



**UNIVERSIDADE
ESTADUAL DO
MARANHÃO**



*Fundação de Amparo à Pesquisa e ao Desenvolvimento
Científico e Tecnológico do Maranhão*

UNIVERSIDADE ESTADUAL DO MARANHÃO - UEMA

CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT

**PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO E
SISTEMAS**

MESTRADO PROFISSIONAL EM SISTEMAS AEROESPACIAIS

CARLOS RONYHELTON SANTANA DE OLIVEIRA

**PROJETO E DESENVOLVIMENTO DE UMA PLATAFORMA EM LINGUAGEM
JAVA PARA SIMULAÇÃO DE VOO DE FOGUETES**

São Luís

2018

CARLOS RONYHELTON SANTANA DE OLIVEIRA

PROJETO E DESENVOLVIMENTO DE UMA PLATAFORMA EM LINGUAGEM
JAVA PARA SIMULAÇÃO DE VOO DE FOGUETES

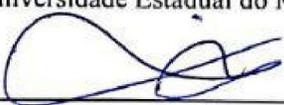
Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia de Computação e Sistemas da Universidade Estadual do Maranhão como parte dos requisitos para obtenção do título de Mestre em Engenharia de Computação, sob a orientação do Prof.º Henrique Mariano Costa do Amaral.

Aprovado em: 27/07/2018



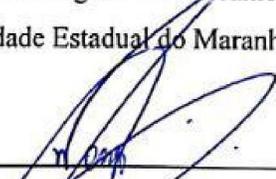
Prof.º Henrique Mariano Costa do Amaral (Orientador)

Mestre em Engenharia de Sistemas e Computação
Universidade Estadual do Maranhão



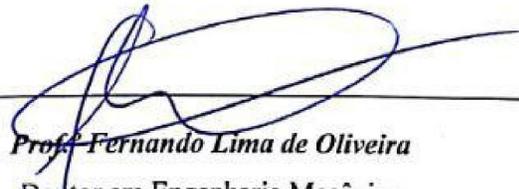
Prof.º Lourival Matos de Sousa Filho (Coorientador)

Doutor em Engenharia Mecânica
Universidade Estadual do Maranhão



Prof.º Mauro Sérgio Silva Pinto

Doutor em Engenharia Elétrica
Universidade Estadual do Maranhão



Prof.º Fernando Lima de Oliveira

Doutor em Engenharia Mecânica
Universidade Estadual do Maranhão

Tudo posso Naquele que me fortalece

(Filipenses 4.13)

A Deus pela luz da vida e Seu amor por mim. A minha Mãe pelo seu cuidado e dedicação. Ao meu Pai por sua força e certeza. Aos meus irmãos Carlos Nerilton, Carlos Nihelton, Anna Crhystina e minha tia/comadre Sueli pelo carinho e apoio tanto a mim quanto aos meus pais e avó. A Daniela pelo seu amor e apoio aos meus projetos. Aos meus familiares, amigos e professores que sempre acreditaram em mim e me apoiaram para que eu chegasse até aqui.

AGRADECIMENTOS

A Deus pela luz da vida e Sua sempre presença em toda a minha existência e em tudo que eu faço na forma inigualável de bênçãos e proteção. Ao senhor toda honra e toda a glória hoje e sempre.

Ao meu casal favorito de todo o cosmos que me concebeu e me formou como homem que hoje sou e sei que a eles devo tudo o que já conquistei e que ainda hei de conquistar. Do mais profundo do meu coração obrigado meus pais Cícero Alves e Ana Amélia!

A minha jornada nunca teria sido a mesma e nem sucessos eu teria alcançado se não fosse o apoio, nas mais diversas situações, dos meus três irmãos: Carlos Nerilton, Carlos Nihelton e Anna Crhystina e também da minha tia/comadre e sempre parceira Maria Sueli.

A minha amada namorada, parceira e sempre amiga Daniela de Sousa a quem eu agradeço muito por seu amor e apoio principalmente no desenvolvimento deste projeto.

A toda a minha família e também às minhas cunhadas Valéria e Larissa. A primeira agradeço por todas as orações e apoio que mesmo na distância nunca me faltaram. A segunda agradeço por toda força que me passou pelo tempo que nos conhecemos e também por sua colaboração no meu início de academia com o presente grandioso dos livros de cálculo e física fundamentais a minha formação e com o qual facilitou meu maior interesse pelas disciplinas que hoje leciono e as duas agradeço pela sabedoria de tantas palavras ditas.

A todos os meus amigos da turma de Engenharia Mecânica 2011.2 pelo sentimento de irmandade que alcançamos e a quem entre muitas curtições e grupos de estudos intensos me apoiaram para chegar até este momento único. Muito obrigado meus amigos engenheiros!

Aos amigos e também professores Kaio Henrique e Pedro Batalha o meu muitíssimo obrigado por todas as parcerias que tivemos principalmente nestes dois anos de mestrado seja na fase de disciplinas ou de pesquisas vocês foram fundamentais na minha jornada até aqui. Muito obrigado meus amigos!

Agradeço de forma muito especial por toda atenção e ajudas prestadas às amigas e secretárias do Programa de Pós-Graduação em Engenharia de Computação e Sistemas Karoline Meireles e Sara Martins e a Jocely de Araújo, secretária do curso de Eng. Mecânica, por todo apoio e força dedicada desde minha graduação até hoje.

A quatro pessoas da turma de Engenharia Mecânica 2011.2 em especial pelo mais que solidário acolhimento que me proporcionaram na chegada a São Luís e por mais que

amizade compartilhada: Jeyce, Rodrigo, Michael, Wangles, Márcio Silva, Marcelo, Kayo Botão e Raphael. Não foram poucas as ajudas que recebi de vocês nas mais diversas experiências (disciplinas, baja, iniciação científica, extensão, congressos) e muito menos capazes de serem um dia quitadas então não me resta muito a dizer a não ser: muito obrigado! Serei eternamente grato a cada um de vocês meus amigos.

A todos os meus professores que muito marcaram minha jornada escolar de uma forma mais que positiva, mais que amiga e solidária. Em especial a professora Dinalva Rodrigues que até hoje não sei como descrever a pessoa maravilhosa que ela é e que no meu Ensino Médio foi uma grande orientadora e amiga cuidadosa com minha educação e bem-estar e que tantos livros me presenteou nesse período para que eu pudesse ter a melhor experiência escolar possível. Muito obrigado tia Dinas!

A professora Me. Amália de Castro a quem nunca pedi algo para que não fosse atendido no dia anterior. Pelas orientações do Baja, pela disciplina de Introdução a Engenharia, pelos conselhos dados, pela recomendação ao mestrado, pelos ensinamentos da sua gestão no Departamento de Engenharia Mecânica e Produção e pelo presente da sua amizade meu muito obrigado professora! Eternamente grato!

Ao meu professor recordista em orientações e trabalhos desenvolvidos: Prof. Me. Pestana Filho quem também me apadrinou uma carta de recomendação ao mestrado. Muito obrigado por todas as oportunidades que o senhor me agraciou e que tanto influenciaram na minha formação acadêmica e pessoal. Por todos os projetos, recomendação do mestrado e por sua amizade muito obrigado Prof. Pestana Filho.

Ao professor que já não habita mais esse plano existencial, mas que deixou a todos os seus alunos as suas sempre sábias palavras e conselhos fundamentas principalmente aos alunos de projetos especiais como eu fui: Prof. Chicão (Francisco Dias) meu muito obrigado!

Aos amigos que eu sempre guardo no peito e que tenho como inspirações nas mais diversas atividades do meu dia a dia. A todos os meus amigos irmãos muito obrigado!

Ao meu professor orientador de iniciação científica e do trabalho de conclusão de curso da graduação em Engenharia Mecânica, pesquisas que se tornaram inspirações para este trabalho: Prof. Dr. Fernando Lima de Oliveira, muito obrigado professor!

Ao curso de Engenharia Mecânica, seus professores e estagiárias por todo o acolhimento no meu ofício onde atuei em paralelo ao mestrado, em especial ao Prof. Me. Paulo Flexa (a quem agradeço por todas as oportunidades concedidas, pela confiança e por ser

uma inspiração de gestão e liderança profissional além do ser humano que és), Prof. Me. Flávio Nunes e Prof. Dr. Adilto Cunha e as estagiárias Joyce e Juliane.

Ao Professor Guilherme da Silveira pela colaboração com este trabalho de uma forma espetacular através de seus ensinamentos. Sua obra foi além de tudo uma fonte de inspiração muito grande para este projeto.

Aos professores que tanto dispuseram de seu tempo e crença nas minhas capacidades para o desenvolvimento do presente trabalho e que a eles muito devo por conhecimentos científicos, técnicos, profissionais, etc. bem como pela oportunidade única de desenvolver este projeto: Prof. Henrique Mariano Costa do Amaral e Prof. Lourival Matos de Sousa Filho. Muito obrigado por tudo que os senhores me proporcionaram nestes anos de trabalho e pelo acolhimento no mestrado. Por vossas amizades e ensinamentos muito obrigado professores!

E por fim, a Universidade Estadual do Maranhão (UEMA), a Fundação de Amparo à Pesquisa e ao Desenvolvimento Científico e Tecnológico do Maranhão (FAPEMA), ao Centro de Lançamento de Alcântara (CLA), ao Instituto de Aeronáutica e Espaço (IAE) e ao Instituto Nacional de Pesquisas Espaciais (INPE) pela oportunidade de realizar este projeto.

RESUMO

Foguetes são veículos destinados ao transporte de cargas e/ou pessoas ao espaço. Hoje, o desenvolvimento desses veículos é de importância estratégica para as mais diversas nações, pois garante a necessária autonomia para o acesso ao espaço. E tanto para auxiliar nas etapas de projeto quanto para a realização dos testes de voo os simuladores de trajetória de foguetes são essenciais para garantirem êxito nessas atividades, uma vez que, as situações de operações dos foguetes são de difíceis reproduções em terra e também pelos altos custos envolvidos nas missões desses veículos. O nível de eficiência dessa reprodução depende diretamente do grau de realismo do modelo adotado e da viabilidade de sua execução. O presente trabalho apresenta o desenvolvimento de uma plataforma computacional com capacidade de simulação de voo de foguetes em 6 GDL (graus de liberdade), sendo três graus de liberdade de posição linear, que descrevem a posição do centro de massa do veículo em relação à origem de um sistema de referência, e três graus de liberdade de posição angular. A integração dos modelos dinâmicos e cinemáticos foi realizada por uma rotina numérica do método de Runge-Kutta de 4ª ordem com incremento de 10^{-6} . O objetivo do desenvolvimento do simulador foi obter a variação no tempo dos parâmetros de velocidade relativa, pressão dinâmica, número de Mach, alcance e altitude. Além da obtenção do comportamento destas variáveis foi alcançada uma característica de flexibilidade quanto a diferentes veículos e missões de foguetes, a portabilidade do simulador quanto aos sistemas operacionais mais populares (Windows, Linux e Mac) e a garantia de integrar todas as suas funcionalidades a uma interface gráfica intuitiva. Por fim, este projeto foi compilado em uma referência didática sobre simuladores de trajetórias de foguetes como uma tentativa de difusão de conhecimentos pertinentes ao setor aeroespacial.

Palavras-chave: 1. Simulador; 2. Foguete; 3. Trajetória; 4. Graus de liberdade.

ABSTRACT

Rockets are vehicles intended for transporting cargo and/or people to space. Today, the development of these vehicles is of strategic importance to the most diverse nations, because it guarantees the necessary autonomy for access to space. And both to assist in the design stages and the performance of the flight tests the rocket trajectory simulators are essential to ensure success in these activities, since the situations of rocket operations are difficult reproductions in And also by the high costs involved in the missions of these vehicles. The level of efficiency of this reproduction depends directly on the degree of realism of the adopted model and the feasibility of its execution. The present work presents the development of a computational platform with simulation capacity of rocket flight in 6 GDL (degrees of freedom), being three degrees of freedom of linear position, which describe the position of the center of mass of the vehicle in Relation to the origin of a reference system, and three degrees of freedom of angular position. The integration of the dynamic and Kinematic models was performed by a numerical routine of the Runge-Kutta 4th order method with an increment of 10^{-6} . The objective of the simulator development was to obtain the time variation of relative velocity parameters, dynamic pressure, Mach number, reach and altitude. In addition to obtaining the behavior of these variables was achieved a characteristic of flexibility for different vehicles and rocket missions, the portability of the simulator for the most popular operating systems (Windows, Linux and Mac) and the guarantee to integrate all of its functionalities into an intuitive graphical interface. Finally, this project was compiled into a didactic reference on rocket-trajectory simulators as an attempt to disseminate knowledge pertinent to the aerospace sector.

Key words: 1. Simulator; 2. Rocket; 3. Trajectory; 4. Degrees of freedom.

LISTA DE FIGURAS

Figura 1 - Perfil da missão do VLS-1	31
Figura 2 - Indicação dos quatro estágios do VLS-1.....	32
Figura 3 - Diagrama simplificado para queda de um corpo	33
Figura 4 - Centros de Lançamento Brasileiros: (a) Centro de Lançamento de Alcântara (CLA) e (b) Centro de Lançamento da Barreira do Inferno (CLBI).....	40
Figura 5 - Proximidade do Centro de Lançamento de Alcântara (CLA) com a Linha do Equador	41
Figura 6 - Sítio de lançamento do CLA com a TMI recuada e o VLS	42
Figura 7 - VLS-1 V02 montado na mesa de lançamento do CLA.....	43
Figura 8 - Início da decolagem do VLS-1 V02	44
Figura 10 – Desenvolvimento de uma classe do simulador no NetBeans	51
Figura 11 - Direção inercial de referência.....	55
Figura 12 - Relação entre velocidade inercial e velocidade relativa	59
Figura 13 – Fluxograma de desenvolvimento do Simulador AMELIA.....	68
Figura 14 – Modelagem do banco de dados no <i>software</i> MySQL WorkBench.....	70
Figura 15 - Formato da conexão da aplicação com o banco de dados.....	71
Figura 16 - Estrutura da classe itens.java.....	72
Figura 17 - Parte da implementação da classe itensDAO.java.....	73
Figura 18 – Rotina de salvamento da matriz de dados em arquivo .txt.....	74
Figura 19 – Interface de abertura de arquivos	75
Figura 20 - Sequência de eventos para salvamento e abertura de arquivos	75
Figura 21 – Interface de acompanhamento do processo de importação de dados.....	76
Figura 22 – Interface de obtenção dos dados das fases.....	78
Figura 23 – Preenchimento das condições de lançamento.....	78
Figura 24 – Sequência de entrada de dados do <i>software</i> AMELIA	79
Figura 25 – Implementação do método readJTable.....	80
Figura 26 - Aplicação do <i>design</i> Nimbus.....	81
Figura 27 – Interface Principal do AMELIA	81
Figura 28 – Subdivisões lógicas da Interface Principal	82
Figura 29 – Interface de resultados ativa	83
Figura 30 – Coleta dos dados para plotagem	84
Figura 31 – Interface Principal com a <i>JInternalFrame</i> ativa	85

Figura 32 – Fluxograma de funcionamento do AMELIA.....	87
Figura 33 – Gráfico do Alcance x Tempo para o VSB-30 no simulador AMELIA.....	90
Figura 34 – Gráfico de comparação do Alcance x Tempo para o VSB-30 nas plataformas AMELIA, RTS e ROSI.....	91
Figura 35 – Comparação do Alcance x Tempo para o VSB-30 no AMELIA e no RTS.....	91
Figura 36 – Gráfico da Altitude x Tempo para o VSB-30 no simulador AMELIA.....	92
Figura 37 – Gráfico de comparação Altitude x Tempo para o VSB-30 nas plataformas AMELIA, RTS e ROSI.....	92
Figura 38 – Gráfico da Altitude x Alcance para o VSB-30 no simulador AMELIA	93
Figura 39 – Gráfico de comparação do Altitude x Alcance para o VSB-30 nas plataformas AMELIA, RTS e ROSI.....	94
Figura 40 – Gráfico da Velocidade Rel. x Tempo para o VSB-30 no simulador AMELIA....	95
Figura 41 – Gráfico de comparação da Velocidade Relativa x Tempo para o VSB-30 nas plataformas AMELIA, RTS e ROSI.....	95
Figura 42 – Gráfico da Pressão Dinâmica x Tempo para o VSB-30 no simulador AMELIA.	96
Figura 43 – Gráfico de comparação da Pressão Dinâmica x Tempo para o VSB-30 nas plataformas AMELIA, RTS e ROSI.....	96
Figura 44 – Gráfico do Número de Mach x Tempo para o VSB-30 no simulador AMELIA .	97
Figura 45 – Gráfico de comparação do Número de Mach x Tempo para o VSB-30 nas plataformas AMELIA, RTS e ROSI.....	97
Figura 46 – Gráfico da Velocidade Angular x Tempo para o VSB-30 no simulador AMELIA	98
Figura 47 – Gráfico da comparação da Velocidade Angular x Tempo para o VSB-30 nas plataformas AMELIA, RTS e ROSI.....	98
Figura A.1 – Interface Principal do AMELIA.....	107
Figura A.2 – Interface Principal com menu e teclas de atalho exibidos.....	107
Figura A.3 - Entrada de dados das condições lançamento.....	108
Figura A.4 - Entrada de dados do foguete.....	108
Figura A.5 – Abrindo nova simulação.....	109
Figura A.6 – Abrindo nova pasta de trabalho.....	109
Figura A.7 – Salvando dados de simulação (Botão Salvar).....	110
Figura A.8 – Salvando dados de simulação (Botão Salvar como...).....	110
Figura A.9 – Carregando dados de simulação anterior.....	111
Figura A.10 – Deletando dados de simulação	111

Figura A.11 – Saídas do <i>Software</i> AMELIA	112
--	-----

LISTA DE TABELAS

Tabela 1 - Balanço energético típico para um veículo biestágio	37
Tabela 2 – Parâmetros das fases de voo do foguete	76
Tabela 3 – Modelamento para simulação do VSB-30 nos simuladores ROSI, RTS e AMELIA	89
Tabela 4 – Comparação dos valores máximos de altitude	93
Tabela 5 – Comparação dos valores máximos de velocidade relativa.....	94

