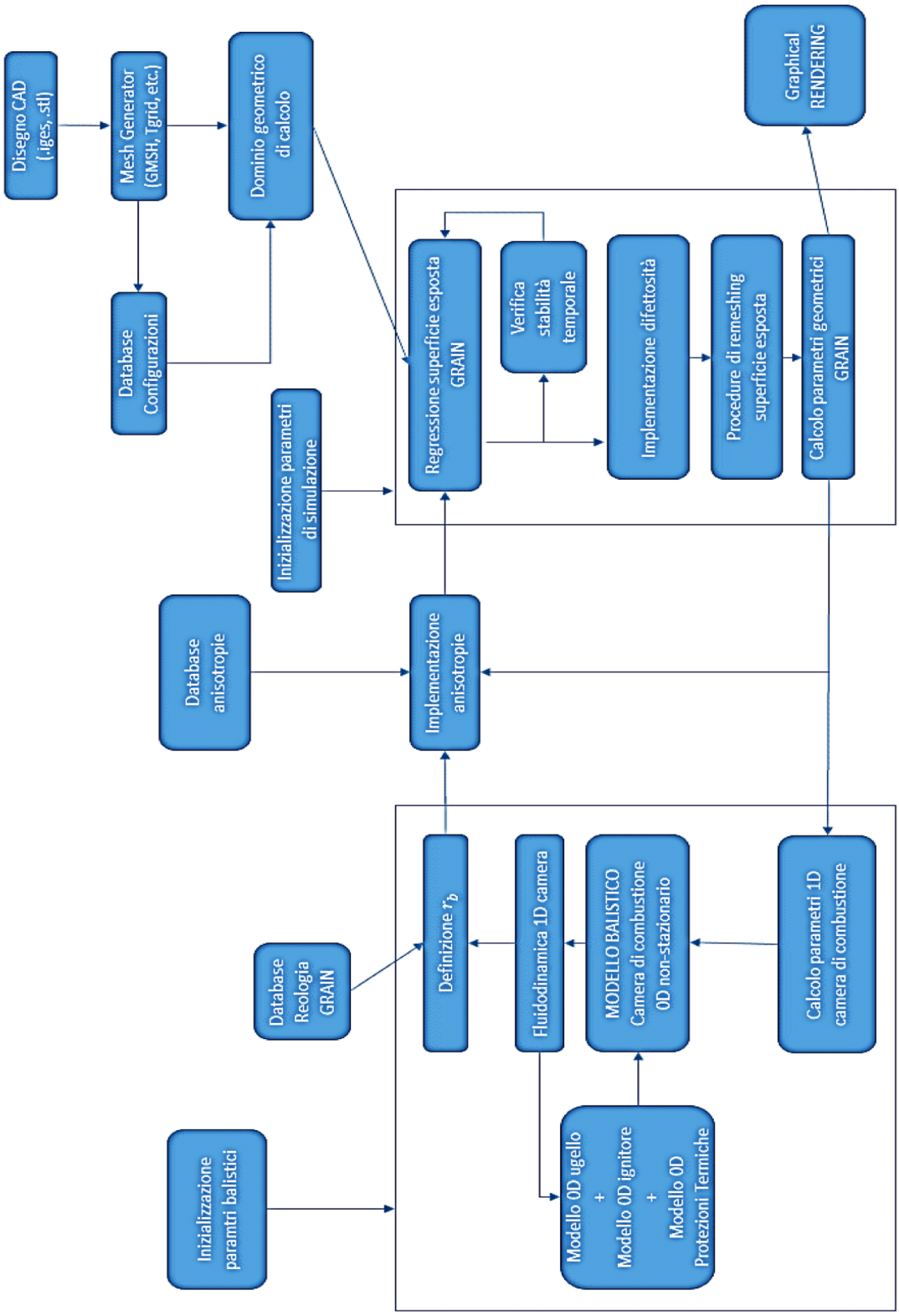



*Figura 13 - Layout generale del codice ROBOOST.*

Specifici modelli 0D relativi alle dinamiche di ignitore (IGN), ugello (NOZ) e protezioni termiche (TP) vengono a scelta implementati nel calcolo.

Il modulo di balistica interna può essere a scelta disattivato per ottenere la sola valutazione dei profili temporali di superficie e volume (singoli segmenti booster, etc.).

Altri input del codice possono essere una mappa 3D dei vettori non isotropi di combustione (mappa anisotropie), una descrizione delle difettosità interne del grano di propellente, un file per le ablazioni delle protezioni termiche.
L'utente può inoltre scegliere, tramite selezione grafica dal pannello principale del codice, quale tipo di algoritmo utilizzare per il calcolo della regressione della superficie in combustione.

I principali output del codice una volta terminata la simulazione sono:

- un sommario delle evoluzioni delle grandezze termodinamiche e prestazionali del motore in ciascuna iterazione del motore

| ***Nome parametro*** | ***Variabile*** | ***Unità*** |
|---|---|---|
| Tempo trascorso | $t$ | *[s]* |
| Spessore medio bruciato | $web$ | *[mm]* |
| Superficie in combustione | $Sb$ | *[m^2]* |
| Volume grain residuo | $Vg$ | *[m^3]* |
| Massa grain residua | $Mg$ | *[kg]* |
| Pressione totale media | $p0$ | *[bar]* |
| Temperatura totale media | $T0$ | *[K]* |
| Volume camera di combustione | $Vcc$ | *[m^3]* |
| Calore specifico a pressione costante | $cp$ | *[J/(kg*K)]* |

| | | |
|---|---|---|
| Rapporto cp/cv | $\gamma$ | *[-]* |
| Superficie protezioni termiche esposte | $Spt$ | *[m^2]* |
| Diametro ugello | $Dnoz$ | *[mm]* |
| Spinta erogata | $F$ | *[kN]* |
| Velocità caratteristica | $C*$ | *[m/s]* |
| Impulso specifico | $Isp$ | *[s]* |

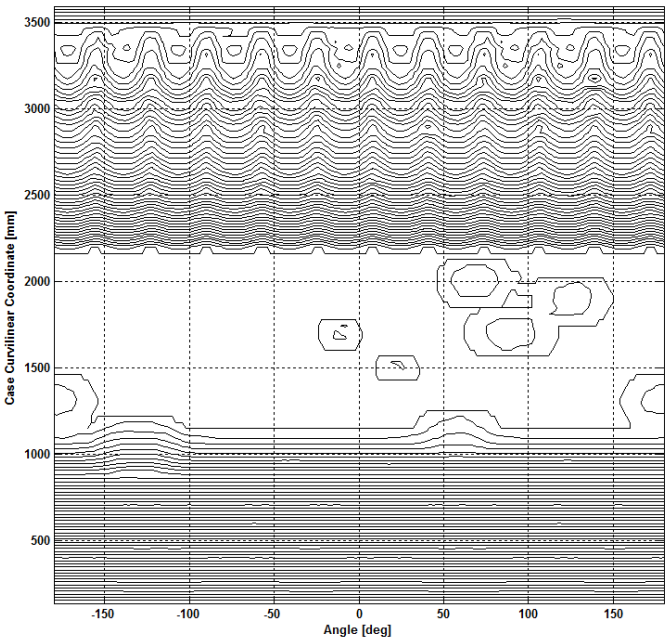
*Tabella 1 - Output del codice ROBOOST(1).*

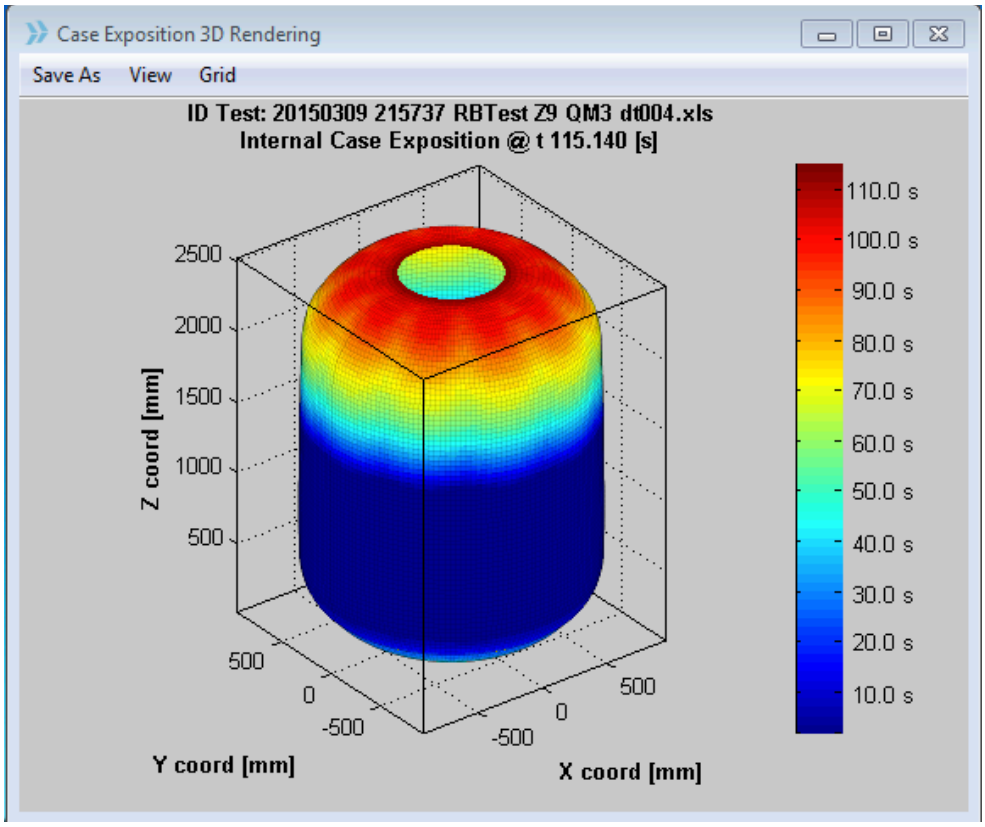
- un resoconto dei parametri iniziali di *setting* della simulazione, l'evoluzione del dominio discreto nelle varie iterazioni e le tempistiche di calcolo richieste

| ***Nome parametro*** | ***Variabile*** | ***Unità*** |
|---|---|---|
| # Iterazione | #iter | *[-]* |
| Elapsed Time | elaptime | *[s]* |
| Matrice triangolazioni | #faces | *[-]* |
| Matrice nodi mesh | #nodes | *[-]* |

*Tabella 2 - Output del codice ROBOOST(2).*

- la mappa 2D della superficie interna del *case*, con le tempistiche di esposizione





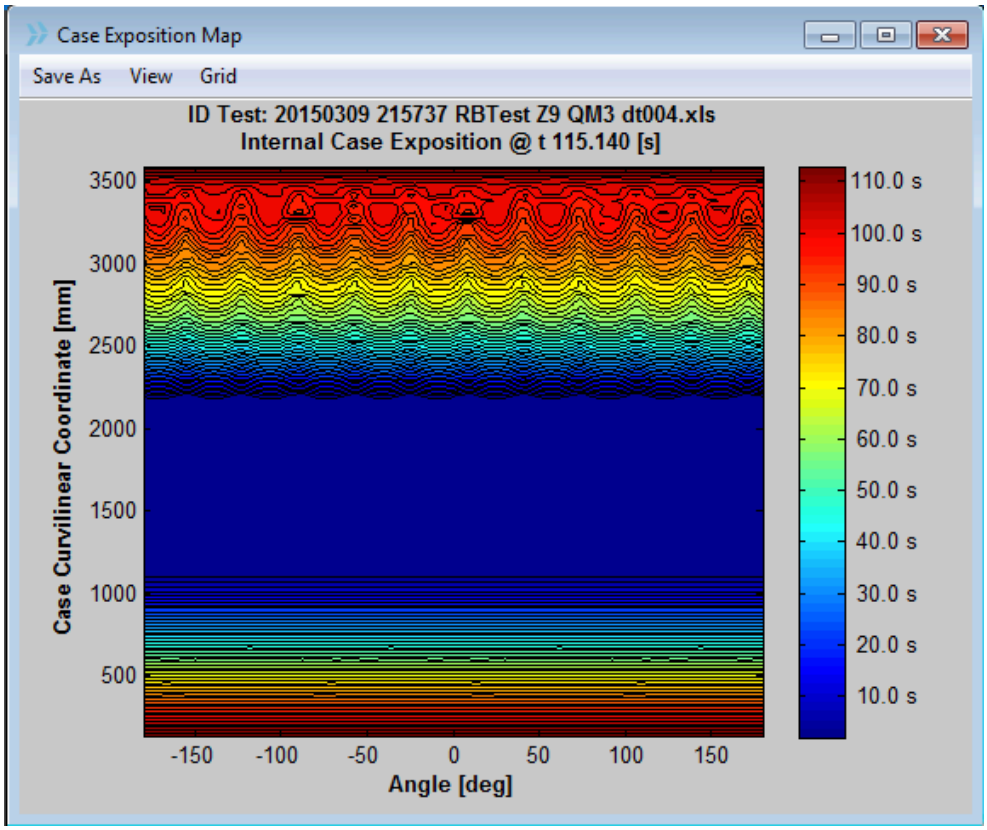




*Figura 14Figura 14 - Output del codice ROOBOST(3).*

- i "*frames*" dell'evoluzione temporale della regressione interna del grano di propellente come mostrato nelle immagini seguenti. Se l'utente lo desiderasse possono essere mostrati in modalità *slideshow*.

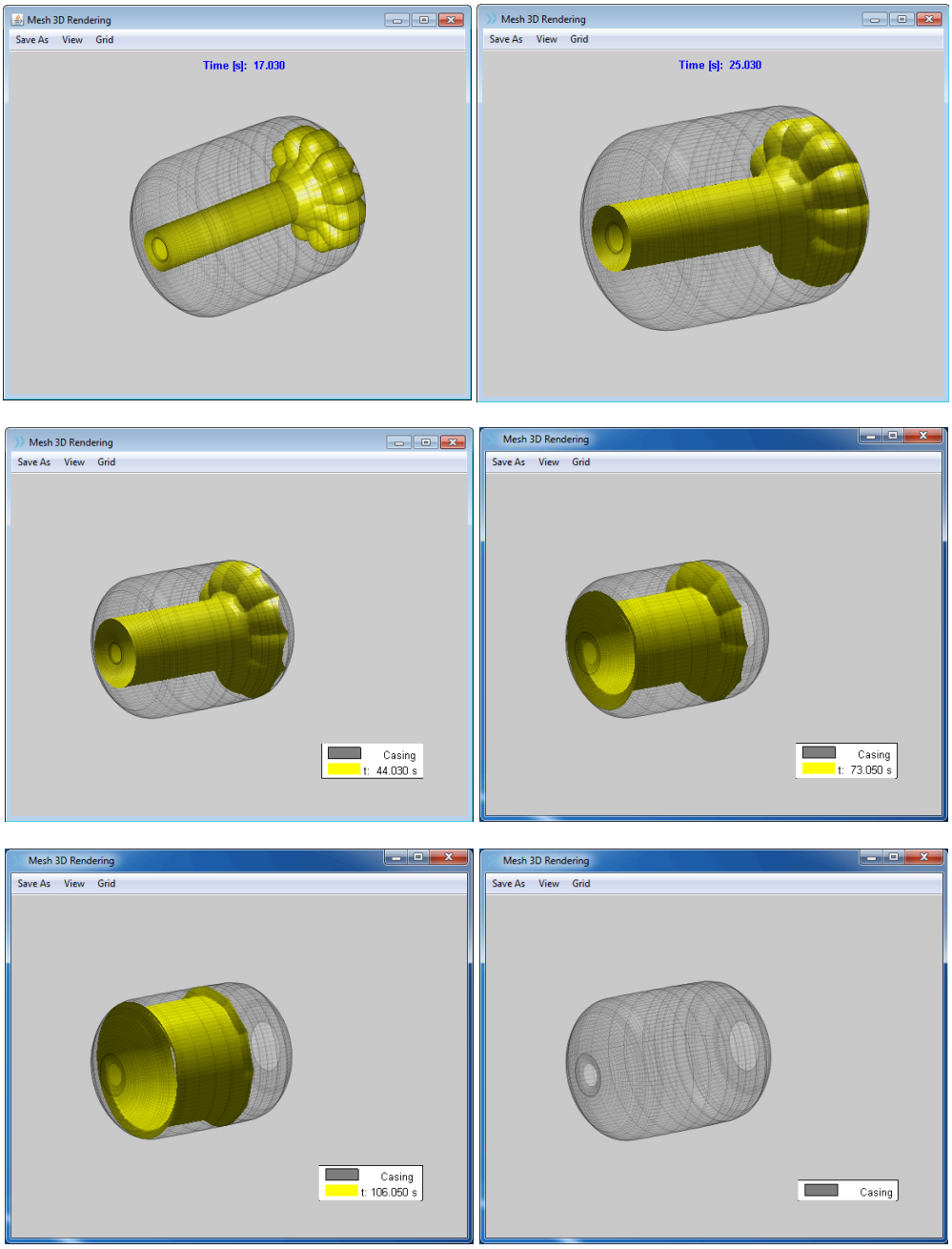



*Figura 15 - Evoluzione temporale grano in regressione.*

Il modulo grafico SPP realizzato durante il lavoro di tesi costituisce un ampliamento dell'attuale pannello per il caricamento delle geometrie in formato IGES e XLS.

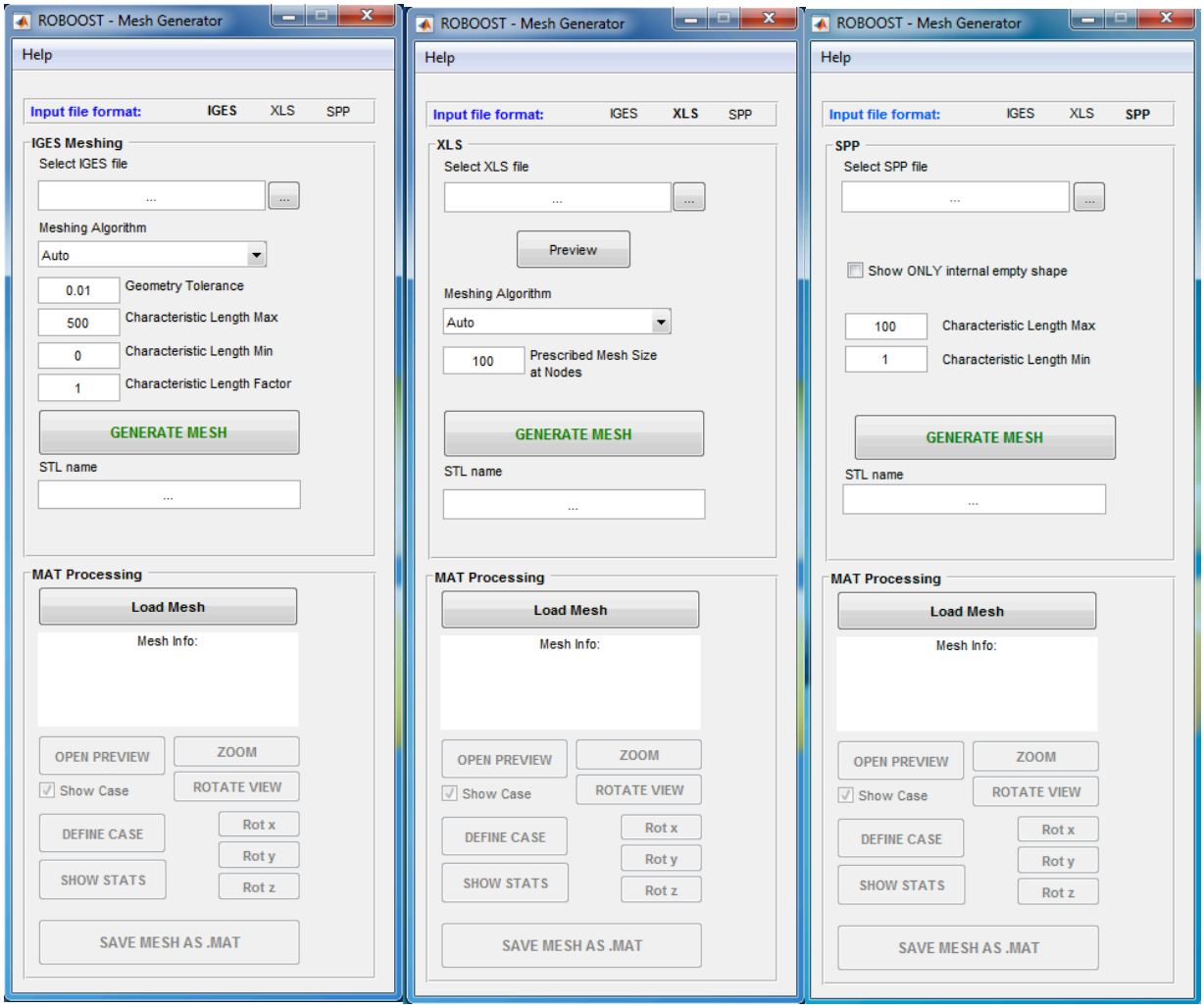



*Figura 16 - Nuovo pannello ROBOOST mesh generator con interfaccia grafica per geometrie SPP.*