LISTA DE ABREVIATURAS E SIGLAS

AEB	-	Agência Espacial Brasileira
API	-	<i>Application Programming Interface</i> (Interface de Programação de Aplicativos)
CLA	-	Centro de Lançamento de Alcântara
CLBI	-	Centro de Lançamento da Barreira do Inferno
GDL	-	Graus de Liberdade
IAE	-	Instituto de Aeronáutica e Espaço
INPE	-	Instituto Nacional de Pesquisas Espaciais
JDBC	-	<i>Java Database Connectivity</i> (conectividade com bancos de dados em Java)
JIT	-	<i>just in time</i>
JVM	-	Máquina Virtual Java
MECB	-	Missão Espacial Completa Brasileira
PNAE	-	Programa Nacional de Atividades Espaciais
PNDAE	-	Política Nacional de Desenvolvimento das Atividades Espaciais
ROSI	-	<i>Rocket Simulation</i>
RTS	-	<i>Rocket Trajectory Simulator</i>
STVLS	-	Programa de Cálculo de Trajetórias do VLS
TMI	-	Torre Móvel de Integração
VLM	-	Veículo Lançador de Microssatélites
VLS	-	Veículo Lançador de Satélites
VSBS	-	Veículo de Sondagem Brasileiro

LISTA DE SÍMBOLOS

A_e	-	Área de saída da tubeira [m^2]
B	-	Sistema do corpo
C_A	-	Coefficiente de força axial
$C_{l\delta}$	-	Derivada do coeficiente de momento de rolamento em relação à incidência do conjunto de empenas
C_{lp}	-	Derivada do coeficiente de momento de rolamento em relação à velocidade de rolamento
C_{mq}	-	Derivada do coeficiente de momento de arfagem/guinada em relação à velocidade de arfagem/guinada
$C_{N\alpha}$	-	Derivada do coeficiente de força lateral/normal em relação ao ângulo de derrapagem/ataque
e_x	-	Excentricidade
\vec{F}	-	Somatório de forças de um corpo em queda livre [N]
\vec{F}_A	-	Força aerodinâmica [N]
\vec{F}_T	-	Força de empuxo [N]
\vec{F}_{total}	-	Soma das forças atuantes sobre o veículo [N]
\vec{g}	-	Aceleração gravitacional [m/s^2]
G	-	Constante da gravitação universal [$m^3 kg^{-1} s^{-2}$]
GC	-	Sistema geocêntrico
h	-	Altitude geométrica do veículo [km]
h_k	-	Passo ou incremento
h_{gp}	-	Altitude geopotencial do veículo [km]
$h_{gp,b}$	-	Altitude geopotencial da base da camada [km]
I	-	Tensor de inércia do veículo [$kg.m^2$]
L	-	Sistema do lançador
L_b	-	Taxa de variação da temperatura em relação à altitude geopotencial na camada considerada [K/km]
L_{ref}	-	Comprimento aerodinâmico de referência [m]
m	-	Massa do corpo [Kg]
\dot{m}	-	Vazão de massa [kg/s]

M	-	Massa da Terra [kg]
m_0	-	Massa inicial do veículo [kg]
M_A	-	Momento aerodinâmico [kg·m/s]
M_{atm}	-	Massa molar do ar atmosférico [g/mol]
m_f	-	Massa após a queima do propelente [kg]
\vec{M}_{total}	-	Soma dos momentos atuantes sobre o veículo [kg·m/s]
ρ_{atm}	-	Densidade atmosférica [kg/m ³]
p_{din}	-	Pressão dinâmica [Pa]
p_l	-	Componentes da velocidade angular local do veículo [rad/s]
p_{ref}	-	Pressão de referência [Pa]
q_l	-	Componentes da velocidade angular local do veículo [rad/s]
R	-	Raio da órbita em relação ao centro da Terra [Km]
r	-	Raio da órbita [Km]
R_{atm}	-	Constante dos gases para o ar atmosférico [J/(mol·K)]
$r_{cm/n}$	-	Posição do centro de massa do veículo em relação ao nariz [m]
r_e	-	Posição do centro de vazão de massa em relação ao centro de massa do veículo [m]
r_l	-	Componentes da velocidade angular local do veículo [rad/s]
\vec{R}_{gc}	-	Posição inercial do veículo [m]
R_{gc}	-	Raio geocêntrico [km]
S_{ref}	-	Área aerodinâmica de referência [m ²]
T_{atm}	-	Temperatura atmosférica [K]
\vec{T}_{cor}	-	Empuxo corrigido [N]
t_p	-	Tempo de queima do propelente [s]
\vec{T}_{ref}	-	Empuxo nominal [N]
T_{tail}	-	Empuxo no início da cauda de empuxo [N]
\vec{V}_{atm}	-	Velocidade atmosférica do veículo [m/s]
\vec{V}_c	-	Velocidade do corpo em órbita circular [m/s]
\vec{V}_e	-	Velocidade média de exaustão do propelente queimado [m/s]
\vec{V}_{esc}	-	Velocidade de escape [m/s]
\vec{V}_p	-	Velocidade vertical do veículo [m/s]

\vec{V}_{rel}	-	Velocidade relativa [m/s]
\vec{V}_{som}	-	Velocidade do som [m/s]
\vec{W}	-	Velocidade do vento [m/s]
\vec{w}	-	Velocidade de rotação [rad/s]
w_p	-	Vazão em peso do propelente [N/s]
x	-	Velocidade efetiva de exaustão [m/s]
x_{cm}	-	Distância entre o centro de massa e o nariz do veículo [m]
x_{cp}	-	Distância entre o centro de pressão e o nariz do veículo [m]
γ	-	Coefficiente de resistência do ar
α	-	Ângulo de ataque [graus]
α_0	-	Aceleração no início da queima do propelente [m/s ²]
β	-	Ângulo de derrapagem [graus]
λ_{gc}	-	Latitude geocêntrica [graus]
δ	-	Incidência do conjunto de empenas [graus]
δ_l	-	Ângulo de lançamento [graus]
θ	-	Ângulo de arfagem [graus]
θ_M	-	Semieixo maior da órbita [graus]
σ	-	Ângulo polar [graus]
ρ_{atm}	-	Densidade atmosférica [kg/m ³]
μ	-	Longitude do veículo [graus]
μt	-	Constante gravitacional da Terra [m ³ kg ⁻¹ s ⁻²]
ϕ	-	Ângulo de guinada [graus]
ψ	-	Ângulo de rolamento [graus]
\vec{w}	-	Velocidade angular inercial do veículo [rad/s]
$\vec{\Omega}$	-	Velocidade angular da Terra [rad/s]
y	-	Razão entre os calores específicos do ar à pressão e a volume constantes

SUMÁRIO

1 INTRODUÇÃO	20
1.1 Motivação do Trabalho.....	21
1.2 Objetivos.....	22
1.2.1 Objetivo Geral.....	22
1.2.2 Objetivos Específicos	22
1.3 Estrutura do trabalho	23
2 REVISÃO DA LITERATURA	25
2.1 Tipos de simulação.....	25
2.2 Simulação Computacional	26
2.3 Configurações de Lançamento (Problema de Valor Inicial - PVI).....	28
2.3.1 Missão do veículo	33
2.3.1.1 Requisito de altitude	33
2.3.1.2 Requisito de velocidade	35
2.3.1.3 Balanço de Energia	37
2.4 Simuladores Modernos.....	38
2.4.1 ROSI (<i>Rocket Simulation</i>)	38
2.4.2 STVLS (Programa de Cálculo de Trajetórias do VLS).....	38
2.4.3 <i>Rocket Trajectory Simulator</i> (RTS)	39
2.5 Centros de Lançamento	39
2.5.1 Centro de Lançamento da Barreira do Inferno (CLBI).....	40
2.5.2 Centro de Lançamento de Alcântara (CLA).....	41
2.5.2.1 Fases do VLS-1 no Centro de Lançamento.....	42
2.5.2.1.1 Fase de pré-lançamento	43
2.5.2.1.2 Fase de lançamento	44
2.5.2.1.3 Fase de pós-lançamento.....	45
2.6 Rotinas de Integração Numérica	45
2.7 Linguagem de programação Java.....	47
2.7.1 Biblioteca JFreeChart	49
2.7.2 Ambiente de Desenvolvimento Integrado NetBeans	50
2.7.3 Persistência de Dados em Java por banco de dados.....	51
3 MODELOS MATEMÁTICOS PARA SIMULAÇÃO DE VOO DE FOGUETES.....	54
3.1 Modelos do ambiente	54

3.2 Modelos dinâmicos e cinemáticos	59
3.2.1 Modelo para movimento do foguete a partir do trilho de lançamento	60
3.2.2 Modelo para movimento do foguete a partir da rampa de lançamento	61
3.3 Modelos dos subsistemas de um foguete.....	62
4 METODOLOGIA DE DESENVOLVIMENTO DO SIMULADOR.....	67
4.1 Armazenamento, acesso e deleção de dados	69
4.1.1 Implementação da persistência de dados por banco de dados.....	69
4.1.2 Implementação da persistência de dados por arquivos (Serialização)	73
4.2. Estrutura da entrada de dados	76
4.3 Interface gráfica	80
4.4 Saídas do <i>Software</i> AMELIA	83
5 RESULTADOS E DISCUSSÕES	86
5.1 Mecanismos de funcionamento.....	86
5.2 Simulação de veículo suborbital não controlado	88
5.2.1 Resultados de alcance.....	90
5.2.2 Resultados de altitude.....	92
5.2.3 Resultados da velocidade relativa	94
5.2.4 Resultados da pressão dinâmica.....	96
5.2.5 Resultados do Número de Mach	97
5.2.6 Resultados da velocidade de rolamento	98
6 CONSIDERAÇÕES FINAIS	99
6.1 Sugestões de trabalhos futuros.....	100
REFERÊNCIAS	102
APÊNDICE A - INTERFACES DA PLATAFORMA AMELIA.....	107

1 INTRODUÇÃO

Desde a pré-história o homem sempre teve interesse na locomoção de objetos, a citar pela invenção da roda que deu origem aos meios de transporte primitivos que com longos períodos de evoluções propiciaram a criação dos veículos modernos. Na sequência, destaca-se outro forte interesse que, há muitos séculos, também acompanha o homem: a sua curiosidade sobre ambientes desconhecidos, representado por inúmeras tentativas de viagens por terra, mares e também pelo espaço sideral na busca de conhecer e conquistar novos territórios, fato que obviamente deu origem aos veículos aeroespaciais.

Hoje, os veículos lançadores ou foguetes espaciais são peças fundamentais para o desenvolvimento da astronáutica, sendo capazes de lançar ao espaço instrumentos como sondas interplanetárias que revelam segredos de planetas distantes e satélites com variadas funções. O desenvolvimento desses veículos, orbitais e suborbitais, é de importância estratégica, pois garante a necessária autonomia para o acesso ao espaço (AEB, 2012).

No contexto desses desenvolvimentos ressalta-se a importância do aumento do poder de processamento dos computadores modernos que abrem mais e mais possibilidades a cada dia para funcionamento das rotinas de simulações de engenharia, que se tornaram ferramentas atrativas e indispensáveis dentro desses processos. De maneira generalista, a principal motivação para o uso de simuladores é a redução do tempo de projeto e a redução de custos relativos ao desenvolvimento de equipamentos e protótipos de hardware (MARTIN; CARVALHO, 2005).

O simulador é o *software* capaz de interpretar o modelo criado e executá-lo em um computador, reproduzindo o comportamento desejado (GOULD; TOBOCHNIK, 1996). A fidelidade dessa reprodução depende diretamente do nível de detalhes do modelo e da viabilidade de sua execução.

Em engenharia aeroespacial, simuladores de voo de foguetes podem ser empregados para os mais diversos propósitos vinculados desde a concepção de projetos desses veículos a fases de lançamento e pós-lançamento. Durante a etapa de concepção, prototipação e desenvolvimento de um foguete, os parâmetros de voo obtidos através das simulações auxiliam no projeto dos diferentes subsistemas do veículo, desde o estrutural ao propulsivo. Nesta fase, empregam-se uma série de simplificações dos modelos de voo uma vez que ainda não se conhecem uma gama de detalhes dos subsistemas que o compõem. A fidelidade à realidade aumenta em proporção ao aumento da definição de parâmetros do foguete.

Na fase de operação de um foguete, os simuladores de voo cooperam para a avaliação do desempenho do veículo para diferentes estratégias de lançamento, determinando a trajetória nominal a ser percorrida pelo mesmo que segue em concordância com a missão para o qual o veículo foi concebido a cumprir e é utilizada nos procedimentos de navegação e controle do veículo durante o período de voo.

As arquiteturas dos sistemas de simulação não se diferenciam muito das arquiteturas de *softwares* de outra natureza. Atualmente, sistemas implementados em linguagens modernas, como por exemplo, C++ e Java, utilizam amplamente os conceitos de orientação a objetos para promover o reuso, flexibilidade e interoperabilidade das aplicações. Esses conceitos são praticados comumente no universo de simuladores para aplicações espaciais e até mesmo para utilizações em sistemas embarcados (TURNER, 2006).

Este trabalho concentra-se exatamente neste contexto. Propõe-se aqui o projeto e construção de um *software* de simulação de trajetória de voo de foguetes programado em linguagem Java. A escolha dessa linguagem foi baseada no nível de portabilidade do programa gerado uma vez que quando se compila um arquivo em Java dele se obtém um bytecode que é interpretado numa Máquina Virtual Java (JVM). Dessa forma, basta que um determinado sistema operacional (Windows, GNU/Linux ou Mac) tenha uma JVM, então nele será possível executar o referido *software*.

1.1 Motivação do Trabalho

Foi publicada, em 21 de janeiro de 2013, pela AEB (Agência Espacial Brasileira), a nova revisão do Programa Nacional de Atividades Espaciais (PNAE), que engloba o período entre 2012 e 2021. Pelo proposto nessa revisão, foram e ainda serão gerados inúmeros desafios à academia e à indústria nacional voltados para o desenvolvimento da política espacial brasileira.

A soberania e autonomia de um país estão proporcionalmente relacionadas à sua capacidade de desenvolvimento tecnológico. A tecnologia espacial é, sem dúvidas, a de maior amplitude nesse cenário (AEB, 2012).

Ainda de acordo com AEB (2012):

O Brasil está assumindo definitivamente esse compromisso de soberania e autonomia plena, ao enfatizar [...] suas prioridades de integração da política espacial às demais políticas públicas em execução, [...] reconhecendo o necessário domínio das tecnologias críticas e de acesso

restrito, com participação da indústria, junto com a competência e o talento existente nas universidades e institutos de pesquisa nacionais.

No contexto desses desafios, é necessário adquirir excelência no desenvolvimento de tecnologias aeroespaciais e com isso ter domínio da modelagem e simulação de voo foguetes figura como algo imprescindível, uma vez, que atualmente existem diversos *softwares* para este fim, mas que na contra partida estão vinculados ao controle governamental das nações onde foram desenvolvidos ou ainda foram implementados como ferramentas que funcionam em programas computacionais de alto valor aquisitivo e ao mesmo tempo também vinculados a outros países.

A consequência imediata no referido desenvolvimento é o crescimento científico do país acompanhado da redução da dependência externa do Brasil nesse ramo do conhecimento. E se tratando da complexidade de sistemas como foguetes e satélites atrelado a dificuldade de reprodução em laboratório das condições ambientais onde operam esses sistemas a utilização de técnicas de simulação é altamente atrativa (SILVEIRA, 2014). A construção de simuladores constitui, portanto, uma disciplina estratégica na jornada do PNAE e fundamental também por se tratar de um ramo do conhecimento que lida com tecnologia de fronteira, em projetos de longa duração, altos custos e que operam em ambientes que não podem ser reproduzidos na Terra (HOFFMANN; PERONDI, 2010).

1.2 Objetivos

1.2.1 Objetivo Geral

Projetar e desenvolver uma plataforma para simulação de voo de foguetes, que plote alcance e altitude por meio da integração das equações dinâmicas de movimento translacionais e rotacionais com aplicação para estudo da dinâmica de voo durante a fase de projetos e também durante a operação do referido veículo.

1.2.2 Objetivos Específicos

- ✓ Proporcionar capacidade de simulação em 6 GDL (graus de liberdade) através da plataforma;

- ✓ Desenvolver o simulador para que apresente flexibilidade quanto a diferentes veículos e missões;
- ✓ Integrar todas as funcionalidades do *software* a uma interface gráfica intuitiva e que dê acesso fácil a elas;
- ✓ Implementar a plataforma para que funcione independente de outros *softwares*, principalmente prioritários;
- ✓ Obter um simulador altamente portátil entre sistemas operacionais (Windows, Linux e Mac);
- ✓ Gerar uma referência didática sobre desenvolvimento de simuladores de trajetórias de voo de foguetes e em paralelo difundir conhecimentos pertinentes ao setor aeroespacial.

1.3 Estrutura do trabalho

Este trabalho está dividido em seis capítulos. No primeiro, como já visto, estão descritos a motivação, os objetivos e a estrutura de todo o seu texto.

O capítulo 2 apresenta uma revisão da literatura que transcorre desde a conceituação do que é simulação, seus tipos (ênfatizando a de natureza computacional) e os principais simuladores de voo de foguetes em utilização no Brasil. Aborda-se também o problema de lançamento de foguetes associado a um Problema de Valor Inicial bem como todos os principais parâmetros envolvidos neste cenário desde a missão do veículo, os requisitos a atingir, a plataforma da qual será lançado e a fases que ocorrem durante o voo.

Ainda neste capítulo são apresentados vários conceitos sobre a linguagem de programação Java e sobre o Ambiente de Desenvolvimento Integrado utilizado para desenvolvimento da plataforma de voo.

No capítulo 3 são definidas as considerações adotadas para formulação dos modelos matemáticos que regem os movimentos dos foguetes. Além disso, são apresentadas as diversas equações e configurações que irão compor a biblioteca de modelos do simulador de voo desenvolvido aqui.

O capítulo 4 aborda o desenvolvimento do simulador, apresenta sua arquitetura, interface e a maneira como foram implementadas as diversas rotinas que o compõem.