Tramite la nuova GUI (*Grafic User Interface*) è possibile selezionare come input per ROBOOST il file contenente le informazioni riguardanti la geometria da costruire ed una volta caricato è possibile decidere la lunghezza caratteristica, massima (*Characteristic Lenght Max*) e minima (*Characteristic Lenght Min)*, dei triangoli che andranno a formare la mesh 3D ottenuta tramite l'algoritmo NETGEN 1D-2D.

È inoltre possibile decidere se ricostruire l'intero grano con il *case* oppure solamente la superficie della cavità interna tramite la spunta nel box *Show ONLY internal empty shape*.
Tale opzione in realtà non permette di proseguire con la simulazione in quanto non è presente la descrizione del *case* ma vuole solamente essere una verifica della corretta realizzazione della geometria interna del grano.

Premendo il pulsante verde GENERATE MESH si avvia il processo di conversione formato, costruzione e mesh della geometria del propellente.

Il pannello sottostante, MAT Processing, si attiva automaticamente una volta caricata la mesh appena generata e rende disponibili all'utente alcuni tool di post-elaborazione.
È infatti possibile effettuare uno zoom della mesh, ruotarla secondo i tre assi cartesiani, visualizzare o nascondere il *case* e mostrare le statistiche riguardo le dimensioni dei triangoli della mesh. Il pulsante DEFINE CASE permette all'utente di definire tramite selezione da mouse il case del motore nel caso in cui non fosse previsto all'interno della descrizione della geometria stessa.

L'operazione di salvataggio della mesh in formato .mat salva la geometria appena realizzata sopprimendo nella mesh tutti i triangoli degeneri ovvero tutti i triangoli sotto una determinata dimensione specifica in base alla grandezza caratteristica dell'intero motore.
Tale operazione può richiedere alcune ore: in una generica mesh dello Z9 con lunghezza caratteristica massima e minima rispettivamente di 100 e 1 possono trovarsi dai ventimila ai quarantamila triangoli degeneri.

Una volta selezionato il file della geometria da caricare, tramite il pannello “Mesh Generator” di ROBOOST, la lettura delle geometrie viene effettuata da una specifica funzione del codice, denominata “Interprete”, che grazie al riconoscimento dei caratteri di inizio ($IN) e fine ($END) sezione presenti all’interno dei file in formato SPP, importa e suddivide le informazioni ivi contenute, in variabili e strutture memorizzate nel Workspace di Matlab.
Si riporta di seguito un esempio di una descrizione di un piccolo motore in formato SPP.

```
$GDM
XPB = 0.0000000E+00, 0.7200000 , 1.720000 , 2.720000 ,
3.720000 , 4.720000 , 5.720000 , 6.720000 , 7.720000 ,
8.720000 , 9.720000 , 10.72000 , 11.72000 , 12.72000 ,
13.72000 , 14.72000 , 15.72000 , 16.72000 , 17.72000 ,
25.72000 , 31.72000 , 32.72000 , 32.82000 , 33.72000 ,
34.72000 , 35.72000 , 36.72000 , 37.72000 , 38.72000 ,
39.72000 , 40.72000 , 41.72000 , 42.72000 , 43.72000 ,
44.72000 , 45.72000 ,
RPB = 2.400000 , 6.120000 , 8.440000 , 10.13000 ,
11.52000 , 12.68000 , 13.66000 , 14.40000 , 15.24000 ,
15.86000 , 16.38000 , 16.86000 , 17.22000 , 17.54000 ,
17.80000 , 18.00000 , 18.18000 , 18.26000 , 3*18.32000 ,
18.20000 , 18.29000 , 18.20000 , 18.06000 , 17.76000 ,
17.48000 , 17.20000 , 16.80000 , 16.36000 , 15.84000 ,
15.22000 , 14.46000 , 13.64000 , 12.64000 , 11.54000 ,
NXR = 36,
BNDPLT = 1, CRSPLT = 2, LNGPLT = 1, GPLT(1) = 5*1,
NWEBS=2, WEBS=2.5,8, NXSTA=5, XSTA=10,15,25,30,40,
COUPLD=0, SYMFAC=16, NPRINT = 6, 9-3
X(1) = 0,5,16,32,45.72, NNX = 5, NX(1) = 10,120,40,120,
Y(1) = 0,20, NNY = 2, NY(1) = 120,
B(1) = 0,12., NB=24, NNB=2,
XSLOT(1) = 45.72, DXSLOT(1) = .04, IGAF(1) = 1, NSLOTS = 1, NDIV = 3,
$END
SSTAR
$IN XA=17.92, RTA=1.77233, RVA=7.2,
XB=45.72, RTB=1.77233, RVB=7.2,
ETA=27.1, R1=.25,
MIRROR=T, $END
CYLINDER CENTRAL PORT
$IN C1=0,0,0, C2=46,0,0, R=1.88, $END
CYLINDER FOR SUBMERGED NOZZLE
$IN C1=34.22,0,0, C2=46,0,0, R=4.55, CR=.75, $END
CONE FOR SUBMERGED NOZZLE
$IN C1=45.72,0,0, C2=39.4,0,0, R1=7.3, R2=4.55, $END
CYLINDER AFT VOLUME
$IN C1=45.7,0,0, C2=46,0,0, R=11.562, $END
FSTAR
$IN XA=17.92, RTA=1.77233, RVA=7.2,
ETA=27.1, R1=.25, R2=0., RDOME=7.2,
MIRROR=T, $END
END FIGURES
```

*Figura 17 - Geometria Delta SRM in formato SPP.*

Nella prima parte della geometria le coordinate XPB ed RPB sono riferite al *case* della motorizzazione, mentre le indicazioni STAR ed FSTAR si riferiscono alle geometrie a stella, da cui il loro nome, che devono essere fuse insieme ai componenti centrali scomposti in cilindri (CYLINDER) e coni (CONI). Per alcune motorizzazioni possono trovarsi le indicazioni per la realizzazione dei FIN, di cui si è precedentemente parlato, i quali sono presenti intorno all'asse centrale del motore per un numero di volte pari alla metà del parametro SYMFAC.

Acquisite tutte le specifiche della geometria selezionata, un'altra funzione procede alla scrittura di un nuovo file in formato *.py* il quale altro non è che una "traduzione" della geometria dal formato SPP ad un formato leggibile dal CaD *SALOME*.
La funzione sopra descritta viene in realtà selezionata in relazione al tipo di geometria: in base a come vengono forniti i parametri che descrivono il grano del propellente, all'interno delle descrizioni in formato SPP, possiamo trovare delle geometrie che prendono il nome Macro.

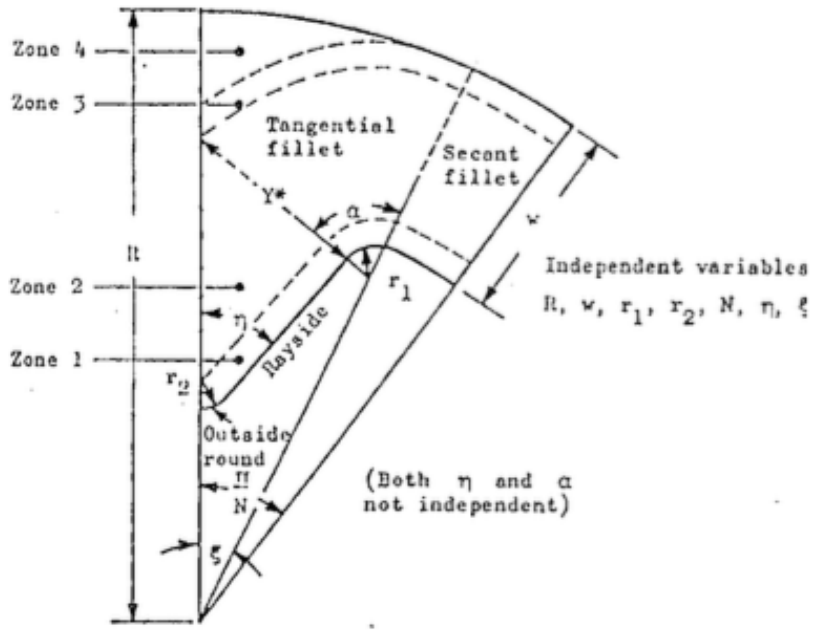



*Figura 18- Tapered Star Design.*

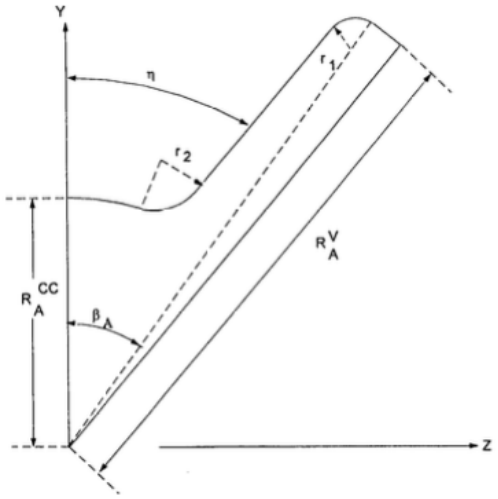



*Figura 19 - Alternate Star MACRO input.*

Una volta che l' "Interprete" ha riconosciuto il tipo di geometria in base ai parametri importati, chiama o la funzione "Main_Writer_macro_cut" nel caso in cui la geometria sia di tipo Macro oppure la funzione "Main_Writer_cut" nel caso in cui non lo sia.
Il suffisso "_cut" nel nome delle funzioni indica il processo con cui tali geometrie verranno realizzate al CaD.
Si è notato infatti come sia computazionalmente più semplice e veloce per *SALOME* realizzare un solido pieno descritto dalle coordinate del *case*, dal quale togliere tutti i singoli solidi semplici per realizzare la geometria finale del grano piuttosto che fonderli insieme. Questa operazione Booleana si chiama appunto Cut da cui il nome delle funzioni sopra citate.

Nel caso in cui l'utente selezioni l'opzione "*Show ONLY internal empty shape*" presente nel pannello Mesh Generator, le funzioni che verranno chiamate saranno "Main_Write_macro" nel caso di geometria di tipo Macro o "Main_Write" la cui unica differenza dalle precedenti sta nel fatto che non vengono fornite le istruzioni per la rimozione dei solidi semplici bensì, non venendo preso in considerazione il case, quelle per la fusione degli stessi tramite l'operazione Booleana Fuse.
Attraverso le funzioni sopra elencate vengono quindi scritte tutte le forme dei solidi semplici, le informazioni del case, se necessario le istruzioni per un corretto assemblaggio di un unico solido, tutte le specifiche per la mesh della geometria risultante ed infine le indicazioni per esportare tale mesh in formato *.stl*.

# CHIUSURA MESH PER CFD

Sempre su richiesta di Avio all'interno del codice ROOBOST è stata inserita una sezione per la chiusura della mesh durante una qualsiasi iterazione della regressione del grano di propellente (combustione) da cui poter ricostruire una mesh tridimensionale per potervi effettuare simulazioni CFD. Tali simulazioni infatti richiedono un dominio chiuso per poter essere effettuate, motivo per il quale non basta semplicemente esportare un qualsiasi *frame* della simulazione ma è richiesto un lavoro più ampio di seguito descritto.

Questa parte del codice prende come dato di ingresso un "fotogramma" del grano di propellente durante la combustione, ne cerca i bordi e li chiude con la parte del *case* dove è già avvenuta la completa combustione del propellente.
I problemi che si riscontrati sono principalmente legati al fatto che per unire la mesh del grano parzialmente combusto con quella del *case* adiacente è necessario avere uno stesso numero di nodi nei bordi delle due mesh, ma quasi sempre tali mesh differivano in alcuni punti: si devono quindi eliminare i nodi in eccesso in modo tale che risulti possibile congiungere le due mesh.

Eliminare un nodo significa aprire un triangolo con cui è descritta la superficie della mesh: bisogna quindi trovare un algoritmo il più robusto possibile che cerchi di richiudere il triangolo che conteneva il nodo eliminato. Nelle figure seguenti si vuole esporre un esempio di come funziona il procedimento appena descritto.

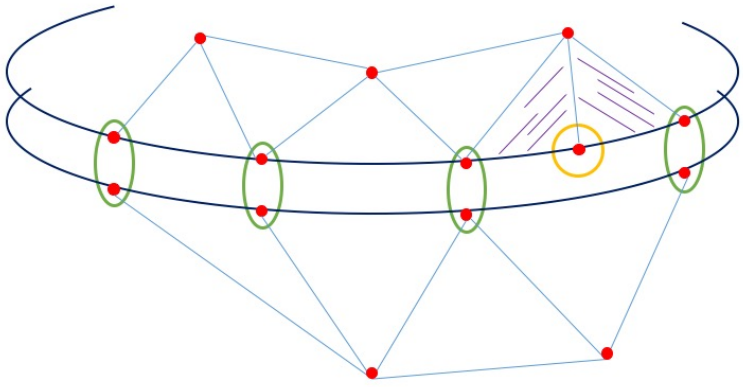


*Figura 20 - Chiusura mesh per simulazioni CFD(1).*

In un possibile scenario i punti rossi rappresentano i nodi della mesh e le due linee blu si riferiscono al bordo della mesh del grano in regressione (linea inferiore) e al corrispettivo bordo del *case* (linea superiore).

Come già anticipato può accadere di non avere un ugual numero di nodi nei due bordi quindi vengono cerchiati in verde i nodi che sono già disposti in una corretta posizione per essere uniti mentre in giallo viene cerchiato il nodo da eliminare che una volta cancellato lascerebbe aperti i due triangoli le cui aree sono tratteggiate in viola. A questo punto basterebbe solamente unire le due aree in una unica (tratteggiata in verde nella figura seguente) per sistemare localmente i triangoli della mesh.

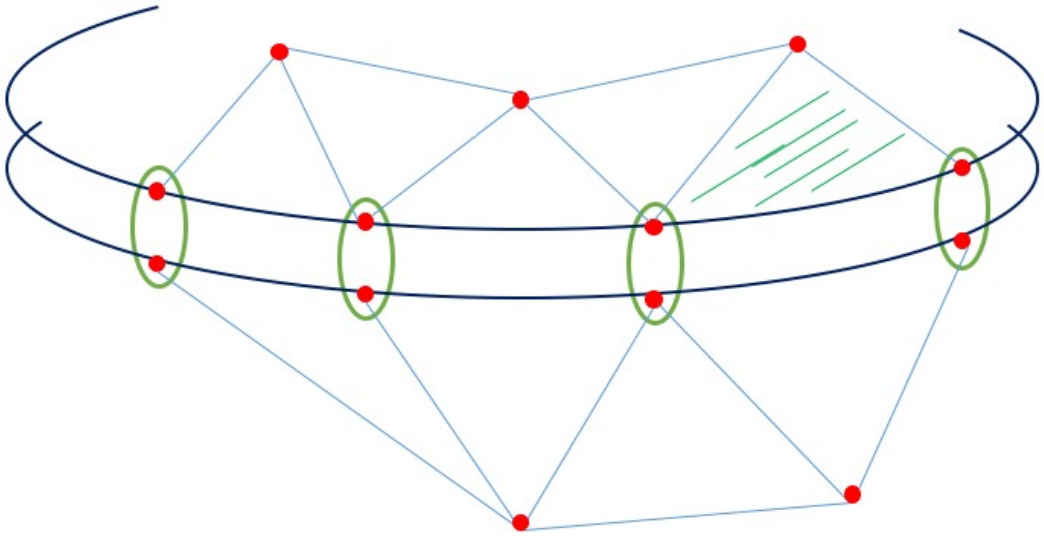


*Figura 21 - Chiusura mesh per simulazioni CFD(2).*

In realtà si possono avere situazioni in cui il nodo da eliminare forma ben più di due triangoli quindi diventa assai più complessa la ripartizione delle aree in quanto una volta cancellato tale nodo produrrebbe un quadrilatero invece che un altro triangolo come nel caso appena descritto.

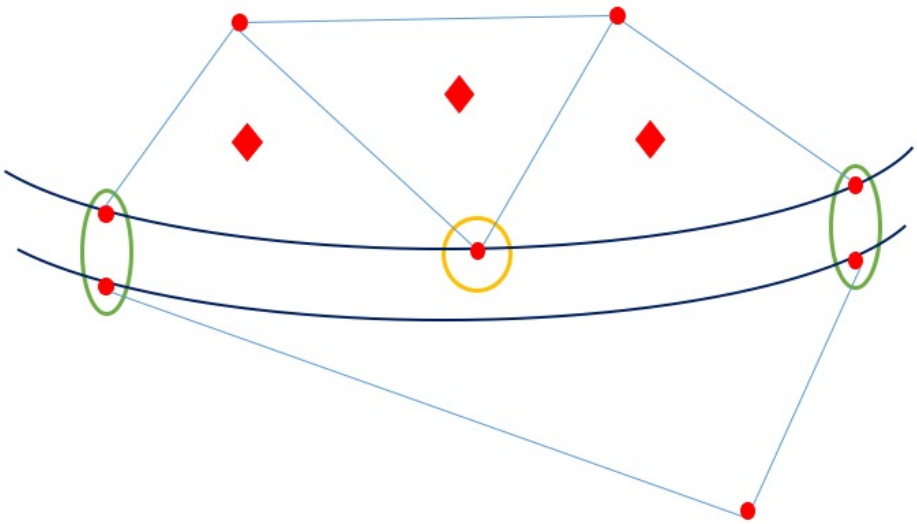


*Figura 22 - Chiusura mesh per simulazioni CFD(3).*

Una soluzione più generica al problema, ed estendibile anche al caso precedente, è quella non di cancellare il nodo in più bensì di spostarlo in uno degli incentri dei triangoli (rappresentati dai rombi rossi nella *Figura 21*) a cui quest'ultimo è collegato. In questo modo non ci sono problemi nella ripartizione delle aree dei triangoli interessati, il nodo in eccesso non è più presente nel bordo da unire a quello del grano in regressione, si ottengono tanti triangoli quanti se ne avevano prima più uno formato dal nodo da eliminare connesso al suo precedente e al suo successivo come mostrato in *Figura 22*.