No capítulo 5 são apresentados os resultados obtidos da simulação de voo do VSB-30 com o simulador desenvolvido bem como uma análise comparativa com resultados

de outras ferramentas de simulação (*Rocket Simulation* – ROSI - e o *Rocket Trajectory Simulator* - RTS) para o mesmo caso.

Por fim, no capítulo 6 são apresentadas conclusões gerais do presente trabalho pertinentes aos resultados obtidos e são feitas sugestões para próximos estudos.

2 REVISÃO DA LITERATURA

Harrel et al. (2002) definem simulação como "um processo de experimentação com um modelo de um sistema real para determinar como o sistema responderá a mudanças em sua estrutura, ambiente ou condições de contorno". E dessa forma para Law e Kelton (1991) o modelo não necessariamente precisa representar todos os elementos do sistema, mas precisa ser capaz de simular a condição ou objetivo para o qual foi proposto.

Assim a simulação pode envolver protótipos - primeiros exemplares de um produto, construídos para testes - ou modelos submetidos a ambientes físicos reais. No caso particular de modelos matemáticos, eles são submetidos a distúrbios matemáticos para avaliar a condição de serviço esperada.

Duarte (2003) define então que, o processo de simulação consiste em transpor um problema que ocorra na realidade para um sistema onde as condições possam ser controladas e os efeitos decorrentes destas condições possam ser observados principalmente ao menor custo possível e economizando o maior tempo para este propósito.

Submeter um sistema real a simulação é evidentemente a melhor forma de se prever uma ou várias respostas do mesmo a solicitações futuras sem a necessidade, normalmente custosa financeiramente, de seguir sequência em etapas de fabricação que serão refletidas em despesas de protótipos com alto potencial de erros. Abaixo se apresentam os principais tipos de simulações existentes.

2.1 Tipos de simulação

De uma forma geral classificam-se as simulações em três tipos:

a) **Simulação Icônica:** aquela que se assemelha à realidade. Acontece quando o sistema físico real (SFR) é representado através de modelos físicos - geralmente com dimensões diferentes dos reais - com o propósito de verificar como ele funcionará. Para Bazzo e Pereira (2006) sua característica básica é o alto grau de semelhança com o seu equivalente real. Um exemplo típico de simulação icônica são os ensaios em túnel de vento para avaliar a influência da forma de um objeto no seu arrasto aerodinâmico (muito usado no projeto de carros e aviões).

b) **Simulação Analógica:** simulação que consiste em fazer um sistema comportar-se de modo análogo a outro. Uma aplicação simples consiste na utilização de uma mola mecânica para ilustrar uma resistência elétrica ou da água para representar o ar passando pelas

pás de uma turbina, por exemplo. A pouca semelhança existente entre os dois sistemas, o análogo e o real, é uma característica marcante deste tipo de simulação. Há dessa forma, uma exigência forte do profissional responsável à aplicação de bons conhecimentos dos fenômenos físicos básicos com os quais se associam essas experimentações como: fluidos, térmica, ótica, eletromagnetismo, dentre outras.

c) **Simulação Matemática:** trata-se de uma técnica de previsão muito útil, na qual as características essenciais dos elementos que compõe o sistema a simular são descritas por simbologia matemática. Dessa forma, os distúrbios nas variáveis envolvidas nas equações verificam o comportamento do sistema representado. Ela, portanto, fornece um modelo de previsão do tipo entrada-saída, onde são introduzidos os dados iniciais e obtém-se, na saída, o resultado final.

Para Lutz e Wendt (2010) os métodos para obtenção de modelos matemáticos são divididos pelas suas formulações e naturezas como analíticos ou teóricos, experimentais ou empíricos. Dessa forma, para eles os modelos analíticos são obtidos por meio de leis físicas enquanto os modelos experimentais são construídos por meio de medições no processo.

Outra divisão proposta por Lutz e Wendt (2010) sobre os modelos solicita suas classificações como paramétricos, quando representados em forma de equações diferenciais ou funções de transferência e não paramétricos, quando representados por curvas, tabela de dados ou funções de resposta do processo. Abdul Kadir et al. (2011) acrescentam que se necessita de apenas uma mudança em uma das variáveis e isso já é motivo para a execução de novos experimentos.

Como opção preferencial em uma realidade em que o avanço da computação é acompanhado em paralelo com a evolução das linguagens de programação, as simulações computacionais são possíveis de serem praticadas em praticamente todas as áreas de conhecimento que podem variar, por exemplo, desde a análise de cargas e escoamentos sobre um determinado corpo de geometrias complexas bem como com o desempenho de sistemas de produção. Evidentemente, pode-se então afirmar que as simulações computacionais se aplicam a todas as áreas as quais sejam necessários um auxílio para tomadas de decisões baseados em dados e eventos do mundo real.

2.2 Simulação Computacional

Para Yamanaka (2006) este tipo de simulação consiste na utilização de determinadas técnicas matemáticas, que podem ser implementadas por meio de

computadores, destinadas a representar o funcionamento de processos do mundo real. Pode ser entendida como uma técnica de solução baseada na análise de um modelo que descreve o comportamento do sistema por meio de um computador.

Entre as vantagens da simulação computacional podem-se citar:

- ✓ Possibilidade de se avaliar um sistema em várias condições sem causar danos ao sistema real;
- ✓ redução de custos na implementação efetiva do sistema;
- ✓ determinação de condições críticas do sistema;
- ✓ possibilidade de otimização do sistema por meio do auto teste em diversas configurações sem danos ao equipamento real;
- ✓ possibilidade de realização de testes sem a necessidade de uma bancada de testes, onde pode ocorrer a utilização de componentes defeituosos, que resultará em incertezas da localização do erro (erro de projeto, defeito de componente ou erros de montagem).

Dada à grande complexidade nos eventos relacionados à mobilidade no espaço tridimensional de sistemas como aviões, foguetes e satélites, combinada com a dificuldade de reprodução em laboratório das condições ambientais onde operam esses veículos, a utilização de técnicas de simulação computacional se torna evidentemente uma opção de suma importância. Baldesi e Toso (2012) afirmam que desde sua concepção, desenvolvimento e operação, a simulação dá suporte a um conjunto de atividades como verificação e validação de *softwares*, validação do projeto de subsistemas, verificação da interação entre diferentes subsistemas e avaliação do desempenho do sistema completo.

Dessa forma, os softwares de simulação de trajetória de foguetes devem apresentar as capacidades listadas por Zipfel (2007) como:

- ✓ Definições dos requisitos de desempenho;
- ✓ avaliação de diferentes configurações;
- ✓ suporte a testes;
- ✓ redução de custo de testes;
- ✓ investigação de ambientes inacessíveis;
- ✓ investigação de fenômenos; e
- ✓ integração entre subsistemas

Estudos de dinâmica de voo acompanham todas as etapas do projeto e utilização de um foguete, auxiliando desde sua concepção até a análise dos dados reais de voo. Sarma et al. (1978) alertam que o nível de fidelidade do estudo varia de acordo com três parâmetros principais: 1) seu objetivo; 2) as informações disponíveis do veículo e de seus subsistemas; e 3) o nível de acurácia desejado. Dessa forma, para uma proposta de solução aos problemas de dinâmica de voo de foguetes os modelos adotados consistem em sistemas de equações diferenciais acopladas e não lineares, mesmo quando o veículo é considerado um corpo rígido. Para tanto, destaca-se então a necessidade evidente da incorporação ao simulador de rotinas computacionais que contenham métodos numéricos direcionados a solução de PVIs (Problemas de Valor Inicial) para resolverem as equações que descrevem o referido fenômeno.

Neste cenário, torna-se então necessário à implementação de métodos computacionais para realizar a integração dos modelos a cada fase da trajetória, fornecendo os resultados dos diversos parâmetros de voo em função do tempo. Abaixo se apresenta um estudo resumido sobre em que consistem os problemas de valores iniciais (PVIs) direcionados ao contexto de foguetes e no tópico 2.6 quais são os métodos numéricos utilizados para suas soluções.

2.3 Configurações de Lançamento (Problema de Valor Inicial - PVI)

Para especificar completamente a solução, uma equação diferencial (ED) é normalmente acompanhada de condições auxiliares. Para equações diferenciais ordinárias (EDOs) de primeira ordem é necessário um tipo de condição auxiliar denominada valor inicial, para determinar a constante e obter uma solução única (CHAPRA, 2013).

As condições iniciais, em geral, têm interpretações físicas tangíveis nas equações diferenciais provenientes de situações de problemas físicos reais. No caso de um foguete, por exemplo, a condição inicial é um reflexo do fato físico de que, no instante zero ($t = 0$) a velocidade inicial é nula ($v = 0$). Se essa condição de movimento fosse alterada no instante zero então a solução teria como consequência alterações para esse valor de velocidade.

Em se tratando de equação diferencial de ordem n , serão necessárias n condições para obter uma solução única. Se todas as condições forem especificadas em um mesmo valor da variável independente então o problema é chamado de problema de valor inicial. Isso para diferenciá-lo dos problemas de valor de contorno, nos quais as especificações das condições ocorrem em valores diferentes da variável independente.

De acordo com a capacidade ou não de inserir algum objeto em órbita os foguetes podem ser classificados em orbitais e suborbitais (AEB, 2012). De maneira geral, o voo destes veículos pode ser dividido pelo menos nas seguintes fases: a) lançamento; b) voo atmosférico (movimento do veículo dentro da atmosfera terrestre); c) voo exoatmosférico (movimento do veículo fora da atmosfera terrestre); e d) inserção em órbita, no caso de veículos orbitais, ou a reentrada atmosférica, no caso de veículos suborbitais.

Segundo Palmerio (2017) a execução da trajetória é o grande objetivo dos estudos em dinâmica de voo, para os quais ele define as ações abaixo:

- ✓ Definição da trajetória nominal, baseada na missão (assunto a ser abordado no item 2.3.1) e nas condições de lançamento;
- ✓ estabelecimento das zonas de impacto das partes do foguete;
- ✓ estabelecimento da sensibilidade às perturbações externas e internas e da repercussão na precisão dos parâmetros finais de entrega da carga útil/paga;
- ✓ definição dos recursos necessários de rastreamento e de recuperação;
- ✓ participação nos estudos dos enlaces de telemetria e telecomando; e
- ✓ análise da trajetória real.

Para o caso dos estudos de definição da trajetória nominal e de sensibilidade com foguetes controlados deveriam então ser incluídas todas as características do sistema de controle a ser implantado o que não convém ao foco deste trabalho devido sua restrição para estes tipos de sistemas.

O cálculo da trajetória de um foguete é desenvolvido considerando o equilíbrio de forças e momentos ao qual o mesmo é submetido, tais como: movimentos do corpo como precessão e outros fatores externos como a perturbação devida ao vento (PALMERIO, 2017).

Já na fase de voo, o veículo pode também sofrer perturbações do vento, como a ocorrência de “derrapagem” através dos esforços normais gerados, consequência do indesejado ângulo de ataque que torna o voo instável. Outro fenômeno é a instabilidade giroscópica, quando o veículo passa a girar em torno do seu próprio eixo de precessão, podendo assim aparecer às situações de precessão, instabilidade e desvio se o veículo estiver em fase propulsada. Neste caso, o empuxo não fica mais tangente à trajetória e a soma desses fatores implica um maior arrasto do veículo, perda de energia, desempenho, precisão e conseqüentemente da dispersão do ponto de impacto, como é o caso de veículos não controlados (PALMERIO, 2017).

Neste cenário, na fase de projeto deve-se encontrar um modelo que simule de modo adequado o sistema físico real para reconstituir sua rotina funcional para análise de comportamento (SHIGLEY, 2005).

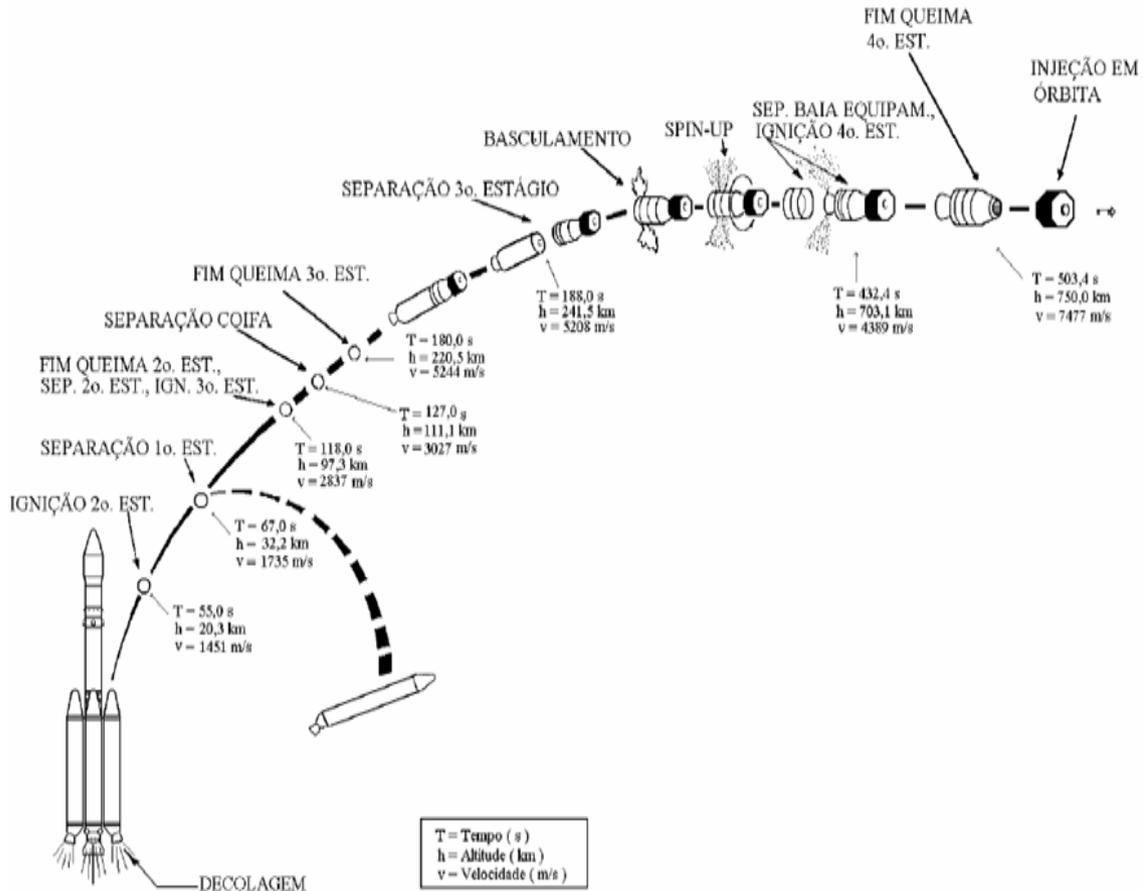
Para diferentes enfoques em função da dinâmica a ser representada vários modelos podem ser empregados para comparar os resultados. E mesmo utilizando técnicas de modelagens matemáticas ficam-se sujeitos as limitações na previsão de respostas dos fenômenos físicos. Erros (mesmo que mínimos) na etapa de formulação dos modelos ao final da simulação produzem resultados incoerentes com a real solução dos problemas. Esse erro então gera como consequência imediata a tomada de decisão, ou mesmo, sequência de decisões equivocadas dependendo da natureza do problema analisado. Para Ogata (2004) a ordem do erro está intimamente ligada às suposições e simplificações efetuadas pelo projetista, relacionando-se diretamente com o que ele entende do problema, o quanto da situação se decide incluir e o que necessita obter da solução.

Visando exemplificar as configurações dos foguetes bem como os eventos aos quais os mesmos são expostos em suas trajetórias de voo utiliza-se aqui o projeto brasileiro do VLS-1 (Veículo Lançador de Satélites-1) que teve início em 1984. Trata-se de um veículo da classe dos pequenos lançadores, descartável e com quatro estágios propulsores na decolagem. Sua propulsão é a base de propelente sólido tendo seus motores capacidade de colocar em órbita circular equatorial satélites de 115 Kg a uma altitude de 750 km (SALGADO, 2008).

O IAE é responsável pelo Programa de Veículos Lançadores nacionais e possui três subprogramas principais: (1) Foguetes de Sondagem; (2) Lançadores para Micro e Pequenos Satélites (subprograma ao qual o VLS-1 integra); e, (3) Lançadores de Satélites de Médio Porte.

A figura abaixo (Figura 1) demonstra quais são as fases de voo cumpridas pelo referido veículo e expõe algumas de suas configurações. Dela obtém-se uma noção detalhada dos principais eventos que serão expostos tanto o VLS-1 quanto todos os demais veículos de missões similares e serve como uma boa base para compreensão do resultado gráfico final do simulador proposto. Pela abrangência dos eventos que ocorrem para este veículo deu-se preferência à explanação dele no tópico que segue em relação ao VSB-30 que será simulado no capítulo 5 deste trabalho.

Figura 1 - Perfil da missão do VLS-1



Fonte: AEB (2012)

De acordo com Salgado (2008) o VLS-1 é constituído de sete subsistemas principais: primeiro estágio, segundo estágio, terceiro estágio, quarto estágio, coifa ejetável, redes elétricas e redes pirotécnicas. O 1º estágio possui quatro propulsores que são acionados simultaneamente para iniciar a decolagem do veículo. Os mesmos são fixados simetricamente ao propulsor central do segundo estágio (ver Figura 2). Próximo a cinco segundos após a queima dos propulsores as cargas pirotécnicas cortam simultaneamente toda a ligação física dos quatro motores ao segundo estágio.

Para permitir controle do veículo as tubeiras do primeiro, segundo e terceiro estágios são movimentadas. Outros subsistemas compõem cada propulsor do primeiro estágio sendo: subsistema de ignição, destruição, fixação e separação de estágios, rede elétrica e sistema de atuação da tubeira móvel (SALGADO, 2008).

Figura 2 - Indicação dos quatro estágios do VLS-1



Fonte: AEB (2012)

É integrado ao 2º estágio propulsor e os subsistemas idênticos ao do primeiro estágio, a exceção de sua tubeira móvel, mais longa adaptada ao voo em altitudes mais elevadas.