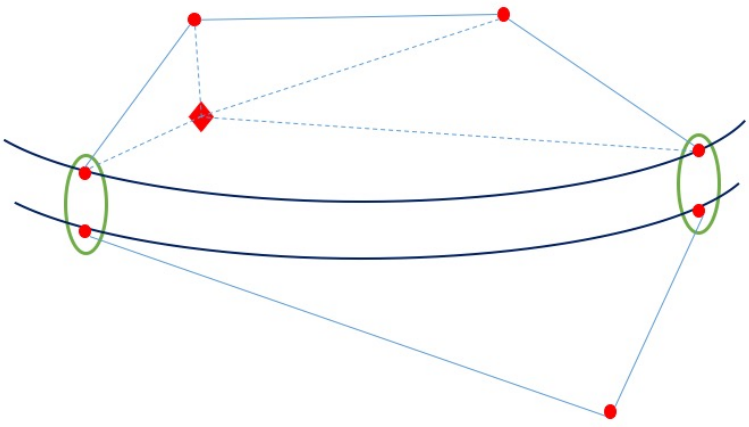


*Figura 23 - Chiusura mesh per simulazioni CFD(4).*

A questo punto è sufficiente cambiare le coordinate del nodo che è stato spostato dal bordo della mesh nell'incentro del triangolo ed aggiungere alla matrice *faces* il nuovo triangolo che si è venuto a creare facendo attenzione che la normale alla sua area sia orientata in maniera concorde con le normali degli altri triangoli costituenti la mesh. In figura viene riportato uno schema a blocchi riassuntivo del procedimento appena descritto.

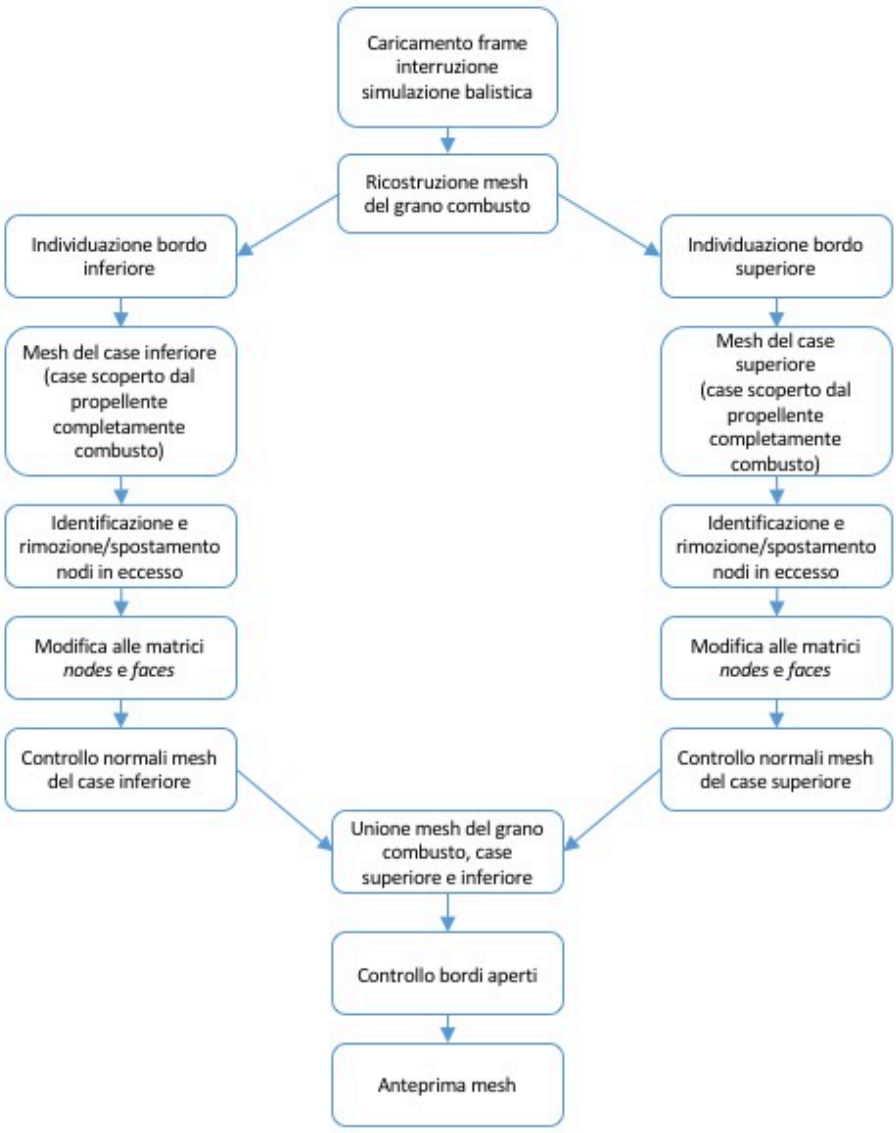



*Figura 24 - Schema a blocchi chiusura mesh per simulazioni CFD.*

Vengono inoltre qui di seguito riportate due immagini per meglio rendere l'idea di ciò da cui si parte per il processo di chiusura delle mesh ed il risultato finale.

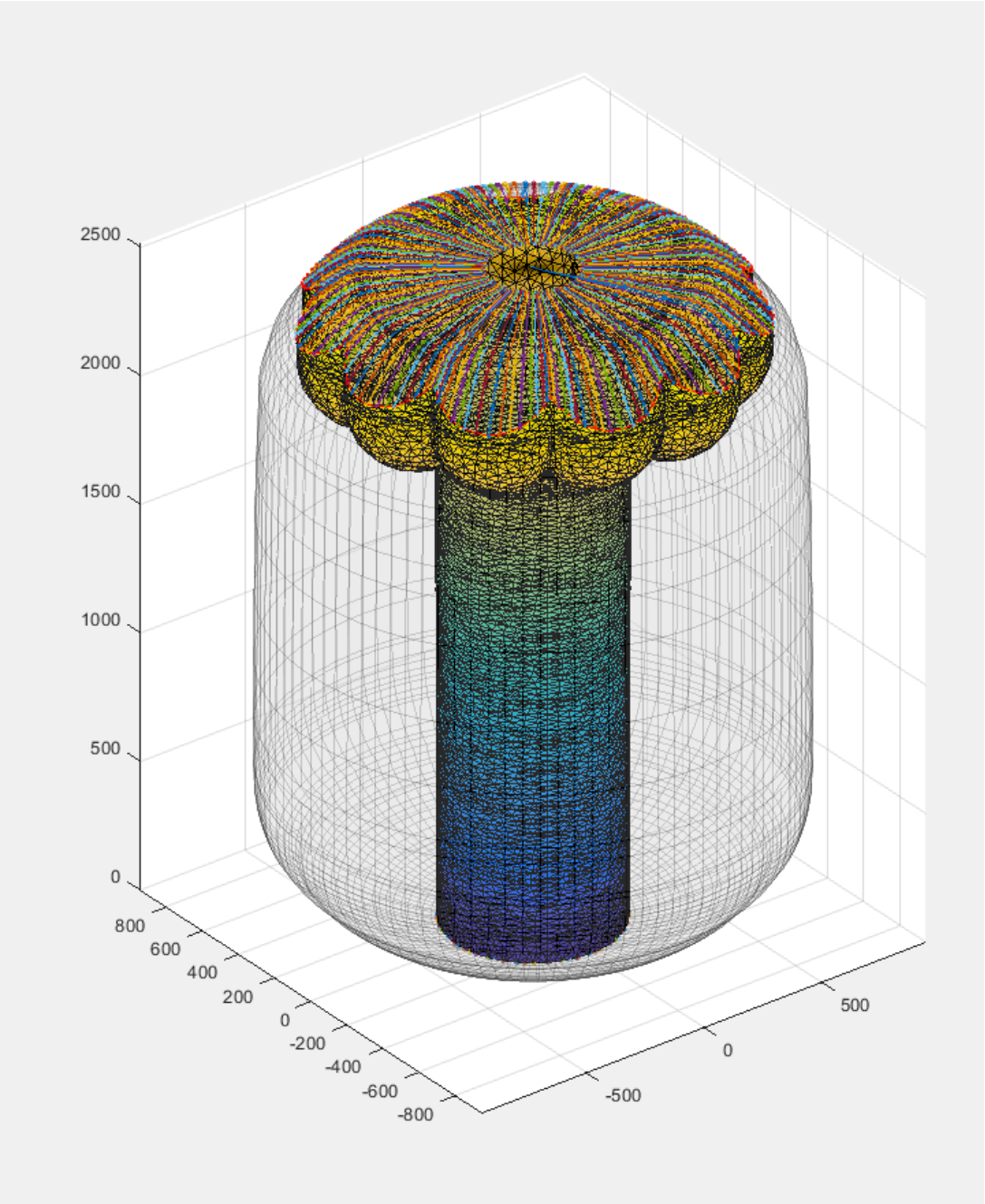



*Figura 25 - Z9 chiusura mesh per CFD.*

In questa figura, rappresentate il grano di propellente dello Z9, il *case* superiore, dove il propellente è completamente combusto, è rappresentato dalle bande colorate che toccano la linea rossa dei nodi della mesh sottostante ovvero del grano in regressione.

A processo ultimato ciò che risulta è una mesh unica formata dai *case* superiore ed inferiore uniti alla mesh del grano in regressione chiusa ed orientata con le normali verso l'esterno come mostrato nella figura seguente.

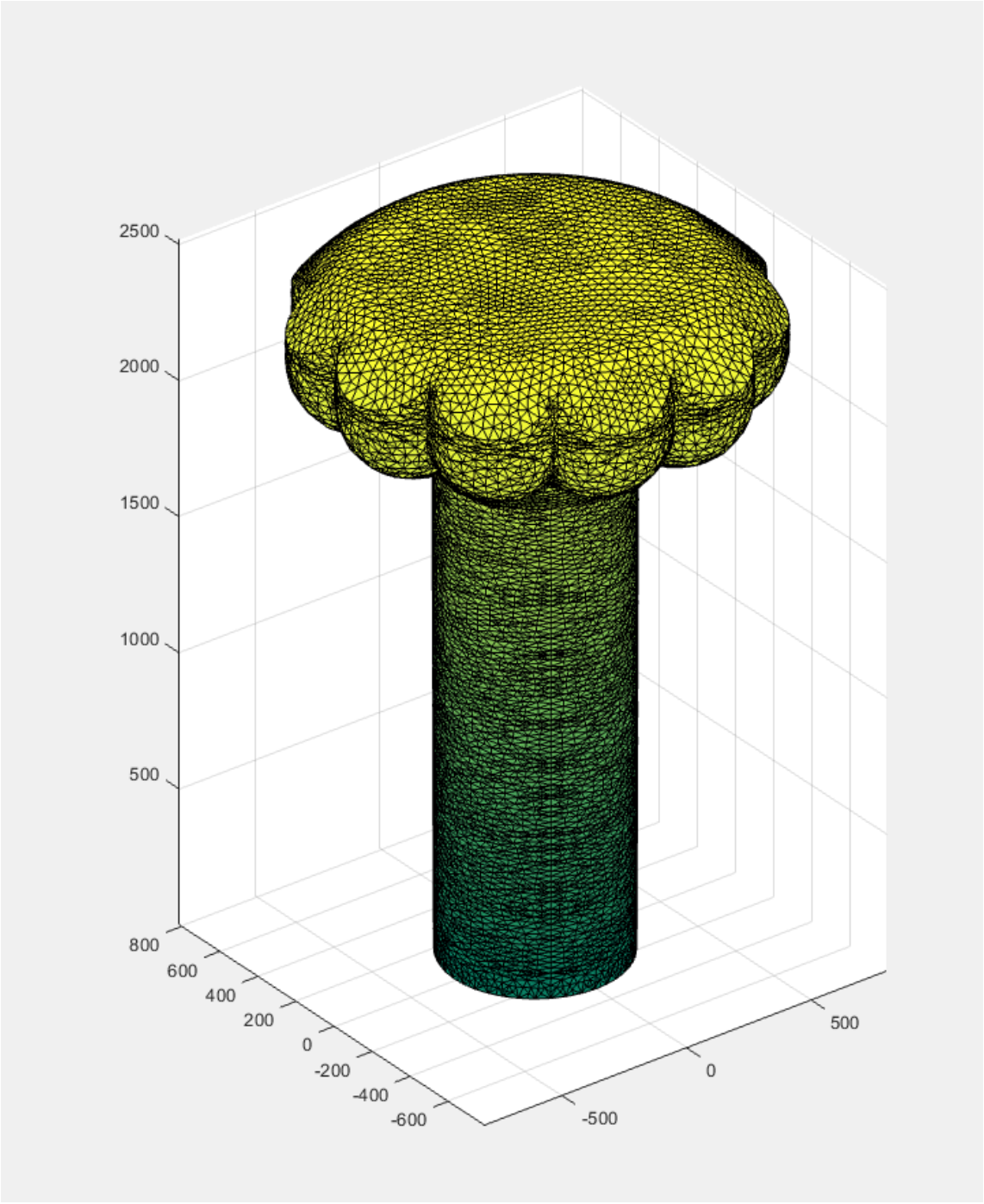



*Figura 26 - Z9 chiuso per mesh CFD.*

# PROBLEMI E SOLUZIONI ATTUALI

Si vogliono ora esporre le soluzioni trovate ai problemi più rilevanti riscontrati durante lo sviluppo di questo lavoro di tesi ovvero la realizzazione dei FIN nella geometria dello Z9, l'operazione Booleana Fuse per alcuni solidi semplici, l'errore di lettura dei file con la funzione "Stl_binary_reader".

I FIN come già anticipato precedentemente, sono delle particolari geometrie tridimensionali ricavate all'interno del grano di propellente. Nel codice SPP queste geometrie venivano ricavate dalla sovrapposizione di una serie di prismi con una faccia in comune e venivano descritte da sole due coordinate: XTAB ed RVTAB. Una volta ricavato un singolo FIN, con le informazioni sopra descritte, il parametro SYMFAC forniva il numero di ripetizioni di tale FIN intorno all'asse del motore : per lo Z9 ad esempio si realizzava metà FIN con SYMFAC=22 ovvero 11 FIN totali intorno al grano di propellente.

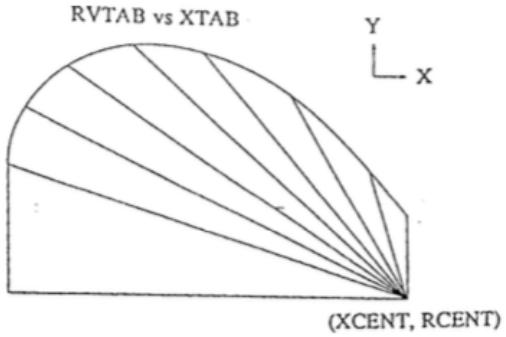



*Figura 27 - Realizzazione FIN con SPP.*

A questa operazione, non effettuabile tramite il CaD utilizzato, si è trovata come soluzione la realizzazione di una serie di "sketches" bidimensionali delle sezioni dei FIN lungo l'asse principale della geometria, i quali vengono estrusi in sequenza uno dopo l'altro per realizzare l'intera stella di FIN. Nella pagina seguente in *Figura 23* si possono notare in rosso gli "sketches" che una volta estrusi (*Figura 24*) formano la stella completa di FIN la quale completa la geometria dello Z9 (*Figura 25*).

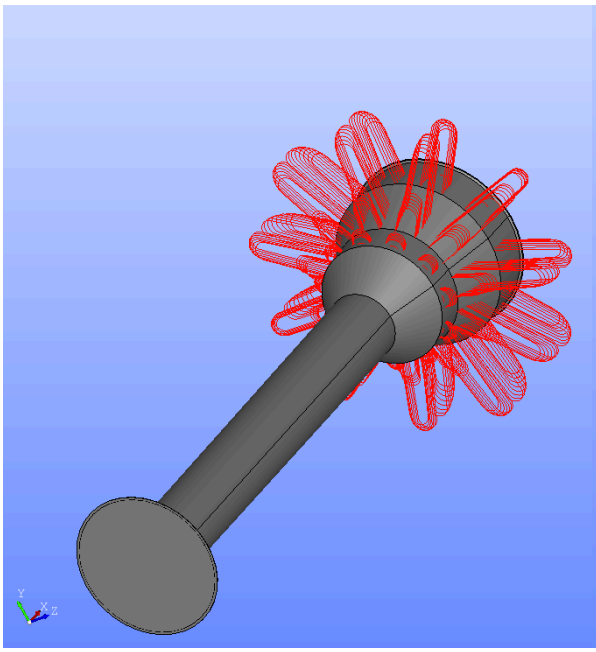


*Figura 28 - Realizzazione FIN su Z9 (1).*

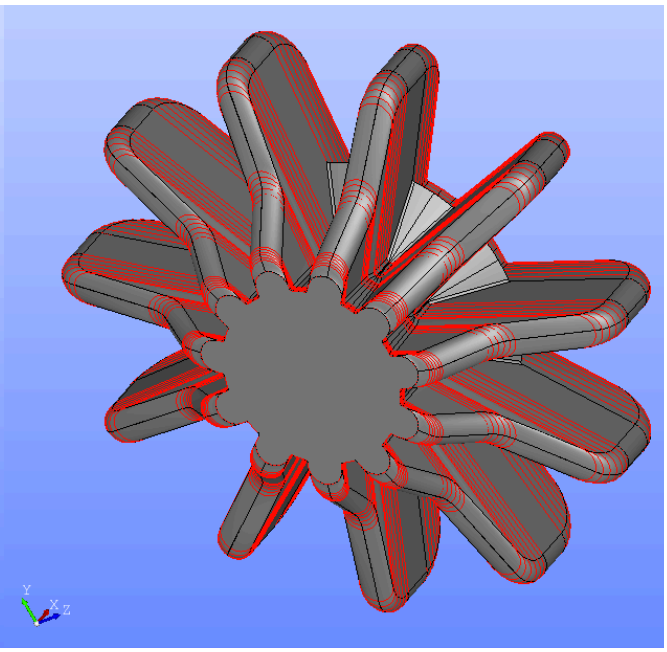


*Figura 29 - Realizzazione FIN su Z9 (2).*

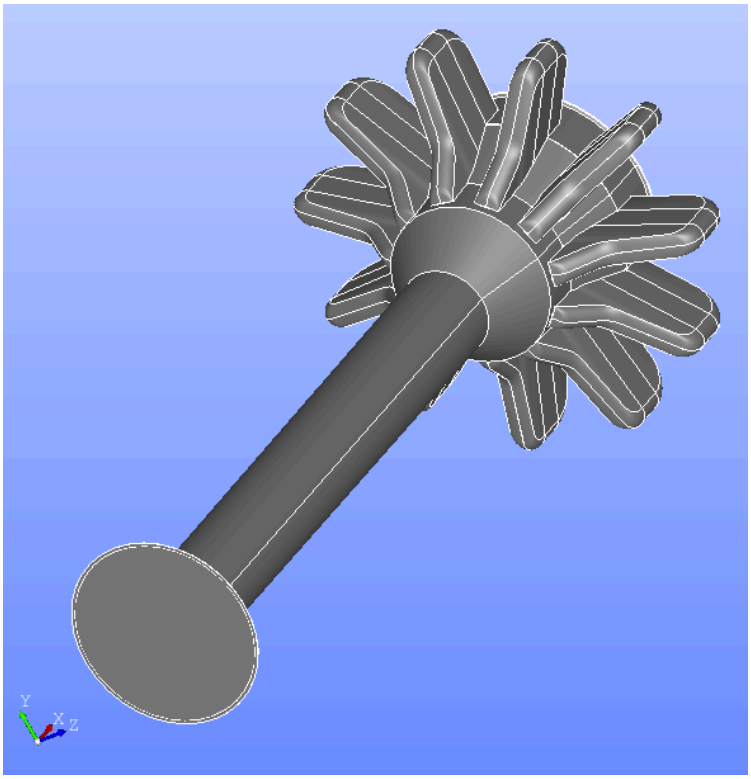


*Figura 30 - Z9 completo con FIN.*

Una volta realizzati i FIN si procede all'unione di tutti i singoli volumi per formare un'unica geometria. L'operazione Booleana Fuse che svolge questo compito riscontrava problemi nel caso in cui i volumi da unire fossero compenetranti, intersecanti tra loro o con bordi non congruenti.

In particolare sempre nella motorizzazione dello Z9, con i punti forniti per la descrizione dei FIN si andava a creare una *spline* anteriore e una posteriore per rappresentarne la forma corretta con il minimo errore; in mezzo a queste due *spline* a completamento dei FIN, andava inserita una geometria detta *stella centrale*.