O terceiro estágio possui o propulsor também equipado com tubeira móvel. Outros subsistemas que estão no terceiro estágio são a baia de controle de rolamento e a baia de equipamentos. O primeiro possibilita um controle no eixo de rolamento do veículo durante o voo do segundo e terceiro estágios.

O quarto estágio possui tubeira fixa e um cone de acoplamento para fixação do satélite. É neste estágio que vão os equipamentos para telemetria, localização e funções de terminação de voo, “*beacon*” e antenas de telecomando e telemetria Banda-S.

Recoberta por uma proteção térmica de cortiça para a manutenção das suas propriedades mecânicas a coifa possui uma forma aerodinâmica favorável ao voo, e realiza a

proteção do satélite desde a fase de preparação do lançamento até o final da travessia da atmosfera.

2.3.1 Missão do veículo

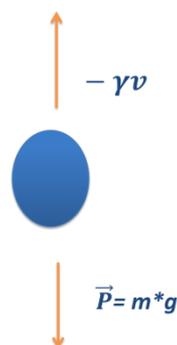
Como mencionado este item trata especificamente sobre como calcular a missão de um veículo aeroespacial. Obviamente, a escolha de colocar ou não uma carga em órbita como missão de propulsão de um foguete implica em definir se ele deverá atingir requisitos de velocidade ou de altitude. E como consequência esse requisito provoca a escolha do tipo de propelente que será utilizado.

2.3.1.1 Requisito de altitude

Para que o cumprimento desta missão seja possível, um determinado satélite deve obter uma dada velocidade horizontal que impeça que ele perca altitude e retorne ao solo.

Em caso de queda acompanhada de falha no sistema de recuperação ou não inserção deste ao sistema a variação da velocidade do corpo poderia ser deduzida seguindo a análise do diagrama da Figura 3 e o equacionamento do modelo é apresentado nas Equações 1 e 2:

Figura 3 - Diagrama simplificado para queda de um corpo



Fonte: Oliveira (2016)

$$\vec{F} = m\vec{g} - \gamma\vec{v} \quad (1)$$

$$m \frac{d\vec{v}}{dt} = m\vec{g} - \gamma\vec{v}$$

$$\frac{d\vec{v}}{dt} = \vec{g} - \frac{\gamma}{m}\vec{v} \quad (2)$$

onde $\frac{d\vec{v}}{dt}$ representa a variação de velocidade no tempo decorrido, \vec{g} é a aceleração da gravidade, m é a massa do corpo em queda e γ é o coeficiente de resistência do ar. Na Equação 1 é feita a consideração de que a resistência do ar é diretamente proporcional à velocidade pela complexidade da modelagem da força de resistência desse elemento.

Pela análise de quedas no sentido contrário ao movimento a equação para a velocidade referida pode ser deduzida a partir da 2ª Lei de Newton para massa constante (Eq. 3), com a Lei da Gravitação Universal (Eq. 4).

$$\vec{F} = m\vec{a} \quad (3)$$

$$\vec{F} = \frac{-GMm}{\vec{R}^2} = \frac{-\mu m}{\vec{R}^2} \quad (4)$$

Neste caso, \vec{F} é a força gravitacional, m é a massa do satélite, \vec{R} é o raio da órbita em relação ao centro da Terra, G é a constante da gravitação universal, M é a massa da Terra e μ é a constante gravitacional da Terra. A partir das Equações 3 e 4 pode-se deduzir a equação do vetor aceleração do satélite (Eq. 5), conhecida por equação do movimento de dois corpos:

$$\vec{R} + \left(\frac{\mu}{\gamma^3} \vec{R}\right) = 0 \quad (5)$$

Para um corpo orbitando a Terra a Equação 5 gera uma equação polar de uma cônica, na qual o módulo do vetor posição é dado em função da localização do vetor na órbita expressa na Equação 6:

$$R = \frac{\theta_M(1 - e_x^2)}{1 + e_x * \cos \sigma} \quad (6)$$

onde θ_M é o semieixo maior da órbita, e_x a excentricidade, e σ é o ângulo polar.

Para um satélite com órbitas circulares, sabe-se que a excentricidade é nula e o semieixo maior é igual a o raio. Desses fatos pode-se deduzir a Equação 7 para a velocidade de um corpo permanecer em órbita.

$$\vec{V}_c = \sqrt{\frac{\mu_t}{r}} = 631,3481 \frac{1}{\sqrt{r}} \quad (7)$$

onde \vec{V}_c é a velocidade do corpo em uma órbita circular e r é o raio da órbita.

Utilizando-se desses raciocínios, pode-se também deduzir as equações para a velocidade do corpo dentre as possíveis órbitas: elípticas, parabólicas ou hiperbólicas.

2.3.1.2 Requisito de velocidade

Quando o requisito é de velocidade a missão descrita pelo veículo é de viajar pelo espaço, saindo então do campo gravitacional da Terra. Para isto o mesmo deve atingir a velocidade de escape necessária que é definida neste caso considerando uma trajetória de escape parabólica (possui a menor energia necessária para escapar da gravidade terrestre) generalizando a distância do corpo à Terra pelo raio da órbita em que ele irá escapar. Dessa forma têm-se a velocidade de escape dada pela Equação 8:

$$\vec{V}_{esc} = \sqrt{\frac{2\mu_t}{r}} \quad (8)$$

onde \vec{V}_{esc} é a velocidade de escape. Aplicados os valores da constante gravitacional da Terra, μ_t , na Equação 8, obtêm-se a Equação 9:

$$\vec{V}_{esc} = 892,8611 \frac{1}{\sqrt{r}} \quad (9)$$

Para o evento caracterizado pela não necessidade de se cumprir a velocidade de escape da gravidade terrestre nem de velocidade circular orbital, mas somente um requisito de altitude, que pode ser atingido a baixas velocidades, deve-se então calcular o apogeu de um veículo partindo da equação do impulso específico e uma vez conhecido o ângulo de lançamento, δ_l , pode-se deduzir a aceleração final do veículo como é mostrado nas Equações 10 a 13:

$$\vec{F} = \frac{I_s W_p}{t_p} \quad (10)$$

$$(\alpha_0)_y = \vec{g} \left[\frac{\vec{F} \text{sen}(\delta_l)}{W_p} \right] \quad (11)$$

$$(\alpha_0)_x = \vec{g} \left[\frac{\vec{F} \text{cos}(\delta_l)}{W_p} \right] \quad (12)$$

$$\alpha_0 = \sqrt{(\alpha_0)_x^2 + (\alpha_0)_y^2} \quad (13)$$

onde I_s é o impulso específico, um parâmetro de eficiência do propelente utilizado, w_p é a vazão em peso do propelente, t_p é o tempo de queima do propelente, e α_0 é a aceleração no início da queima do propelente, em sua resultante e componentes.

Dessa forma, pode-se calcular a velocidade vertical do veículo após a queima do propelente, como mostrado na Equação 14, e usá-la para calcular a altitude atingida no apogeu do veículo, quando ele atinge o estado de velocidade nula, como mostra a Equação 15:

$$(\overline{V_p})_y = x * \ln\left(\frac{m_0}{m_f}\right) \text{sen}(\delta) - \vec{g}t_p \quad (14)$$

$$y_z = \frac{(u_p)_y^2}{2\vec{g}} \quad (15)$$

onde x é a velocidade efetiva de exaustão, m_0 é a massa inicial do veículo, e m_f é a massa após a queima do propelente. Pode-se então calcular o ganho obtido pela inércia do voo do foguete (fase de voo não propulsada) pelas Equações 16 e 17:

$$y_p = xt_p \left[1 - \frac{\ln\left(\frac{m_0}{m_f}\right)}{\left(\frac{m_0}{m_f} - 1\right)} \right] \text{sen}(\delta_l) - \frac{\vec{g}t_p^2}{2} \quad (16)$$

$$\Delta y = y_z - y_p \quad (17)$$

Humble et al. (1995 Apud Ribeiro, 2013), realizam comparações entre alguns parâmetros dos diferentes tipos de propelentes e suas aplicações definindo-os como de suma importância na definição do motor a ser utilizado a cada missão. Esses parâmetros são:

- ✓ Impulso específico alcançado por cada tipo de propelente, parâmetro que fornece uma estimativa da massa de propelente necessária,
- ✓ tempo de funcionamento de cada tipo de motor,
- ✓ fator de empacotamento de cada propelente,
- ✓ densidade dos gases de exaustão,
- ✓ capacidade de modulação do empuxo e de parada e reinício do funcionamento.

Dessa forma evidencia-se mais uma vez que para cada tipo de missão haverá um tipo de propelente que mais se adequa as necessidades impostas.

2.3.1.3 Balanço de Energia

O balanço energético de um foguete é obtido pela consideração de todas as parcelas de energia a serem disponibilizadas pelo foguete e as parcelas de energia a serem perdidas durante o voo. Se expressa, na grande maioria dos casos, o balanço energético em termos da velocidade final a ser alcançada.

Kaplan (1995, Apud PALMERIO, 2017) apresenta o balanço energético típico de um veículo biestágio, como se segue abaixo:

Tabela 1 - Balanço energético típico para um veículo biestágio

Primeiro estágio					
Incremento de velocidade ideal	Ganho devido à rotação da Terra	Perda de empuxo na atmosfera	Perda devida ao arrasto	Perda gravitacional	Incremento de vel. do estágio
3230 m/s	+ 350 m/s	-100 m/s	-50 m/s	-1070 m/s	+2360 m/s
Segundo estágio					
Incremento de velocidade total	Perda gravitacional	Perda por manobras	Perdas devidas à janela de lançamento	Incremento de velocidade do estágio	
6400 m/s	-460 m/s	-200 m/s	-300 m/s	5440 m/s	

Fonte: Adaptado de Palmerio (2017)

Palmerio (2017) aponta que o incremento total de velocidade de 7800 m/s é a velocidade necessária para injeção em uma órbita circular baixa. Dos valores apresentados, na Tabela 1 pode-se concluir que:

- ✓ A perda de empuxo relativa ao voo atmosférico representa 3% da energia total do primeiro estágio. Não há consequências ao segundo estágio porque voa no vácuo;
- ✓ a perda devida ao arrasto é de 1,5% da energia total do primeiro estágio;
- ✓ no segundo estágio, a perda gravitacional corresponde a 7,2% da energia do segundo estágio;
- ✓ em termos da energia perdida em relação à energia total disponível, a maior perda é ainda a gravitacional, representando 16% da energia total do veículo.

Realizada toda esta abordagem sobre os tipos de simulação, enfatizada a simulação computacional e por último debatido sobre Problemas de Valor Inicial resolvidos por qualquer plataforma de voo de foguetes, expõe-se abaixo a respeito das ferramentas de simulação já existentes e sua disponibilidade e utilização no Instituto de Aeronáutica e Espaço (IAE), órgão de pesquisa brasileiro responsável pelo projeto e desenvolvimento de veículos lançadores.

2.4 Simuladores Modernos

2.4.1 ROSI (*Rocket Simulation*)

No IAE, atualmente o ROSI é a ferramenta de simulação de trajetória utilizada nas missões de lançamento dos foguetes de sondagem brasileiros (VS-30, VSB-30 e o VS-40). Quanto a utilização com veículos guiados, o ROSI já foi utilizado para simular a trajetória do Sonda IV. Trata-se de um simulador não comercial desenvolvido em linguagem de programação FORTRAM desenvolvida no que é o atual Centro Aeroespacial Alemão (DLR- Deutsches Zentrum für Luft- und Raumfahrt), na Alemanha, na década de 1970 (SILVEIRA, 2014).

O ROSI conta com modelos dinâmicos para simulação do voo de um foguete não flexível comum, com 1 grau (utilizado para simular o movimento do veículo no trilho de lançamento), 3 graus (utilizado para simular o voo balístico do foguete no vácuo ou simular, de maneira simplificada, a reentrada atmosférica) ou 6 graus (simula o voo no espaço tridimensional) de liberdade. O IAE possui os códigos fonte deste simulador.

2.4.2 STVLS (Programa de Cálculo de Trajetórias do VLS)

Genuinamente brasileiro e também desenvolvido em linguagem FORTRAN, segundo Silva (1990) ele foi concebido no Instituto de Aeronáutica e Espaço na década de 1990 e como o próprio nome sugere especificamente para a simulação de trajetória do Veículo Lançador de Satélites (VLS-1). Dessa forma o STVLS dispõe de um modelo dinâmico com seis graus de liberdade que é utilizado desde a ignição do foguete até o instante de injeção da carga útil em órbita (SILVEIRA, 2014).

Programado para ter como saída a variação temporal de diversos parâmetros de voo inclui também a exposição das condições do veículo no instante de inserção da carga útil

em órbita. O STVLS possui ainda rotinas para cálculo dos pontos de impacto das partes alijadas do foguete: motores do primeiro, segundo e terceiro estágios e coifa.

2.4.3 *Rocket Trajectory Simulator* (RTS)

Este é o mais recente simulador desenvolvido no Brasil como ferramenta computacional para simulação de voo de veículos lançadores. Gerado como um produto de pesquisa de mestrado ele permite, por meio da determinação de diversos parâmetros de voo como a posição, velocidade e a aceleração do veículo, à realização de análises relacionadas a dinâmica de diversos tipos e missões de foguetes desde veículos não controlados aos controlados, em seis graus de liberdade.

Implementado com técnicas de programação modular, sua flexibilidade foi garantida por meio de uma biblioteca de modelos dinâmicos, dos subsistemas do veículo e ambientais, cuja combinação pode dar origem a modelos de voo de diferentes veículos lançadores. Este é hoje o mais amplo em suas capacidades de simulação dentre as plataformas de voo brasileiras.

Todos os seus códigos encontram-se a disposição do IAE. Trata-se de um *software* desenvolvido como rotinas da plataforma prioritária MATLAB® e foi escolhido como a maior referência de simulador para o desenvolvimento deste trabalho, respeitando-se os requisitos estabelecidos nesta jornada, para o qual se optou por seguir na utilização e testes da grande maioria dos diversos modelos matemáticos de dinâmica de voo que estão sendo adotados e avaliados desde o STVLS ao RTS.

2.5 Centros de Lançamento

A discussão seguida neste item se faz necessária como uma contextualização das etapas que percorrem o antes, durante e depois do cumprimento da simulação e estudos da dinâmica de voo de foguetes no Brasil, por onde os dados gerados pelo simulador são destinados e como são considerados nas etapas de concepção, projeto, preparação e lançamento.

A AEB (2017) destaca que veículos lançadores são máquinas grandes e extremamente complexas e, para sua devida operação, são necessárias várias instalações e atividades especializadas que se realizam pelos centros de lançamento. Também chamados centros espaciais, esses são locais de onde partem os foguetes com ou sem carga útil. Nesses

centros são realizados desde o lançamento em si até operações de solo como montagem e finalização da construção dos veículos lançadores.

É comum que os centros de lançamento sejam um complexo de várias instalações que se subdividem, por exemplo, em um prédio destinado à integração, onde o lançador e a carga útil são montados; um centro de controle e operações, onde o lançamento é supervisionado e controlado sendo, portanto neste setor que os dados de voo gerados pelo simulador são tratados para a efetivação do lançamento; e por último o sítio de lançamento, onde o veículo lançador é abastecido e ignitado.

O Brasil possui dois centros de lançamento mostrados na Figura 4: o Centro de Lançamento de Alcântara (CLA) no estado do Maranhão e o Centro de Lançamento da Barreira do Inferno (CLBI) localizado no Rio Grande do Norte. Por causa da proximidade com Natal o CLBI não realiza lançamentos de veículos grandes, apenas de foguetes de sondagem (AEB, 2017).

Figura 4 - Centros de Lançamento Brasileiros: (a) Centro de Lançamento de Alcântara (CLA) e (b) Centro de Lançamento da Barreira do Inferno (CLBI).

(a)



(b)



Fonte: AEB (2017)

2.5.1 Centro de Lançamento da Barreira do Inferno (CLBI)

O Centro de Lançamento da Barreira do Inferno (CLBI), criado em 12 de outubro de 1965, é direcionado a execução e prestação de apoio às atividades de lançamento e rastreamento de tecnologias aeroespaciais e de coleta e processamento de dados de suas cargas úteis, bem como executar os testes, experimentos, pesquisa básica ou aplicada e outras atividades de desenvolvimento tecnológico de interesse da Aeronáutica, relacionados com a Política da

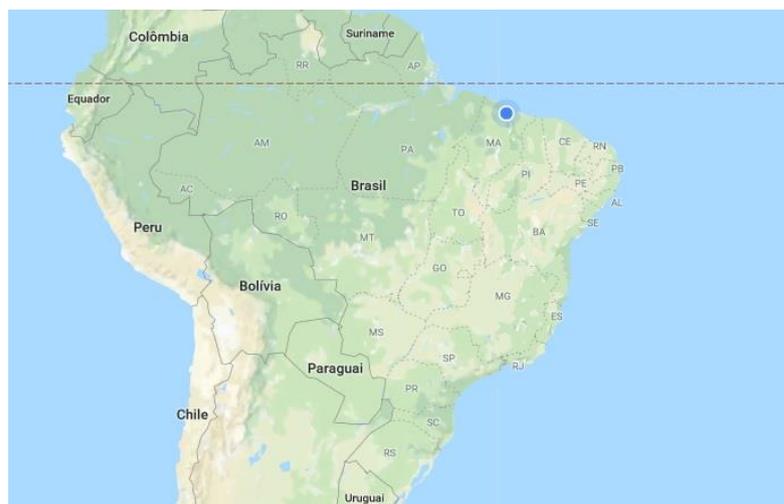
Aeronáutica para Pesquisa e Desenvolvimento e com a Política Nacional de Desenvolvimento das Atividades Espaciais (PNDAE) (AEB, 2017).

2.5.2 Centro de Lançamento de Alcântara (CLA)

Centro brasileiro criado no início da década de 80, na cidade de Alcântara no estado do Maranhão, é um dos três segmentos da Missão Espacial Completa Brasileira (MECB) e tem por missão executar as atividades de lançamento e rastreamento de engenhos aeroespaciais e de coleta e processamento de dados de suas cargas úteis, bem como a execução de testes e experimentos de interesse do Comando da Aeronáutica, relacionados com a Política Nacional de Desenvolvimento das Atividades Espaciais (PNDAE).