Con il metodo di estrusione degli "sketches" bidimensionali per i FIN però la *stella centrale* risultava perfettamente allineata alla *spline* anteriore ma di dimensioni maggiori rispetto a quella posteriore: ciò non avrebbe rappresentato alcun problema in quanto *spline* anteriore e posteriore e *stella centrale* sarebbero state "annegate" all'interno della geometria principale del motore una volta effettuata l'operazione Fuse per la realizzazione di un unico solido.
Tuttavia proprio la non congruenza tra la *stella centrale* e la *spline* posteriore (*Figura 30*) rendeva impossibile la corretta esecuzione della funzione Booleana Fuse.

La soluzione trovata è stata quindi quella di non costruire la *stella centrale* come un solido a se stante ma di realizzarla per estrusione di "sketches" bidimensionali in particolare la faccia anteriore della *stella centrale* coincidente con l'ultimo degli "sketches" della *spline* anteriore e, allo stesso modo, la faccia posteriore con il primo degli "sketches" della spline posteriore.

Seguendo questo procedimento costruttivo non solo si sono risolti i problemi di incongruenza tra le *spline* e la *stella centrale* ma si è anche velocizzato l'intero processo di costruzione del grano di propellente in quanto è computazionalmente più semplice effettuare un'operazione Booleana di Fuse su solidi che hanno tuti gli spigoli coincidenti piuttosto che su solidi con facce adiacenti incongruenti tra loro.

Per realizzare la stella completa dei Fin ora si fondono la *spline* anteriore, quella posteriore e la *stella centrale* che sono tutti e tre solidi semplici, con facce congruenti tra loro, ottenuti tramite estrusione di "sketches" (*Figura 31*).

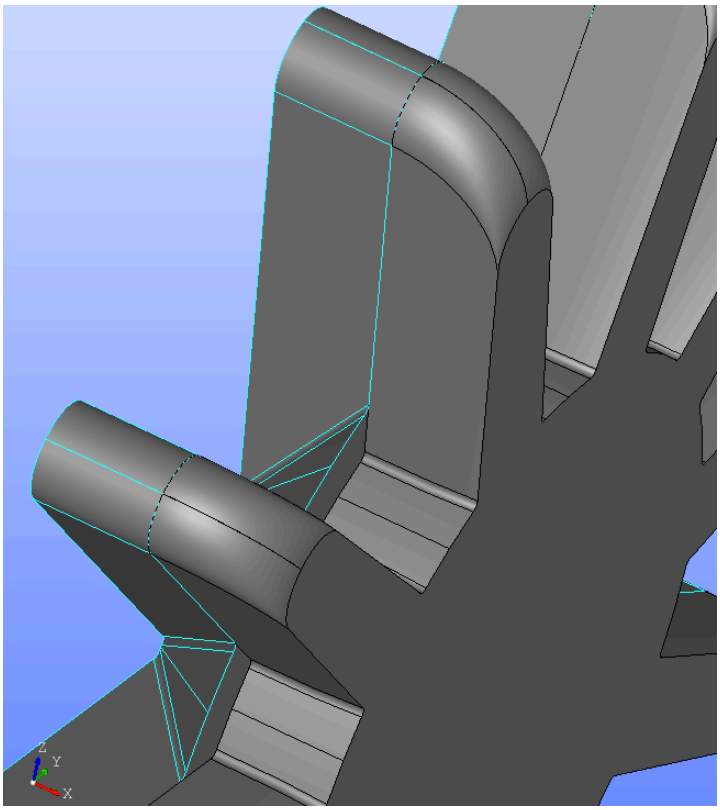


*Figura 31 - Incongruenza stella centrale con spline posteriore.*

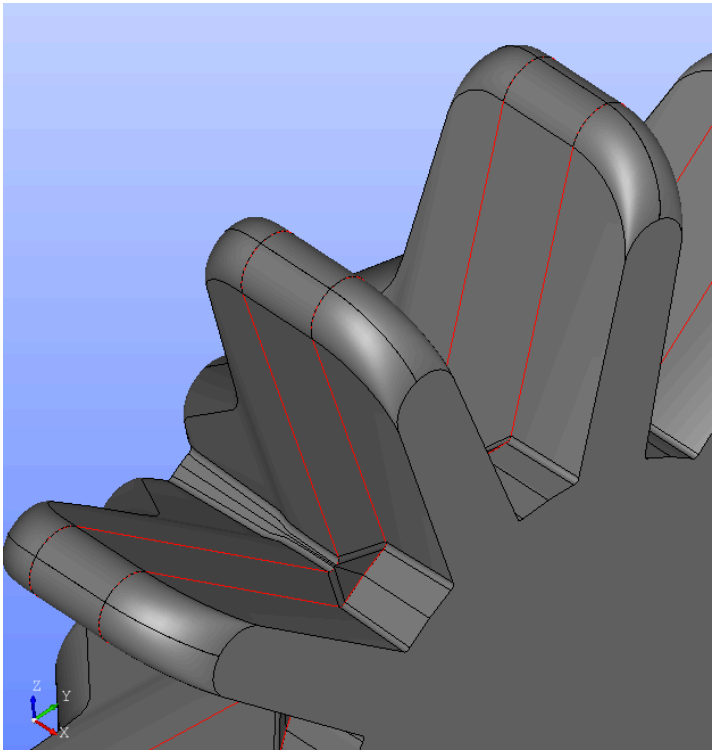


*Figura 32 - Estrusione stella centrale con "sketches" delle spline.*

Una volta costruita la geometria complessiva, quest'ultima viene affidata al modulo di meshing del CaD nel quale, dopo che vengono assegnati i parametri e le lunghezze caratteristiche precedentemente definite dall'utente, viene effettuata la mesh 3D.
Terminata questa operazione la mesh viene esportata in formato *.stl* per essere caricata e visualizzata dal codice ROBOOST.

La lettura del file *.stl* contenente la mesh è affidata alla funzione "Stl_binary_reader" che riceve come input il nome del file appena esportato da *SALOME* e fornisce come output due matrici: *nodes* contenente le coordinate dei vertici dei triangoli che compongono la mesh i quali vengono descritti nella matrice *faces* che riporta per ogni riga, ovvero ogni triangolo, gli indici dei nodi che lo compongono.
Il problema riscontrato con questa funzione è da attribuire alla velocità di scrittura del file esportato da SALOME. L'errore che si presentava era infatti l'impossibilità, da parte della funzione "Stl_binary_reader" di leggere il file appena creato.
Attraverso varie prove ed una debug approfondita si è giunti alla conclusione che nonostante il file sia presente, quindi esportato dal CaD, non è disponibile alla lettura poiché ancora in attesa di essere completato e chiuso da SALOME.

In un primo momento si è quindi provato a mettere in pausa l'esecuzione del codice prima di leggere il file ma, variando il tempo di scrittura del file *.stl* in base alle sue dimensioni e quindi in base alla grandezza del grano di propellente che viene costruito, si ottenevano risultati positivi solo con piccoli motori mentre per altri più grandi, come ad esempio lo Z9, l'errore persisteva.

Non avendo la possibilità di accedere ad una consolle di debug che fornisca informazioni riguardo i processi attualmente in uso da SALOME e allo stesso tempo non disponendo di un'elevata "verbosità" da parte dello stesso CaD, si è ritenuto utile pensare all'errore di lettura del file come un segnale per valutare la chiusura dello stesso una volta effettuata l'esportazione. La funzione "Stl_binary_reader" è stata posta infatti all'interno di una struttura *try-catch* ovvero il codice tenta di leggere il file e nel caso il cui si presenti l'errore dopo una pausa di due secondi il codice ritenta le lettura, fino a quando, una volta che il file viene chiuso dal Cad, non presentandosi l'errore l'operazione va a buon fine. In questo modo non si pongono problemi di

attesa in base alle dimensioni delle geometrie da caricare in quanto, essendo un'operazione ciclica, l'esecuzione del codice prosegue non appena viene terminata la scrittura del file da parte di SALOME e viene quindi immediatamente letto dalla funzione.
Effettuata la lettura del file e create le matrici *faces* e *nodes*, fondamentali per le operazioni successive e per il funzionamento dell'intero codice di simulazione ROOBOST, si è sicuri che il lavoro del Cad è terminato e viene quindi data l'istruzione di chiusura del CaD SALOME.
Infine vengono caricate le mesh e ne viene mostrata un'anteprima come in *Figura 20*.

# RISULTATI E SVILUPPI FUTURI

Si vogliono ora mostrare i risultati ottenuti, in particolare quelli della motorizzazione Delta. Si può notare dalle figure seguenti come la curva della superficie combusta in funzione del tempo ricavata da una simulazione effettuata su ROBOOST con il caricamento della geometria SPP (raffigurata in alto), sia uguale alla stessa ricavata da una simulazione con il vecchio codice SPP (raffigurata in basso).