A sua posição geográfica, estrategicamente privilegiada, a uma posição de 2°18' ao sul da linha do Equador (ver Figura 5) reflete além das condições de segurança de voo, economias consideráveis de combustível de foguetes, possibilitando redução do custeio de seguros cobrados pelos lançamentos e configurações que elevam a um diferencial muito competitivo o CLA em escala global.

Figura 5 - Proximidade do Centro de Lançamento de Alcântara (CLA) com a Linha do Equador



Fonte: Google Maps (2018)

O estado atual das instalações e sistemas operacionais do CLA atendem, de forma irrestrita, lançamentos de sondagem e investigação científicas, contemplando, inclusive, os satélites orbitais (AEB, 2017). A referida economia de propelente citada acima faz com

que considerando dois veículos de uma mesma classe, tem-se no lançamento realizado no CLA como resultado um acréscimo na capacidade de satelização que pode atingir um patamar entre 13 e 31% superior à do segundo veículo, se lançado de outros centros localizados em latitudes mais elevadas.

Dadas as diferenças de desenvolvimento e planejamento entre o CLA e o CLBI é do primeiro que o Brasil planeja lançar o Veículo Lançador de Microssatélites (VLM). O CLA conta com uma Torre Móvel de Integração (TMI) mostrada na Figura 6 que combina o prédio de integração e o sítio de lançamento ao mesmo tempo. É nela que as diferentes partes do foguete são montadas umas sobre as outras para posteriormente serem lançadas. Esta torre possui a capacidade de mover-se para trás e assim se distanciar do foguete, fato que lhe conferi um grau de segurança elevado.

Figura 6 - Sítio de lançamento do CLA com a TMI recuada e o VLS



Fonte: AEB (2017)

Abaixo segue um resumo de como ocorrem as etapas de preparação, lançamento e pós lançamento no CLA.

2.5.2.1 Fases do VLS-1 no Centro de Lançamento

Dada à complexidade dos eventos envolvidos desde a preparação do foguete ao seu lançamento na plataforma ou trilho de lançamento seguido do recebimento dos dados de telemetria para avaliação do voo, aborda-se neste tópico um resumo das atividades que são

essenciais para o sucesso da missão de um veículo de acordo com o protocolo utilizado no CLA. Essa abordagem é essencial para a perfeita compreensão da importância dos dados gerados na fase de estudos e simulação de voo de foguetes.

2.5.2.1.1 Fase de pré-lançamento

A Figura 7 exhibe o VLS-1 V02 montado na plataforma de lançamento do CLA, com sua torre móvel de integração já afastada e expondo o veículo na mesa de lançamento, para os testes finais.

Segundo Salgado (2008) as principais atividades durante a fase de pré-lançamento ocorridas no CLA envolvem: a) integração e testes do veículo na plataforma de lançamento; b) montagem do satélite; c) alinhamento do veículo; d) carregamento e pressurização dos tanques e garrafas de gás; e e) testes e ativação de todos os subsistemas e armação dos componentes pirotécnicos de voo.

Figura 7 - VLS-1 V02 montado na mesa de lançamento do CLA



Fonte: AEB (1999, Apud SALGADO, 2008)

2.5.2.1.2 Fase de lançamento

Esta fase inicia pelo acionamento simultâneo dos quatro propulsores do primeiro estágio. Antes do fim da queima do primeiro estágio ocorre a ignição do segundo estágio e segundos depois, a separação do primeiro estágio. O terceiro estágio é acionado 1 s após o fim da queima do segundo estágio e da separação deste. No seu início ocorre a abertura e separação da coifa ejetável de proteção do satélite.

Após o fim da queima, o motor do terceiro estágio e a baia de controle de rolamento são separados do veículo. Começa então o trabalho do computador de bordo em calcular a orientação e o instante de ignição do quarto estágio. Segue-se uma manobra que visa posicionar o conjunto quarto estágio/satélite na atitude desejada (SALGADO, 2008).

Após o veículo ser orientado ele passa a ser rotacionado de 2 a 3 rotações por segundo pelo sistema impulsor de rolamento, composto de quatro micropropulsores a propelente sólido denominados de Propulsor Impulsor de Rolamento (PIR).

A estabilização é feita e então ocorre a separação da baia de equipamentos. Na sequência cumpre-se a ignição do quarto estágio, objetivando a inserção de velocidade necessária para o satélite alcançar a órbita desejada. Ao fim de sua queima dá-se a separação do satélite do quarto estágio e a consequente injeção do satélite em órbita. A Figura 8 ilustra a decolagem do VLS-1 V02 no CLA em dezembro de 1999.

Figura 8 - Início da decolagem do VLS-1 V02



Fonte: AEB (1999, Apud SALGADO, 2008)

2.5.2.1.3 Fase de pós-lançamento

As últimas atividades destacadas aqui após o lançamento são o recebimento de dados de telemetria para avaliação do voo e as atividades da operação de retorno de equipamentos e equipes para o IAE. A avaliação do lançamento ocorre pelo tratamento dos dados obtidos na telemetria desde a decolagem do veículo até a injeção do satélite em órbita bem como alijamento de seus componentes. É nesta fase que há o confronto dos valores de simulação com os reais obtidos e confirmação do cumprimento ou não do êxito da missão proposta.

Abaixo se apresenta a explanação de como o simulador através dos modelos formulados por equações diferenciais e de posse dos valores das condições iniciais de lançamento é capaz de gerar os resultados numéricos para diversos parâmetros a serem analisados em confronto com os dados da telemetria.

2.6 Rotinas de Integração Numérica

Grande parte dos problemas de engenharia apresentam elevados graus de complexidade que dificultam ou até mesmo inviabilizam suas soluções por meio de métodos analíticos (fórmula algébrica fechada). Uma técnica comum utilizada para solucionar esses tipos de problemas consiste na sua simplificação até o ponto em que a solução analítica se torna aplicável. Essa prática, porém, consiste na adoção de altos fatores de segurança devido às diversas incertezas consideradas ou resultam em soluções muito propícias a portarem erros. Com a ação combinada dos métodos numéricos e o advento da computação foi possível o desenvolvimento de métodos numéricos versáteis capazes de solucionar tais problemas (CHAPRA, 2013).

A capacidade dos métodos numéricos de resolverem um número muito grande de equações decorrentes de problemas reais transcritos em modelos matemáticos, solucionar problemas de não linearidade e geometrias complexas (cenários muito presentes dentro das mais diversas engenharias nos quais os procedimentos analíticos de cálculo não ofertam respostas mesmo que se prove que haja soluções possíveis) exemplifica muito bem a distinção de atuação e quão vasto é o universo de aplicações existentes para os métodos numéricos (AMARAL, 2012).

Vários matemáticos renomados (para os quais existem métodos que levam os seus nomes) como Newton, Laplace, Euler, Gauss, etc. se empenharam no desenvolvimento desse

ramo do conhecimento que de fato teve real intensificação de utilização após a evolução necessária da computação. A capacidade de execução de rotinas de cálculos muito extensas e cansativas para seres humanos trabalharem manualmente atrelado as possibilidades criadas em resultados gráficos exuberantes e animações com alta riqueza de fidelidade tornaram os computadores elementos de altíssima requisição dentro dos estudos e aplicações dos métodos numéricos desenvolvidos em algoritmos computacionais.

O matemático suíço Leonhard Euler, pai da Matemática Discreta, foi o responsável pela notação de função como se conhece atualmente. A mesma notação é utilizada pelas funções da programação estruturada e pelos métodos de programação orientada a objetos empregados em Java por exemplo.

Os algoritmos de Euler (primeira ordem) e de Runge-Kutta (segunda e quarta ordens) são utilizados para resolver numericamente uma equação diferencial de primeira ordem, conhecendo-se, sua condição inicial que é uma premissa dos Problemas de Valor Inicial formulados nas Equações 18 e 19 (AMARAL, 2012).

$$y' = f(y, x) \quad (18)$$

$$y(x_0) = y_0 \quad (19)$$

O que difere os três algoritmos é a atividade Iteragir: no de Método de Euler utiliza-se as Equações 20 e 21, no de Runge-Kutta de segunda ordem utiliza-se o sistema de equações composto pelas Equações 22 a 24 e no de Runge-Kutta de quarta ordem utiliza-se o sistema composto pelas Equações 25 a 29.

$$k_1 = f(x_i, y_i) \quad (20)$$

$$y_{i+1} = y_i + h_k * k_i \quad (21)$$

$$k_1 = f(x_i, y_i) \quad (22)$$

$$k_2 = f\left(x_i + \frac{h_k}{2}, y_i + h_k * k_1/2\right) \quad (23)$$

$$y_{i+1} = y_i + h_k * k_2 \quad (24)$$

$$k_1 = f(x_i, y_i) \quad (25)$$

$$k_2 = f\left(x_i + \frac{h_k}{2}, y_i + h_k \frac{k_1}{2}\right) \quad (26)$$

$$k_3 = f\left(x_i + \frac{h_k}{2}, y_i + h_k \frac{k_2}{2}\right) \quad (27)$$

$$k_4 = f(x_i + h_k, y_i + h_k k_3) \quad (28)$$

$$y_{i+1} = y_i + \frac{k_1 + k_4}{6} + (k_2 + k_3)/3 \quad (29)$$

Os mesmos algoritmos são utilizados para resolver uma equação diferencial de ordem superior e também um sistema de equações diferenciais e, para tanto, são feitas algumas modificações algébricas na equação ou no sistema que estão descritas (AMARAL, 2012).

Tais algoritmos possuem as seguintes ordens de erro: $O(h^2)$ para o de Euler, $O(h^3)$ para o de Runge-Kutta de segunda ordem e $O(h^5)$ para o de Runge-Kutta de quarta ordem.

Dentre os três algoritmos, o de Euler é o menos utilizado e o de Runge-Kutta de quarta ordem o mais utilizado (como é sugerido pela própria precisão do método) na resolução numérica de equações diferenciais na engenharia e foi baseado nele que se implementou a rotina computacional que resolve os problemas de valor inicial do presente simulador de voo de foguetes foco deste trabalho adotando um incremento (h_k) de 10^{-6} como critério norteador exigido para precisão de funcionamento do algoritmo.

Para o desenvolvimento não só desta rotina, mas também de toda a plataforma foi necessária à utilização de uma linguagem de programação que agregasse os recursos essenciais para o cumprimento dos requisitos de compilação do *software* independente e da portabilidade em relação aos sistemas operacionais exigidos. Para esta finalidade elegeu-se a linguagem de programação Java apresentada no tópico seguinte. O devido domínio dos conceitos apresentados nos tópicos abaixo é de suma importância no desenvolvimento deste simulador.

2.7 Linguagem de programação Java

O interesse em Java para esta aplicação reside principalmente na neutralidade dessa linguagem em relação à plataforma. Outras linguagens, como C/C++ ou Pascal, exigem a recompilação do código fonte toda vez que se precisa migrar de plataforma. No caso de Java, o mesmo código já compilado pode executar em mais de uma plataforma sem a necessidade de alterações do código fonte e recompilação.

A linguagem Java consegue esse feito utilizando uma máquina virtual independente da plataforma, a qual permite que um programa desenvolvido, por exemplo, em

um computador Samsung com sistema operacional Microsoft Windows 10 possa ser executado sem modificações em uma máquina Lenovo que utiliza o sistema operacional Linux.

De acordo com Silva (2007) a linguagem Java tem como características:

- ✓ O paradigma de orientação a objetos,
- ✓ Ser distribuída. Java foi uma linguagem projetada desde o início para computação distribuída em uma rede heterogênea de computadores.
- ✓ Ser ao mesmo tempo compilada e interpretada. Os programas fonte em Java (extensão .java) são compilados para um formato binário de código conhecido como *bytecode* (extensão .class) que é independente de plataforma. O *bytecode* será então executado em outra máquina de forma interpretada.
- ✓ Ser uma linguagem neutra em relação à arquitetura, o que garante portabilidade entre plataformas,
- ✓ Ser multitarefa. Java suporta o conceito de threads permitindo a execução simultânea de diversos segmentos de código,
- ✓ Ser dinâmica. A ligação (*linking*) do programa com as bibliotecas de terceiros são executadas dinamicamente, ao contrário de outras linguagens que precisam ser ligadas estaticamente,
- ✓ Ser segura. Devido à característica distribuída de Java, ela foi pensada para incorporar diversos recursos de segurança, raramente encontrada em outras linguagens. Podem se especificar quais permissões de acesso o programa em Java terá em relação à rede ou disco rígido,
- ✓ Ser simples quando comparada a C++. A linguagem Java apesar de ser derivada da linguagem C++, evitou características problemáticas dessa linguagem que causavam confusão e falta de legibilidade como, por exemplo, herança múltipla,
- ✓ Ser compacta. A maioria das máquinas virtuais e programas em Java ocupam pouco espaço de memória. Isso permite que Java possa funcionar em máquinas com poucos recursos como celulares e palmtops,
- ✓ Ter alto desempenho. O emprego de compiladores *just in time* (JIT), permite a compilação de partes críticas do código (*hotspots*) e alcançar velocidades de execução próximas a linguagens compiladas como C.

O desenvolvimento de aplicações em Java é realizado através da manipulação de classes. Uma aplicação Java possui a seguinte estrutura:

```
class NomeDaClasse {
    // Atributos
    // Métodos
    public static void main ( String[] args ) {
        //corpo do programa
    }
}
```

2.7.1 Biblioteca JFreeChart

O segundo Gilbert (2018) *JFreeChart* é uma biblioteca de gráficos Java totalmente gratuita que facilita aos desenvolvedores exibirem gráficos de qualidade profissional em seus aplicativos. Pela sua própria definição o extenso conjunto de recursos do *JFreeChart* inclui:

- ✓ Uma API (*Application Programming Interface*, do português “Interface de Programação de Aplicativos”) consistente e bem documentada, suportando uma ampla variedade de tipos de gráficos;
- ✓ um design flexível que é fácil de estender e direciona os aplicativos do lado do servidor e do lado do cliente;
- ✓ suporte para muitos tipos de saída, incluindo componentes Swing e JavaFX, arquivos de imagem (incluindo PNG e JPEG) e formatos de arquivos gráficos vetoriais (incluindo PDF, EPS e SVG);

Foi desenvolvida para ser utilizada em aplicações *desktop*, *applets*, *servlets* e JSP. O *JFreeChart* é *open source* e *free software* (biblioteca de código aberto e livre). Nasceu no início dos anos 2000 e é distribuído sob os termos da GNU Lesser General Public Licence (LGPL), que permite o seu uso em aplicativos proprietários. Os tipos de gráficos por ela suportados são: a) gráficos de pizza (2D e 3D); b) gráficos de barras (horizontais e verticais, regulares e empilhados); c) gráficos de linhas; d) gráficos de dispersão e) gráficos de séries temporais, f) gráficos *high-low-open-close*; g) gráficos *candlestick*; h) gráficos de Gantt; e muito mais (Gilbert, 2018).

Além do fato de ser livre, possui a vantagem de ser bastante robusta e flexível. Consegue dar saída em JPG, PNG, SVG, EPS e até mesmo exibir em componentes Swing. Possui como classe principal a *org.jfree.chart.JFreeChart*, ela representa um gráfico e pode assumir vários formatos (torta, barra, pontos, linhas etc.).

A implementação direta com essa classe pode ser considerada um tanto quanto complexa pela definição dos seus próprios criadores e dessa forma opta-se por desenvolver através de uma segunda classe que porta uma série de métodos estáticos que facilitam o trabalho de criação dos gráficos: a *org.jfree.chart.ChartFactory*.

Exemplo de uso desse método para salvar em um arquivo PNG:

```
OutputStream arquivo = new FileOutputStream("grafico.png");  
ChartUtilities.writeChartAsPNG(arquivo, grafico, 550, 400);  
fos.close();
```

Para o desenvolvimento do simulador proposto em linguagem Java, no entanto se faz necessário um ambiente de programação chamado IDE (Ambiente de Desenvolvimento Integrado) que agrega tanto um editor de texto quanto o compilador da linguagem e recursos de otimização de desenvolvimento. O IDE selecionado para este propósito foi NetBeans IDE comentado abaixo.

2.7.2 Ambiente de Desenvolvimento Integrado NetBeans

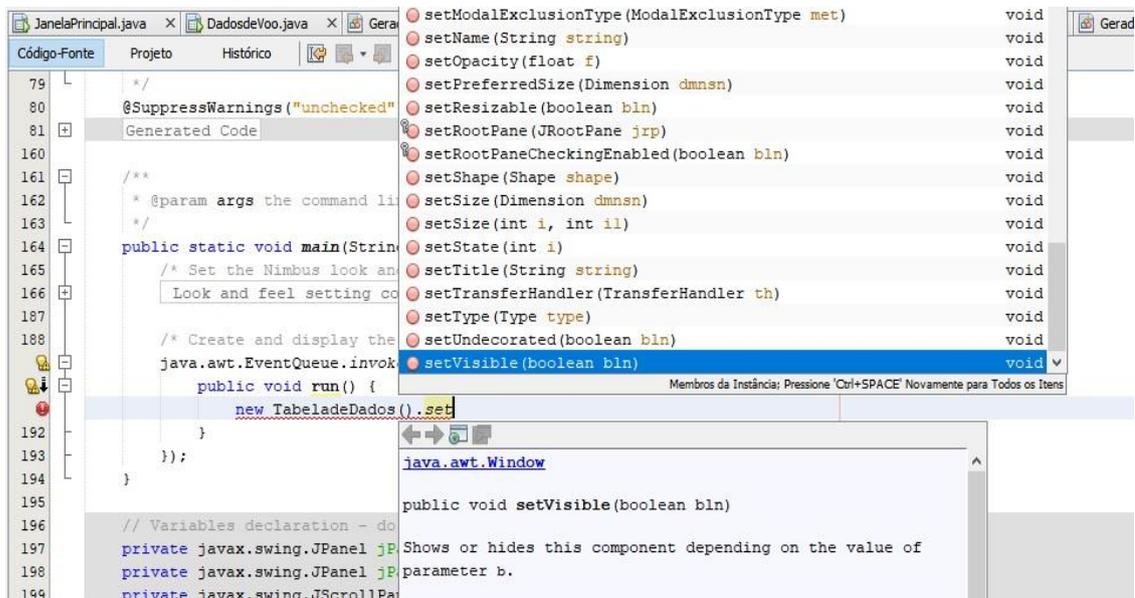
De acordo com a definição do próprio site oficial do NetBeans IDE (2018) ele permite o desenvolvimento rápido e fácil de aplicações *desktop* Java, móveis e Web e também aplicações HTML5 com HTML, JavaScript e CSS. Trata-se de uma plataforma gratuita e de código-fonte aberto, com uma grande comunidade de usuários e desenvolvedores em todo o mundo.