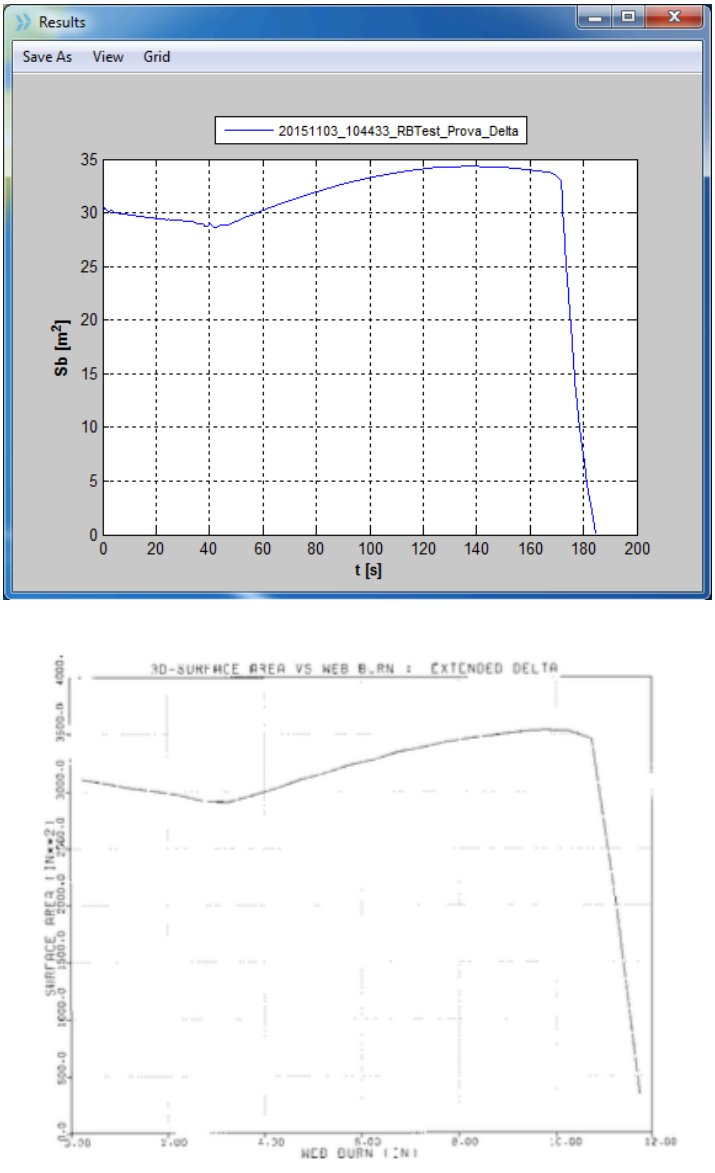



*Figura 33 - Confronto curve superficie combusta in funzione del tempo (Delta).*

Inoltre si riportano dei *frames* della regressione del grano di propellente ottenuti dalla simulazione con ROBOOST(sulla destra) paragonati a quelli delle simulazioni con il vecchio codice SPP (sulla sinistra) i quali vengono rappresentati con una diversa lunghezza degli assi da cui l'effetto "tozzo" e schiacciato di tali immagini.

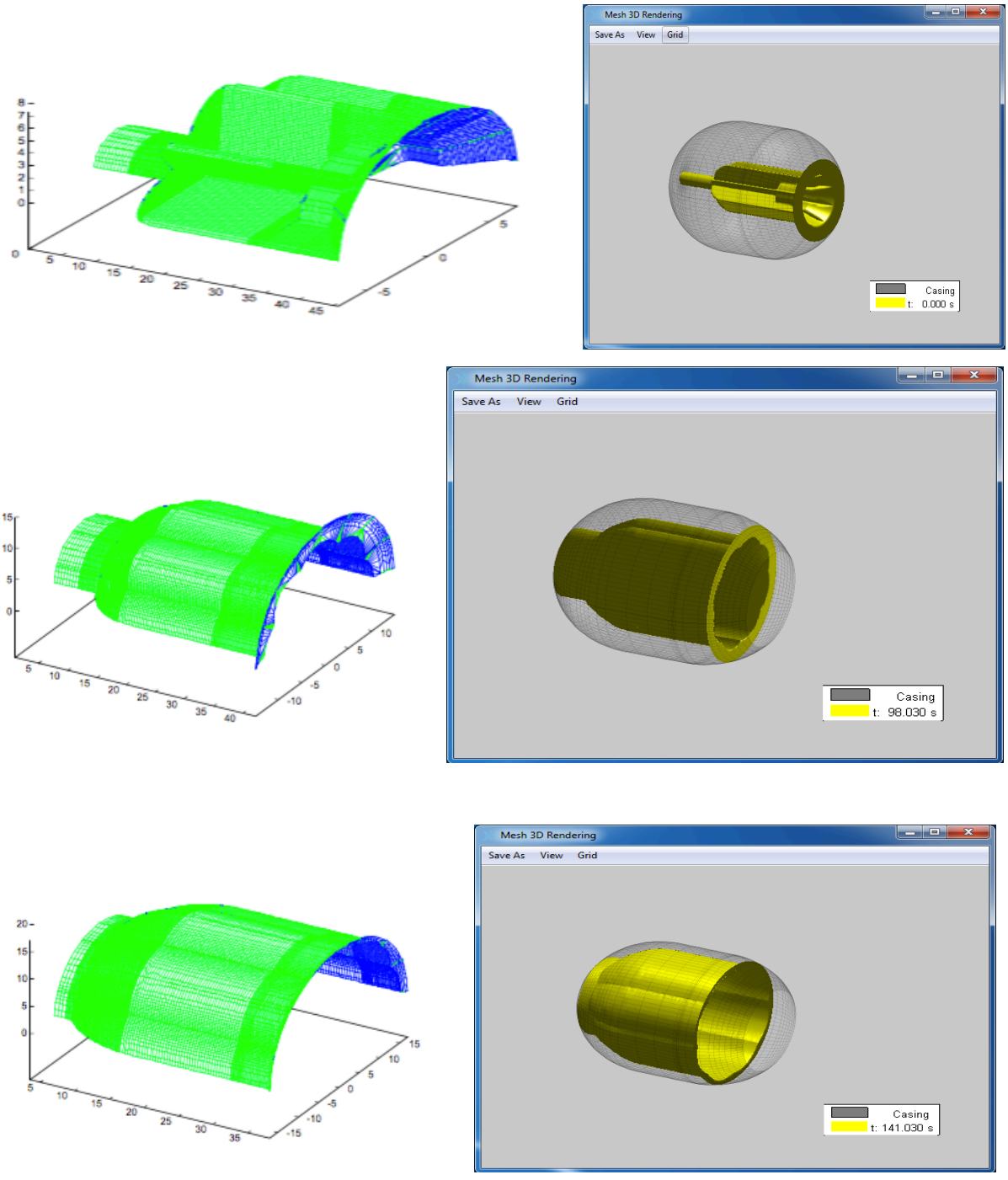



*Figura 34 - Confronto evoluzione temporale regressione grano (Delta).*

Anche da questo confronto si può notare come il grano regredisca durante la combustione secondo le aspettative durante i vari intervalli temporali presi in considerazione, indice del buon funzionamento del codice realizzato e quindi del raggiungimento degli obbiettivi prefissati.

Il lavoro di tesi qui esposto presenta molte possibilità per futuri sviluppi.
Avendo a diposizione solamente due geometrie con cui sviluppare l'interfaccia grafica, il codice risulta ottimizzato per queste motorizzazioni ma allo stesso tempo molto specifico. Disporre di un maggior numero di configurazioni del grano di propellente darebbe la possibilità di rendere il codice affetto da meno probabilità di errore, più robusto e più efficiente testandolo sotto diversi aspetti e con una più ampia casistica di geometrie dei grani di propellente.

Si potrebbe inoltre rendere più efficiente anche il metodo con cui vengono realizzate le geometrie dal CaD: il processo utilizzato per l'estrusione degli "sketches" dei FIN si potrebbe estendere, se accuratamente revisionato, per realizzare l'intera geometria del grano essendo sviluppata lungo un asse principale di simmetria.
Tale idea è già stata sperimentata durante lo sviluppo del codice ma con risultati insoddisfacenti a causa di una scarsa versatilità di SALOME; per questo miglioramento si dovrà quindi aspettare un aggiornamento del CaD poiché non ancora molto efficiente sotto alcuni aspetti costruttivi.

Una generica revisione alle funzioni ed agli script che vengono chiamati e mandati in esecuzione dal codice permetterebbe una miglior ottimizzazione delle variabili definite e della memoria utilizzata rendendo il processo più efficiente e dal punto di vista computazionale meno costoso e più veloce.

# CONCLUSIONI

In definitiva, è stata illustrata la realizzazione e l'implementazione dell'interfaccia grafica richiesta da Avio Spa all'interno del codice ROBOOST.

Sono stati trattati tutti gli aspetti relativi a tale codice in particolare l'integrazione di software di terze parti come SALOME all'interno del processo ricostruttivo della geometria interna delle motorizzazioni e le problematiche legate alla ancora scarsa versatilità di tale CaD, la realizzazione delle funzioni che permettono la decodifica delle geometrie dal formato SPP e quelle che permettono la conversione di tali informazioni in un formato compatibile con il CaD.

I risultati ottenuti e precedentemente esposti, soddisfano i requisiti richiesti anche se hanno un margine di miglioramento sia dal punto di vista della stabilità e robustezza del codice sia dal punto di vista della velocità di esecuzione.

Il fatto di avere a disposizione solamente due geometrie ha limitato la possibilità di rendere subito effettivi questi miglioramenti e l'impossibilità di gestire al massimo le potenzialità del CaD ha fatto si che si trovassero soluzioni alternative ad alcuni problemi le quali hanno inevitabilmente creato imprecisioni che sono comunque rimaste trascurabili o ininfluenti.

Si considerano quindi il codice sviluppato e l'interfaccia grafica realizzata ed implementata all'interno del codice soddisfacenti per gli obbiettivi prefissati.

# BIBLIOGRAFIA


[1] *Filippo Carra* - Modellazione Della Combustione Di Endoreattori A Solido In Presenza Di Difettosità Interne Al Grano – LM tesi 2013-2014.

[2] *Roberto Bertacin* - Modellazione Tridimensionale del Processo di Combustione di un Razzo a Propellente solido - PhD thesis, 2013.

[3] *Matteo Marcantoni* - Realizzazione manuale di utilizzo per codice di simulazione balistica 3D ROBOOST (ROCKET BOOST SIMULATION TOOL) finalizzato all'azienda Avio spa. – 2015.

[4] *G. P. Sutton, O. Biblarz* – Rocket Propulsion Elements – 7th edition, 2001.

[5] *B.J. Lee and P. B. Burchfield* - Solid Propellant Rocket Motor Performance Computer Programs Using The Group Transformation Method – 1966.

[6] *National Aeronautics and Space Administration* – Solid Rocket Motor Performance Analysis and Prediction – 1971.

[7] *D. E. Coats, J. N. Lavine* (*National Technical Information Service*) – A Computer Program for the Prediction of Solid Propellant Rocket Motor Performance. Volume 1. – 1975.

[8] *S.S. Dunn and D.E. Coats* - AIAA 97-3340 – 3-D Grain Design and Ballistic Analysis Using the SPP97 Code – 1997.