Características que fazem o IDE tão famoso e utilizado incluem o fato dele trabalhar com recuo de linhas automático, associar palavras e colchetes e realçar códigos-fonte por sintática e semanticamente. Também fornece modelos de código, dicas de codificação e ferramentas de refatoração.

Além do editor suportar várias linguagens, como Java, C/C++, XML, HTML, PHP, Groovy, JavaScript e JSP ele é extensível o que significa que ele aceita adição a suporte

para outras linguagens. A Figura 10 mostra facilidades (auto completar e descrição de funções) de programação ofertadas pelo NetBeans na construção de uma das classes do projeto deste simulador.

Figura 9 – Desenvolvimento de uma classe do simulador no NetBeans



Fonte: Autores (2018)

Dentro deste contexto, mais uma ferramenta será de grande necessidade para o desenvolvimento da aplicação proposta neste trabalho uma vez que o simulador precisa guardar os dados gerados com a simulação para futuro acesso, fato que em linguagens de programação é conceituado como persistência de dados e é o assunto do próximo tópico. O suporte a banco de dados será implementado pelo *software* MySQL WorkBench.

2.7.3 Persistência de Dados em Java por banco de dados

JDBC significa *Java Database Connectivity* (conectividade com bancos de dados em Java). É uma API para a Linguagem Java que define como um cliente pode acessar um banco de dados para realizar sobre ele funções como inserção, deleção e atualização de valores.

A API do Java, o JDBC, permite compartilhar uma única linguagem entre as diversas aplicações. Além disso, provê um conjunto de interfaces com o objetivo de criar um ponto em comum entre as aplicações e os mecanismos de acesso a um banco de dados (CLARO e SOBRAL, 2008).

Lançada em 19 de fevereiro de 1997, como parte do *Java Development Kit 1.1* esta API é focada em bancos de dados relacionais. Segundo Claro e Sobral (2008) uma API ao nível do SQL, caso do JDBC, permite construir sentenças SQL e colocá-las dentro das chamadas da API Java. Assim, JDBC permite uma transmutação transparente entre o mundo do Banco de Dados e as aplicações JAVA. Os resultados das bases de dados são retornados através de variáveis do Java e problemas com acessos são recebidos pelas exceções, que contém uma classe especial chamada de *SQLException*.

O banco de dados foi construído em linguagem SQL (*Structure Query Language* – Linguagem de Consulta Estruturada) que é atualmente a linguagem mais popular para realizar acesso e gerenciamento do conteúdo armazenado em banco de dados. O sistema gerenciador utilizado para implementação e comunicação do banco foi o MySQL Workbench da companhia Oracle *Corporation*, selecionado principalmente por possuir uma versão gratuita (*MySQL Community Edition* - Edição da Comunidade) de alta performance e amplamente popular entre desenvolvedores de *softwares*. Para tanto, ele conta tanto com o servidor quanto com uma interface gráfica cliente.

No MySQL Workbench, pode-se executar várias atividades em uma interface altamente intuitiva como: a) modelar, criar e manter a base de dados através do seu ambiente integrado; b) consultas SQL, e c) administrar o sistema.

A JDBC se encontra nos pacotes `java.sql`, onde está a `javax.sql`. Entre as classes e interfaces mais importantes podem-se citar:

- 1 - classe *DriverManager*: gerencia o acesso a múltiplos bancos de dados;
- 2 - interface *Driver*, que representa um driver JDBC;
- 3 - interface *Connection*, que representa uma conexão com o banco de dados;
- 4 - interfaces *Statement*, *PreparedStatement* e *CallableStatement* que permitem a execução de comandos SQL e *stored procedures*;
- 5 - interface *ResultSet* que representa os dados obtidos do banco;
- 6 - classe *Types* que define os tipos de dados JDBC;
- 7 - interface *DatabaseMetaData* que obtém informações sobre as capacidades do banco que são disponibilizadas pelo driver em uso.

Uma sessão típica de acesso a um banco de dados segue o padrão abaixo:

```
// estabelecimento da conexão
Connection con = DriverManager.getConnection(
    url, // caminho de acesso ao SGBD
```

```
        userid, // login
        password // senha );
...
// uso da conexão para atividades de insert, select, update, etc.
...
// encerramento da conexão
con.close();
```

O objeto *Statement* é utilizado para a execução de comandos SQL através do driver da base de dados suportando os métodos que causam alteração na base de dados e operações de consulta. Os comandos SQL são passados ao objeto *Statement* como *strings*.

```
String createString = "create table Dados (" + "Nome_ID INTEGER,"
                    + "Nome_nome VARCHAR(30),"
                    + ")";
...
try {
    Statement stmt = con.createStatement();
    stmt.executeUpdate(createString);
    stmt.close();
} catch(SQLException ex) { ... }
```

3 MODELOS MATEMÁTICOS PARA SIMULAÇÃO DE VOO DE FOGUETES

A dinâmica de voo trata de todos os movimentos que o veículo cumpre, desde o momento de sua ignição, até seu retorno à Terra, quando a missão inclui a reentrada. Em essência, o objetivo das atividades em dinâmica de voo de foguetes consiste em projetar e executar o movimento do CG (centro de gravidade) desde a decolagem até a separação da carga útil/paga. O sistema de controle de atitude é utilizado quando se deseja precisão na evolução da trajetória ou quando manobras são necessárias. O projeto da trajetória requer apenas a determinação do movimento do CG do foguete. Na prática, efeitos externos e internos atuam sobre o foguete influenciando em sua trajetória, o que obriga a simulação do movimento do corpo do foguete, considerado rígido, tanto na fase de projeto quanto na fase de reconstituição da trajetória real executada.

O desenvolvimento deste capítulo apresenta os diversos modelos dinâmicos implementados no simulador sendo estes a compilação dos modelos encontrados e retirados principalmente dos documentos e materiais didáticos do INPE (Instituto Nacional de Pesquisas Espaciais), IAE (Instituto de Aeronáutica e Espaço) e bem como da literatura de Silveira (2014), onde se vale ressaltar que este último propõe modelos fruto de intensas pesquisas bibliográficas e da análise de códigos e manuais de diversas ferramentas de simulação e por isso são altamente refinados e adequados à utilização nesta obra.

Para a abordagem dos referidos modelos é necessário previamente a explanação sobre a definição dos modelos relacionados ao ambiente, que portam influência direta sobre os esforços que atuam no veículo durante o voo, bem como o sistema de referência sob o qual estão apoiados os desenvolvimentos dos modelos dinâmicos e por fim os modelos dos subsistemas dos foguetes.

3.1 Modelos do ambiente

O ponto de partida da modelagem do ambiente se dá pelas definições necessárias a respeito de como considera-se a Terra em função de um sistema referencial e também quanto a sua geometria e distribuição de massa.

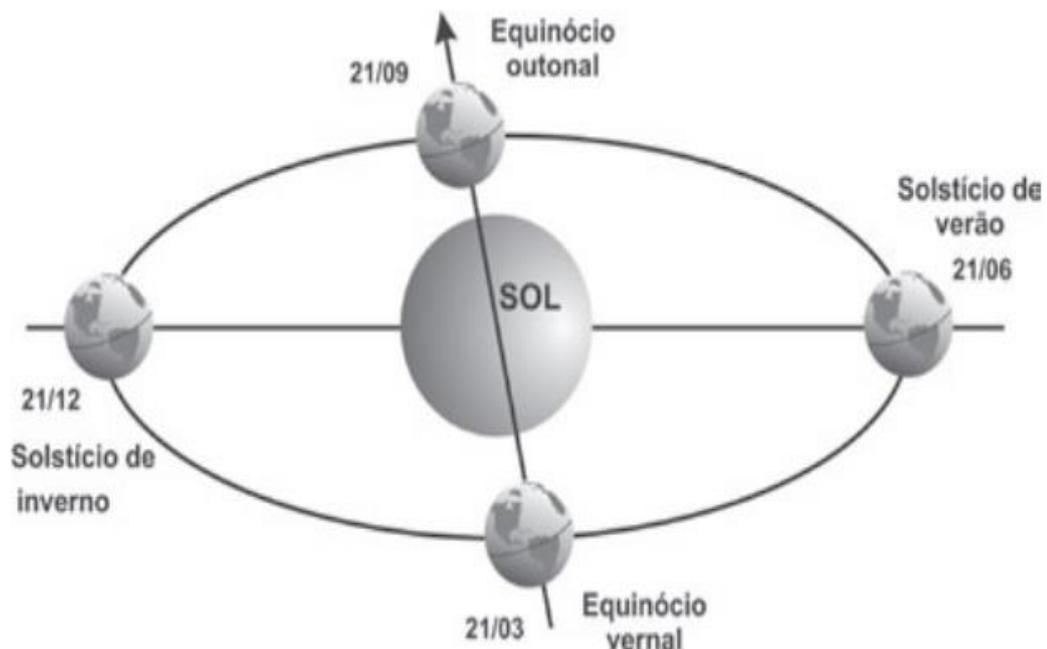
A atitude ou orientação de um corpo no espaço é definida por um conjunto de parâmetros que permitem, de forma unívoca, correlacionar num instante de tempo qualquer um sistema de coordenadas fixo ao corpo a outro sistema supostamente fixado a uma base (Carrara, 2012). Em geral assume-se que este último seja inercial ou quase inercial, o que

significa que seu movimento, em relação a um sistema verdadeiramente inercial, seja desprezível quando comparado com o movimento próprio do corpo. Na contrapartida, a compreensão do que se chama de sistema inercial requer que se façam concessões ao rigor matemático. De fato, considera-se que um sistema inercial seja aquele que não apresente movimento de rotação.

Carrara (2012) demonstra que é possível adotar que o momento angular de um corpo não sujeito a torques e em movimento de rotação tende a permanecer imóvel inercialmente do que se retira a possibilidade de se estabelecer então direções inerciais baseadas em eixos de rotação. Porém, na prática é muito difícil encontrar corpos efetivamente livres de torques.

Através desse princípio realiza-se a adoção do eixo de rotação da Terra como uma das direções inerciais (mostrado na Figura 11) mesmo sabendo que ele efetua um movimento cônico ao redor de um eixo perpendicular à órbita da Terra ao redor do Sol em um período de cerca de 20 mil anos. Entretanto, este se move lentamente o suficiente para ser utilizado como fins de referência de movimentos com a duração do voo de um foguete. Sendo este segundo refém de movimentos em função de outras estrelas e do restante da galáxia e o mesmo raciocínio se repete sucessivamente aos demais astros que se considera na sequência.

Figura 10 - Direção inercial de referência



Fonte: Palmerio (2017)

Entretanto, atribui-se aos quasares distantes a melhor aproximação de um sistema inercial real, pois, embora quase nada possa ser confirmado ainda a respeito dos seus movimentos angulares transversais sabe-se que estes são muito pequenos a ponto de poderem ser desprezados.

Dessa forma então se adotou a consideração de que o sistema geocêntrico celeste é inercial, uma vez que seus eixos coincidem com o eixo de rotação da Terra e com a direção da interseção do plano do equador com o plano da eclíptica (projeção sobre a esfera celeste da trajetória aparente do Sol observada a partir da Terra).

Outra consideração ainda importante a ser feita sobre a Terra é quanto ao seu tipo de modelo de geometria da superfície e distribuição de massa a ser adotado neste trabalho. Existem 3 modelos que descrevem as características da Terra (geometria da sua superfície e a aceleração gravitacional) que interferem no movimento dos foguetes. Esses modelos são:

a) **Modelo da Terra plana:** o planeta é considerado como uma superfície plana e inercial na qual a aceleração gravitacional pode ser considerada constante ou variável com a altitude tendo direção normal à superfície e apontando para baixo.

b) **Modelo da Terra esférica com distribuição de massa homogênea:** o planeta é considerado uma esfera com o vetor aceleração gravitacional apontando para o centro da Terra.

c) **Modelo da Terra esférica com distribuição de massa axissimétrica:** o planeta é considerado um esferoide oblato gerado com a rotação de uma elipse em torno de seu semieixo menor e nesse caso a aceleração gravitacional não aponta para o centro da Terra e na verdade é composto de uma componente radial e outra tangencial.

Um modelo de excelente precisão (não adotado nesse trabalho em função da sua complexidade de implementação ao Simulador) considera o Sistema Geodésico Mundial 1984 (WGS84) uma vez que a Terra não apresenta geometria esférica perfeita e pode-se aproximar a um modelo de um elipsoide.

Neste modelo, o geóide pode ser aproximado por um elipsoide de revolução em torno de seu eixo menor, o qual coincide com o eixo de rotação da Terra. Adota-se um elipsoide padrão com semieixo maior de 6.378,137 km e semieixo menor de 6.356,752 km (BRASILEIRO, 2007). A vertical local no elipsoide não passa pelo centro de massa da Terra e forma com o plano do equador um ângulo chamado latitude geodésica ou latitude geográfica, L. (BATE, MUELLER e WHITE, 1971).

A mais importante fonte perturbadora das órbitas de objetos como satélites, por exemplo, é a força devido ao campo gravitacional da Terra. O fato da distribuição de massa

não ser perfeitamente simétrica, propicia componentes de força normal ao raio vetor que une o satélite ao centro da Terra. Por estas componentes, a órbita real desvia da órbita kepleriana. A perturbação, devido ao efeito do campo gravitacional da Terra, decresce com o aumento da distância entre o satélite e a Terra (variações seculares ou perturbações seculares) (CORNELISSE, 1979, Apud BRASILEIRO, 2007).

Por questões de simplificações dos modelos matemáticos aqui utilizados se fez a adoção do modelo da Terra com superfície plana onde se pode escrever a aceleração da gravidade e do vetor aceleração gravitacional respectivamente pelas Equações 30 e 31:

$$g = g_0 \left(\frac{r_0}{r_0 + h} \right)^2 \quad (30)$$

$$[\vec{g}]^V = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \quad (31)$$

onde $g = 9,806 \text{ m/s}^2$ (aceleração ao nível do mar) e $r_0 = 6378,14 \text{ km}$ (raio médio terrestre).

O segundo ponto de partida que se deve descrever aqui está associado à presença do vento e sua capacidade de modificar a velocidade dos foguetes em relação ao ar atmosférico provocando diretamente esforços aerodinâmicos aos mesmos.

Segundo o IAE (2017) um modelo bastante simples de vento consiste em considerar conhecida a velocidade do vento em relação à superfície da Terra em função da altitude. Assim, as componentes da velocidade do vento podem ser obtidas diretamente por meio de interpolação e foi essa técnica que se adotou no desenvolvimento do simulador.

O terceiro e último modelo a se definir relacionado ao ambiente é o modelo da atmosfera utilizado para determinar as propriedades atmosféricas no ponto onde o veículo se encontra. Foi implementado no simulador o modelo atmosférico *U.S. Standard Atmosphere* 1976. Segundo Silveira (2014) este considera o ar um gás ideal e em altitudes menores que 86 km o ar é adotado como homogêneo e considerado estar em equilíbrio hidrostático, possuindo uma massa molar constante. Assim a atmosfera é dividida em camadas onde em cada uma delas a temperatura varia linearmente em função da altitude geopotencial, como dado pela Equação 32:

$$T_{atm} = T_{atm,b} + L_b(h_{gp} - h_{gp,b}) \quad (32)$$

onde $T_{atm,b}$ é a temperatura atmosférica na base da camada considerada, L_b é a taxa de variação da temperatura em relação à altitude geopotencial na camada considerada, h_{gp} é a altitude geopotencial do veículo e $h_{gp,b}$ é a altitude geopotencial da base da camada. A relação entre a altitude geopotencial e a altitude geométrica é definida pela Equação 33:

$$h_{gp} = \frac{r_0 h}{r_0 + h} \quad (33)$$

onde h é a altitude geométrica do veículo. A pressão atmosférica é expressa pelas seguintes relações (Equações 34 e 35) de acordo com o valor de L_b :

$$p_{atm} = p_{atm,b} \left(\frac{T_{atm,b}}{T_{atm}} \right)^{\left(\frac{g_0 M_{atm}}{R_{atm} L_b} \right)} \text{ se } L_b \neq 0 \quad (34)$$

$$p_{atm} = p_{atm,b} e^{\left(\frac{-g_0 M_{atm} (h_{gp} - h_{gp,b})}{R_{atm} T_{atm,b}} \right)} \text{ se } L_b = 0 \quad (35)$$

Para $R_{atm} = 287,0531 \text{ J/KgK}$ (constante particular do ar atmosférico), $T_{atm} = 273 \text{ K}$ (temperatura atmosférica adotada) e M_{atm} é o valor da massa molar do ar atmosférico.

Acima de 86 km ocorre a difusão e o transporte vertical de espécies individuais dos gases constituintes do ar o que faz a hipótese de equilíbrio não ser mais válida e o modelo torna-se mais complexo. Para este caso, deu-se preferência por adotar uma tabela com os valores da temperatura, pressão, densidade atmosférica e massa molar do ar, em função da altitude geométrica.

Nessa sequência, a densidade atmosférica e a velocidade do som são dadas respectivamente pelas Equações 36 e 37:

$$\rho_{atm} = \frac{p_{atm} M_{atm}}{R_{atm} T_{atm}} \quad (36)$$

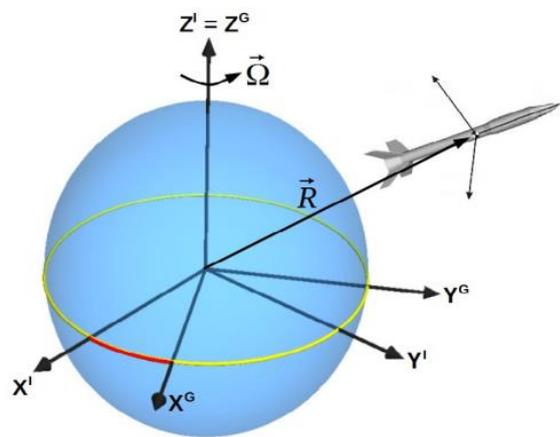
$$V_{som} = \sqrt{\frac{\gamma R_{atm} T_{atm}}{M_{atm}}} \quad (37)$$

onde γ é a razão entre os calores específicos do ar à pressão constante e avolume constante.

3.2 Modelos dinâmicos e cinemáticos

O conjunto de equações que serão listadas abaixo compõe o modelo dinâmico para movimentos de foguetes com avaliação das derivadas no sistema de referência do corpo (fato que permite que as equações sejam decompostas em componentes ao longo dos eixos desse sistema) e também partindo da adoção da velocidade relativa (em relação à superfície da Terra, ver Figura 12) em substituição da velocidade inercial. Nesse modelo são quantificadas 12 variáveis de estado.

Figura 11 - Relação entre velocidade inercial e velocidade relativa



Fonte: IAE (2017)

A Equação 38 trata do movimento translacional do veículo e a Equação 39 do movimento rotacional expressas por:

$$\left. \frac{\delta \vec{V}_{rel}}{\delta t} \right|_B = \frac{\vec{F}_{total}}{M} - \vec{\Omega} \times (\vec{\Omega} \times \vec{R}_{gc}) - (\vec{w} + \vec{\Omega}) \times \vec{V}_{rel} \quad (38)$$

$$I \cdot \left. \frac{\delta \vec{w}}{\delta t} \right|_B = \vec{M}_{total} - \vec{w} \times (I \cdot \vec{w}) \quad (39)$$

Nestas expressões considera-se $\vec{\Omega}$ como sendo a rotação da Terra, \vec{w} a rotação do foguete, \vec{V}_{rel} a velocidade em relação à superfície do planeta e \vec{F}_{total} a soma das forças aerodinâmica, gravitacional e de empuxo que agem sobre o foguete.

A posição linear (\vec{R}_{gc}), é definida pelo módulo do raio geocêntrico (R_{gc}), pela longitude (μ) e pela latitude geocêntrica (λ_{gc}) sendo as relações de variações desses parâmetros mostradas respectivamente pelas Equações 40 a 42:

$$R_{gc} = -V_{relz}^V \quad (40)$$

$$\mu = \frac{1}{R_{gc} \cos \lambda_{gc}} V_{rely}^V \quad (41)$$

$$\lambda_{gc} = \frac{1}{R_{gc}} V_{relx}^V \quad (42)$$

As relações para o cálculo da atitude local são dadas pelas Equações 43 a 45:

$$\theta = q_l \frac{\cos \phi}{\sin \psi} - r_l \frac{\sin \phi}{\cos \psi} \quad (43)$$

$$\psi = q_l \sin \phi + r_l \cos \phi \quad (44)$$

$$\phi = p_l - q_l \cdot \tan \psi \cdot \cos \phi + r_l \cdot \tan \psi \cdot \sin \phi \quad (45)$$

onde p_l, q_l e r_l são as componentes da velocidade angular local do veículo e ϕ, θ, ψ são respectivamente os ângulos de guinada, arfagem e rolamento que descrevem a atitude do foguete.

A dedução destes modelos bem como explicações sobre as considerações matemáticas necessárias para as suas obtenções são apresentadas com maiores detalhes na obra de Silveira (2014).

3.2.1 Modelo para movimento do foguete a partir do trilho de lançamento

Para foguetes lançados a partir de trilhos, como os foguetes de sondagem brasileiros, durante os instantes iniciais do voo o foguete permanece fisicamente conectado ao trilho, de forma que seu movimento pode acontecer apenas ao longo do trilho.

Nesse caso, a única componente não nula da força sobre o foguete atua ao longo do eixo X_B , de forma que as equações dinâmicas do movimento translacional são apresentadas respectivamente pelas Equações 46 e 47:

$$\left. \frac{\delta \vec{V}_{relx}}{\delta t} \right|_B = \frac{\vec{F}_{total}}{M} - \vec{\Omega} \times (\vec{\Omega} \times \vec{R}_{gc}) - (\vec{w} + \vec{\Omega}) \times \vec{V}_{rel} \quad (46)$$

$$\left. \frac{\delta \vec{V}_{rely}}{\delta t} \right|_B = \left. \frac{\delta \vec{V}_{relz}}{\delta t} \right|_B = 0 \quad (47)$$

Além disso, a velocidade angular do foguete se mantém constante e igual a velocidade angular da Terra, de modo que a equação da dinâmica rotacional é definida pela Equação 48:

$$\left. \frac{\delta \vec{w}}{\delta t} \right|_B = 0 \quad (48)$$

As equações cinemáticas são as mesmas já apresentadas para 6 GDL.

3.2.2 Modelo para movimento do foguete a partir da rampa de lançamento

Esse modelo é utilizado para simular o lançamento de veículos a partir de rampa de lançamento. Ele simula o comportamento do veículo durante o intervalo de tempo entre a ignição do (s) motor(es) do primeiro estágio (início da simulação) e o momento em que o veículo perde o contato com a rampa. Esse intervalo pode ser devido ao fato de a força de empuxo não ser suficiente para vencer o peso do veículo ou ao fato de o veículo estar mecanicamente preso à rampa. Nesse caso, a velocidade relativa é igual a zero e a velocidade angular é constante e igual à velocidade angular da Terra, resultando nas seguintes equações dinâmicas (Equações 49 e 50):

$$\left. \frac{\delta \vec{V}_{rel}}{\delta t} \right|_B = 0 \quad (49)$$

$$\left. \frac{\delta \vec{w}}{\delta t} \right|_B = 0 \quad (50)$$

As equações cinemáticas são as mesmas já apresentadas para 6 GDL.

3.3 Modelos dos subsistemas de um foguete

Após a determinação das equações do movimento, o próximo passo é modelar os esforços que atuam sobre o foguete. Uma vez que está sendo utilizado o sistema do corpo para escrever as equações do movimento, precisam-se conhecer os esforços ao longo dos eixos desse sistema (IAE, 2017).

Vários podem ser os subsistemas que integram um foguete, fato que auxilia na quantificação dos esforços que atuam no mesmo e dessa forma, neste trabalho, contou-se com os seguintes subsistemas:

1 – Subsistema estrutural: subsistema encarregado de portar informações a respeito da massa do foguete que é dividida em massa estrutural e massa de propelente. Como durante as várias fases de voo esse veículo perde massa pela possível separação de seus estágios e também pela queima do combustível os modelos adotados são capazes de simular a forma como ocorre essa variação de massa, momentos e produtos de inércia e também da posição do seu centro de massa.

Neste simulador, calcularam-se todas essas variações através de interpolações de valores previamente tabelados para cada veículo.

2 – Subsistema propulsivo: modelos que preveem o comportamento dos conjuntos propulsores do foguete através da determinação das forças e momentos gerados por esses conjuntos ao longo da missão do veículo. Diversos são os tipos de conjuntos propulsores empregados em veículos lançadores compreendendo conjuntos com diferentes números de motores, a propelente sólido ou líquido, motores com empuxo vetorizado ou não, etc. (IAE, 2017).

Para a determinação dos seus valores foi adotada aqui a estratégia comum a foguetes a propelentes sólidos que é a aplicação da curva de empuxo previamente conhecida para esses tipos de foguetes. Na sequência, para um dado instante de tempo do voo será

possível por interpolação descobrir qual o valor desse empuxo associado. Mesmo no caso de outros tipos de propelentes é comum ser conhecida o valor dessa força reatora para uma determinada pressão atmosférica de referência

A expressão de correção do empuxo pela pressão atmosférica local é dada pela Equação 51:

$$T_{cor} = T_{ref} + (p_{ref} - p_{atm})A_e \quad (51)$$

adotando que p_{ref} seja a pressão de referência e A_e a área de saída da tubeira do foguete.

Se esse empuxo ocorre em uma tubeira fixa então seu vetor associado é expresso como (Equação 52):

$$[F_T]^B = \begin{bmatrix} T_{cor} \\ 0 \\ 0 \end{bmatrix} \quad (52)$$

A partir desse parâmetro e do vetor posição de aplicação do empuxo em relação ao centro de massa do foguete (r_e) pode-se calcular os dois momentos ao qual estão sujeitos os foguetes: o momento propulsivo (Eq. 53) e o momento de Coriolis (Eq. 54). Este último momento a ser calculado é função tanto da vazão mássica (\dot{m}) quanto da variação do momento de inércia (I) do foguete que são fatores obtidos na fase de projeto de qualquer tipo de foguete.

$$\vec{M}_T = [\vec{F}_T] \times r_e \quad (53)$$

$$\vec{M}_c = -\frac{dI}{dt}\vec{\omega} + m\vec{r}_e \times (\vec{\omega} \times \vec{r}_e) \quad (54)$$

O parâmetro r_e é calculado por (Eq. 55):

$$r_e = r_{T/n} - r_{cm/n} \quad (55)$$

3 – Subsistema aerodinâmico: modelos que determinam forças e momentos aerodinâmicos que atuam nos foguetes em suas trajetórias. Como parâmetros de entrada

desses modelos são utilizadas tabelas de coeficientes específicos para cada veículo que relacionam os esforços aerodinâmicos ao seu regime de voo.

Como os esforços aerodinâmicos são dependentes da velocidade do foguete (\vec{V}_{rel}) em relação à velocidade do vento (\vec{W}) pode-se escrever a Equação 56 da seguinte forma:

$$\vec{V}_{atm} = \vec{V}_{rel} - \vec{W} \quad (56)$$

Esse parâmetro nos auxilia para o cálculo das funções do Número de Mach (Eq. 57), ângulo de ataque (Eq. 58), ângulo de derrapagem (Eq. 59) e pressão dinâmica (Eq. 60) de um foguete.

$$Mach = \frac{V_{atm}}{V_{som}} \quad (57)$$

$$\alpha = \tan^{-1} \left(\frac{V_{atm,z}}{V_{atm,x}} \right) \quad (58)$$

$$\beta = \tan^{-1} \left(\frac{V_{atm,y}}{\sqrt{V_{atm,x}^2 + V_{atm,z}^2}} \right) \quad (59)$$

$$p_{din} = \frac{\rho_{atm} V_{atm}^2}{2} \quad (60)$$

Em mecânica dos fluidos o Número de Mach tem sua importância por ser utilizado como um parâmetro para determinar tipos de regimes de voo de um veículo, como são mostrados abaixo:

- ✓ Regime subsônico: estado de voo onde o veículo se desloca com velocidade abaixo da velocidade do som. Quantidade significativa dos aviões civis voam neste regime.
- ✓ Regime transônico: neste regime o veículo se desloca com velocidade próxima a velocidade do som (Mach 0,9 e 1,1). Como este regime é turbulento, o ideal é ultrapassá-lo rapidamente para se evitar danos aos equipamentos embarcados (PALMERIO, 2017).

- ✓ Regime supersônico: regime de voo compreendido entre Mach 1,1 e Mach 1,5. A maioria dos lançadores de foguetes trabalha nessa região.
- ✓ Regime hipersônico: regime para o qual o número de Mach ultrapassa o valor 5 (Mach >5) e é atingido por corpos durante a reentrada na atmosfera terrestre.

Os ângulos de ataque e de derrapagem são necessários para o cálculo das componentes (Equações 62 a 64) do vetor força aerodinâmica expresso na Equação 61:

$$[F_A]^B = \begin{bmatrix} F_{Ax} \\ F_{Ay} \\ F_{Az} \end{bmatrix} \quad (61)$$

$$F_{Ax} = -C_A \rho_{din} S_{ref} \quad (62)$$

$$F_{Ay} = -C_{yB} \beta \rho_{din} S_{ref} \quad (63)$$

$$F_{Az} = -C_{N\alpha} \alpha \rho_{din} S_{ref} \quad (64)$$

onde: C_A é o coeficiente de força axial, S_{ref} é a área aerodinâmica de referência do veículo, C_{yB} é a derivada do coeficiente de força lateral em relação ao ângulo de derrapagem β , e $C_{N\alpha}$ é a derivada do coeficiente de força normal em relação ao ângulo de ataque α .

Uma vez definida a força aerodinâmica faz-se necessário a quantificação do vetor momento aerodinâmico dado na Equação 65:

$$[M_A]^B = \begin{bmatrix} M_{Ax} \\ M_{Ay} \\ M_{Az} \end{bmatrix} \quad (65)$$

Para calcular os valores das componentes desse momento torna-se necessário a apresentação das componentes adimensionais da velocidade do foguete expressas nas Equações 66 a 68:

$$\bar{p} = p \frac{L_{ref}}{2V_{atm}} \quad (66)$$

$$\bar{q} = q \frac{L_{ref}}{2V_{atm}} \quad (67)$$

$$\bar{r} = r \frac{L_{ref}}{2V_{atm}} \quad (68)$$

L_{ref} é o comprimento aerodinâmico de referência do foguete.

Para completar a definição das componentes do momento aerodinâmico deve-se adotar que $C_{l\delta}$ e C_{lp} são os coeficientes de momento de rolamento devido às empenas, δ é o valor da incidência das empenas e que C_{mq} e C_{nr} são os coeficientes de amortecimento devido ao movimento angular do foguete. Assim, as expressões das componentes descritas nas Equações 69 a 71:

$$M_{Ax} = (C_{l\delta}\delta - C_{lp}\bar{p})p_{din}S_{ref}L_{ref} - y_{cm}F_{Az} + z_{cm}F_{Ay} \quad (69)$$

$$M_{Ay} = \left[-C_{N\alpha} \left(\frac{x_{cp} - x_{cm}}{L_{ref}} \right) \alpha - C_{mq}\bar{q} \right] p_{din}S_{ref}L_{ref} \quad (70)$$

$$M_{Az} = \left[C_{y\beta} \left(\frac{x_{cp} - x_{cm}}{L_{ref}} \right) \beta - C_{nr}\bar{r} \right] p_{din}S_{ref}L_{ref} \quad (71)$$

4 METODOLOGIA DE DESENVOLVIMENTO DO SIMULADOR

Para que um *software* realize simulação da trajetória de um foguete o mesmo deve ser capaz de resolver problemas matemáticos denominados de Problema de Valor Inicial (PVI), onde para um dado sistema de equações diferenciais que descrevam esse fenômeno e conhecido suas condições iniciais de estado pode-se então obter a solução do sistema ao longo do tempo como já foi detalhado no item 2.3 e 2.6.

Ao simulador desenvolvido foi conferido o nome de AMELIA em sua versão inicial chamada de 2018a e o mesmo conta com os seguintes elementos lógicos para seu funcionamento: a) Modelos matemáticos, b) interface para entradas de parâmetros do foguete, c) interface para entrada de condições iniciais; d) banco de dados para armazenar os dados de entrada e saída do programa; e) rotina numérica de integração das equações diferenciais e f) interface de saída dos gráficos.

A criação do simulador proposto necessitou de 4 grandes fases de desenvolvimentos sendo esta sistematização necessária para organização e tentativa de simplificação desse processo. Para tanto as etapas foram:

1. **Organização e conexão dos modelos do simulador:** nesta fase os modelos matemáticos mostrados nos capítulos 2 e 3 foram organizados, reescritos em notação matricial e vinculados para criação de todas as rotinas de cálculos. A ideia central desta etapa foi preparar a solução matemática desses modelos.

2. **Divisão das rotinas e bibliotecas necessárias para criação das funcionalidades:** Nesta etapa nasceram os códigos do programa destinados a tarefas específicas do simulador. A ação central nesta fase foi a separação das partes do programa em estruturas simples e o mais reduzidas possíveis para que, dessa forma, se tentasse facilitar se desenvolvimento. Foram tanto implementadas rotinas computacionais que contemplassem funções para soluções matemáticas das equações quanto arquivos de entradas de dados necessários à solução dos PVIs.

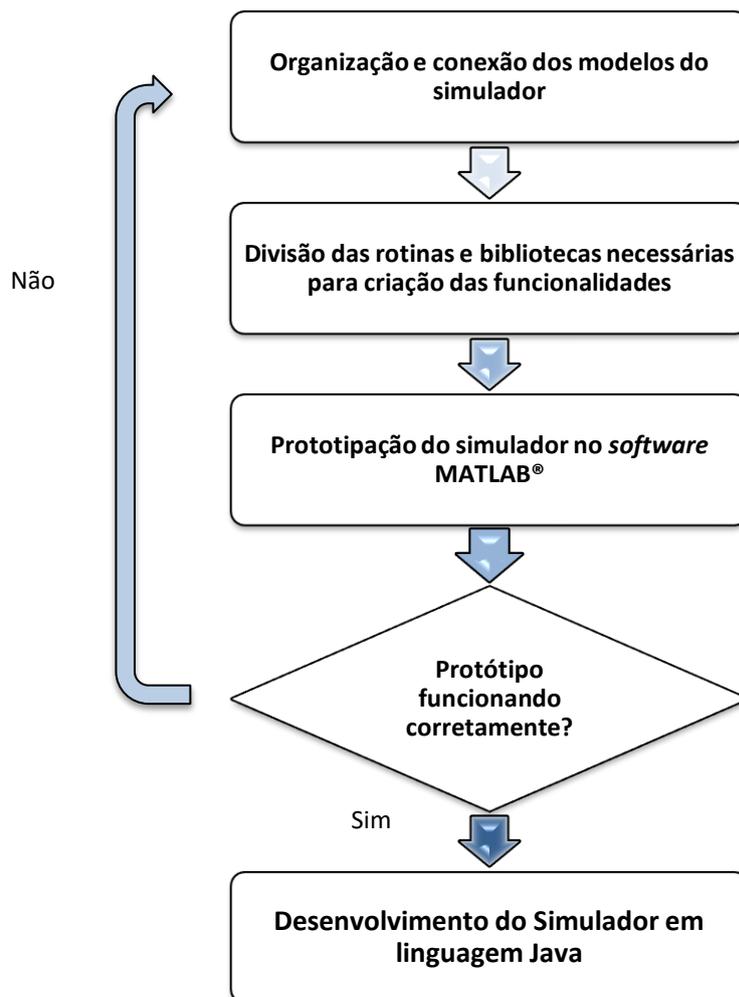
3. **Prototipação do simulador no *software* MATLAB®:** Nesta 3ª fase o simulador foi previamente prototipado neste ambiente para teste das estruturas criadas nas etapas anteriores. A escolha do MATLAB® para isso se deu principalmente pela facilidade de programação na sua linguagem (linguagem de alto nível computacional) e comodidade da plataforma para se trabalhar com matrizes. Uma vez que os modelos dinâmicos utilizados para a simulação são modelos vetoriais, a utilização de matrizes facilita a sua implementação em linguagem computacional. Outra grande vantagem são as funções nativas do MATLAB®

para solução de equações diferenciais como a função `ode45` que já dispõe em uma sintaxe simples as soluções desses problemas matemáticos mais complexos em torno dos PVI e dessa forma diminuem o tempo de programação e facilitam o desenvolvimento do mesmo.

4. **Desenvolvimento do Simulador em linguagem Java:** Uma vez que todas as etapas anteriores foram cumpridas com êxito a última fase do desenvolvimento do simulador consistiu em reescrever todo o programa em linguagem Java com a finalidade principal de desenvolver um *software* com alto potencial de compatibilidade entre sistemas operacionais (Windows, Linux e Mac, como discutido nos capítulos 1 e 2) bem como obter independência de programas prioritários adicionais para seu funcionamento.

O resumo da estrutura das etapas de desenvolvimento do simulador proposto é apresentado na Figura 13.

Figura 12 – Fluxograma de desenvolvimento do Simulador AMELIA



Fonte: Autores (2018)

A ênfase deste capítulo é apresentar como foi desenvolvida a etapa 4 responsável pela implementação do *software* na linguagem de programação Java sendo, portanto, responsável pelo produto final deste trabalho.

4.1 Armazenamento, acesso e deleção de dados

Motivado pela necessidade de conceder a aplicação produto deste trabalho uma capacidade de organizar e salvar dados de forma permanente para acesso rápido do mesmo e também pela habilidade de compartilhar principalmente os seus resultados com outros *softwares* foi desenvolvido no *software* AMELIA as opções de persistência de dados através de dois mecanismos que atendem sequencialmente estes requisitos impostos: a persistência por banco de dados e também a serialização abordadas detalhadamente nos tópicos a seguir.

4.1.1 Implementação da persistência de dados por banco de dados

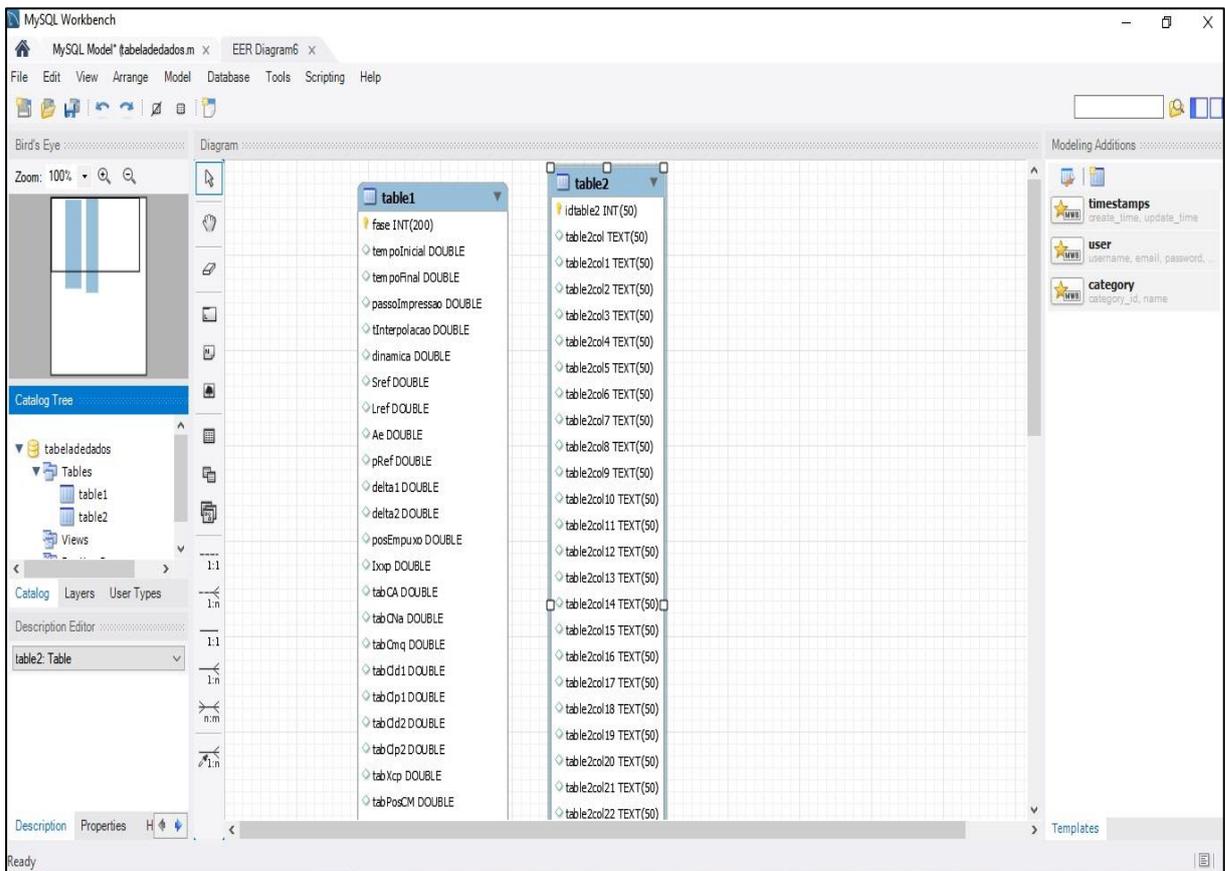
De posse dos modelos e da arquitetura desenvolvida na etapa 2 e testada na anterior, deu-se início a esta atividade pelo desenvolvimento do banco de dados que foi responsável por armazenar tanto os parâmetros das fases de voo dos foguetes como os valores referentes as condições de lançamento numa primeira tabela e após processamento da simulação os resultados salvos numa segunda tabela do mesmo banco de dados. Para tanto, foi utilizado o ambiente MySQL WorkBench, já apresentado no item 2.7.8.

O banco de dados criado foi nomeado de tabeladedados e as tabelas adicionadas receberam os nomes de table1 e table2 na ordem das necessidades esclarecidas acima. Na table1 o item “fase” recebeu a função de chave primária (elemento para referenciar o relacionamento entre entidades ou tabelas da base de dados) e na table2 o item “idtable2”. Ambos os itens também foram desenvolvidas para possuírem auto incrementação, assim facilitando a numeração dos mesmos já que será por acréscimo de uma unidade a cada nova linha e evitando passagem de valor por *insert*.

A chave primária é essencial para o funcionamento correto da base de dados, representando um registro único que propicia buscas e garante que cada valor dentro da tabela será diferente do outro. Todo registro na tabela deve possuir somente uma chave primária e elas não podem assumir valor nulo.

A modelagem do banco e suas tabelas podem ser vistas na Figura 14.

Figura 13 – Modelagem do banco de dados no *software* MySQL WorkBench



Fonte: Autores (2018)

Para acesso aos dados do banco foi utilizada a Java *DataBase Connectivity API* (*Application Programming Interface*) ou JDBC API. Ela possibilita ao programa as funcionalidades definidas pelo padrão *SQL Call Level Interface* ou *SQLCLI* e sua principal característica para este *software* é a portabilidade da aplicação cliente, inerente a linguagem Java.

Para o acesso aos dados a primeira atividade foi estabelecer uma conexão da aplicação com o banco de dados e para este propósito foi criada uma classe Java (*conexao.java*) específica para esta função. Duas ações ocorreram então sequencialmente: primeiro carregar o driver JDBC para o banco de dados na JVM da aplicação. Assim que carregado, o driver se registra para o *DriverManager* e se torna disponível para a aplicação. O segundo passo foi a utilização da classe *DriverManager* para abrir uma conexão com o banco de dados. A interface *Connection* designa um objeto, no caso *con*, para receber a conexão estabelecida.

Para a utilização dos recursos necessários a construção da classe `conexao` uma série de classes foram importadas a ela:

- ✓ `java.sql.Connection;`
- ✓ `java.sql.DriverManager;`
- ✓ `java.sql.PreparedStatement;`
- ✓ `java.sql.ResultSet;`
- ✓ `java.sql.SQLException;`
- ✓ `java.util.logging.Level;`
- ✓ `java.util.logging.Logger.`

São mostrados na Figura 15 os detalhes dessa conexão.

Figura 14 - Formato da conexão da aplicação com o banco de dados

```
public class conexao {

    private static final String DRIVER = "com.mysql.jdbc.Driver";
    private static final String URL = "jdbc:mysql://localhost:3306/tabeladedados";
    private static final String USER = "root";
    private static final String PASS = "robotica";

    public static Connection getConnection() {
        try {
            Class.forName(DRIVER);

            return DriverManager.getConnection(URL, USER, PASS);
        } catch (ClassNotFoundException | SQLException ex) {
            throw new RuntimeException("Erro na conexão: ", ex);
        }
    }

    public static void closeConnection(Connection con) {

        try {
            if(con!=null) {
                con.close();
            }
        }
    }
}
```

Fonte: Autores (2018)

A captura de exceções com `SQLException` foi feita por ser uma obrigação em Java para a utilização do JDBC. Depois de ser acessado é sempre importante liberar os recursos alocados pelo banco de dados e desta forma foi criado a função `closeConnection` que encerra a comunicação com o banco de forma direta.

Depois de obtida a conexão, pôde-se enviar consultas ao banco de dados por meio de comandos SQL. No caso das instruções que atualizam a base por meio de `insert`, `delete` ou `update`, os passos rigorosamente cumpridos foram:

- 1 - Carregar o driver;
- 2 - Desenvolver a conexão (*DriverManager.getConnection*);
- 3 - Preparar a consulta SQL;
- 4 - Executar a consulta (*executeUpdate*).

Para isto foram criadas duas classes Java a *itens.java* e a *itensDAO.java*, sendo a primeira responsável por conter os métodos *get* e *set* e dessa forma se apresenta equivalente ao modelo de entidade do banco de dados e a segunda classe é responsável pelo acesso aos dados do banco propriamente separando assim suas funcionalidades da classe lógica (primeira). Esta ação é recomendada como boa prática de programação por facilitar futuramente a manutenção do programa em caso de necessidade.

A Figura 16 exibe parte do código da classe *itens.java* expondo o funcionamento da estrutura dos métodos *get* e *set* enquanto a Figura 17 exibe parte da classe *itensDAO.java*.

Figura 15 - Estrutura da classe *itens.java*

```
8  /**
9  *
10 * @author Carlos Ronyhelton
11 */
12 public class itens {
13
14     /**
15      * @return the fase
16      */
17     public int getFase() {
18         return fase;
19     }
20
21     /**
22      * @param fase the fase to set
23      */
24     public void setFase(int fase) {
25         this.fase = fase;
26     }
27
28     /**
29      * @return the tempoInicial
30      */
31     public double getTempoInicial() {
32         return tempoInicial;
33     }
34 }
```

Fonte: Autores (2018)

Figura 16 - Parte da implementação da classe itensDAO.java

```

26 public class itensDAO {
27
28     public void create(itens item){
29
30         Connection con = conexao.getConnection();
31         PreparedStatement stmt = null;
32
33         try {
34             stmt = con.prepareStatement("INSERT INTO table1 (tempoInicial, tempoFinal, passoImpressao, tIn
35                 + "delta2, posEmpuxo, Ixxp, tabCA, tabCNa, tabCmq, tabCld1, tabClpl, tabCld2, tabClp2, tabXcp, tal
36                 + "tabMomentoControle, tabVento )VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);
37             stmt.setDouble(1, item.getTempoInicial());
38             stmt.setDouble(2, item.getTempoFinal());
39             stmt.setDouble(3, item.getPassoImpressao());
40             stmt.setDouble(4, item.getInterpolacao());
41             stmt.setDouble(5, item.getDinamica());
42             stmt.setDouble(6, item.getSref());
43             stmt.setDouble(7, item.getLref());
44             stmt.setDouble(8, item.getAe());
45             stmt.setDouble(9, item.getpRef());
46             stmt.setDouble(10, item.getDelta());
47             stmt.setDouble(11, item.getDelta2());
48             stmt.setDouble(12, item.getPosEmpuxo());
49             stmt.setDouble(13, item.getIxxp());
50             stmt.setDouble(14, item.getTabCA());

```

Fonte: Autores (2018)

4.1.2 Implementação da persistência de dados por arquivos (Serialização)

A persistência por serialização foi realizada através das bibliotecas para manipulação de arquivos de texto. O padrão de salvamento de dados em arquivos no formato txt foi dado preferência pela própria simplicidade da natureza dos dados a salvar serem apenas matrizes do tipo *double* obtidos após processamento da simulação e também pela facilidade de comunicação deste formato com outros *softwares* para o caso de reutilização. Um percentual muito significativo de programas consegue manipular este formato de arquivo.

Duas formas de salvar arquivos de texto foram implementadas no *software* AMELIA: a primeira salva o arquivo no diretório atual de trabalho do programa e a segunda o usuário pode especificar uma pasta diferente para o mesmo propósito. No primeiro caso, a interface com o usuário consiste apenas na chamada de uma *JOptionPane.showInputDialog* que solicita ao mesmo que digite um nome ao arquivo e pede confirmação para o nome ou cancelamento da ação.

A chamada da classe *JOptionPane* nos proporciona uma série de métodos estáticos que ao serem invocados exibem caixas de diálogos muito simples e objetivas ao propósito sugeridos pelos seus nomes. A solicitação de uma caixa desse tipo no *software* AMELIA é visualizada no código abaixo:

String nome = JOptionPane.showInputDialog("Por favor inserir novo nome para o arquivo .txt \nseguido da extensão do mesmo (Ex: Nome.txt):");

O segundo caso consiste na chamada da classe *JFileChooser* que por sua implementação nos permite direcionar o diretório para salvamento do arquivo. Neste caso, utilizou-se uma variável do tipo *JFileChooser* para instanciar a classe como é mostrado abaixo:

```
JFileChooser salvar = new JFileChooser();
```

Em ambos os casos uma matriz de dados primeiramente recebe os dados advindos do componente *JTable* (nomeado de *jtabela2*) da classe *TabeladeResultados* (criada com o propósito de exibir todos os dados gerados na simulação) e então esses dados serão serializados em bytes para o arquivo de salvamento. O método propriamente dito para salvar os dados em arquivos de texto é realizado pela chamada das classes *FileWriter* e *BufferedWriter*. A classe *FileWriter* serve para escrever diretamente no arquivo, enquanto a classe *BufferedWriter* possui alguns métodos que são independentes do sistema operacional, como quebra de linhas.

A rotina de salvamento, fechamento dos buffers e notificação do sistema de que o arquivo não está mais sendo utilizado é visualizada na Figura 18.

Figura 17 – Rotina de salvamento da matriz de dados em arquivo .txt

```
BufferedWriter outputWriter = null;
outputWriter = new BufferedWriter(new FileWriter(salvar.getSelectedFile()+".txt"));
for (int i = 0; i < Mat.length; i++) {
    outputWriter.write(Arrays.toString(Mat[i])+"");
    outputWriter.newLine();
}
outputWriter.flush();
outputWriter.close();
```

Fonte: Autores (2018)

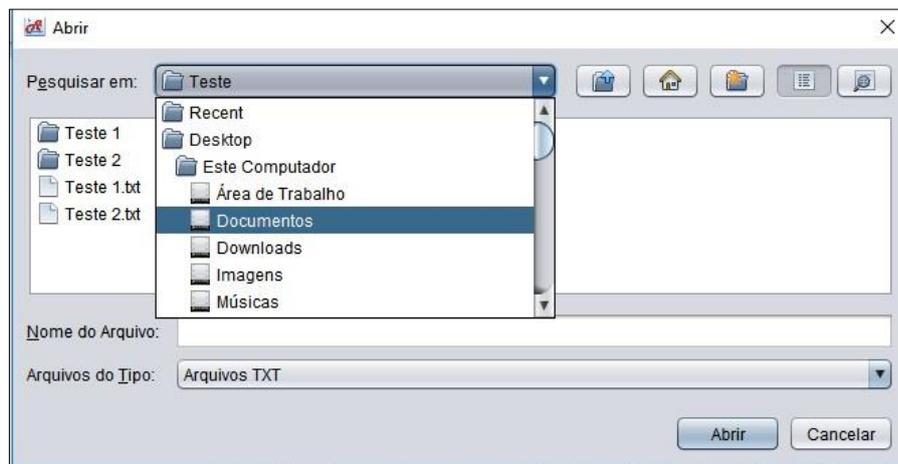
Assim como foi implementada a funcionalidade de salvamento em arquivo de texto dos dados de simulação como já é esperado também foi programada a função de abertura de arquivo para recuperação de simulações anteriores. Nesse caso foram utilizadas as classes *FileReader* e *BufferedReader* que servem para ler arquivos em formato de texto. A lógica de implementação neste caso segue a seguinte sequência:

1 - leitura do arquivo txt;

- 2 - extração dos dados para a matriz de dados;
- 3 – transferência dos valores da matriz para o banco de dados;
- 4 – exibição dos dados do banco na jTablea2.

A interface com o usuário neste caso também é realizada pela chamada da classe *JFileChooser* com possibilidade de abertura de arquivos limitadas ao formato “.txt” como é mostrado na Figura 19.

Figura 18 – Interface de abertura de arquivos



Fonte: Autores (2018)

A sequência de eventos para salvamento (sentido da esquerda para a direita) e abertura (sentido da direita para a esquerda) de arquivos de texto podem ser visualizadas na Figura 20.

Figura 19 - Sequência de eventos para salvamento e abertura de arquivos

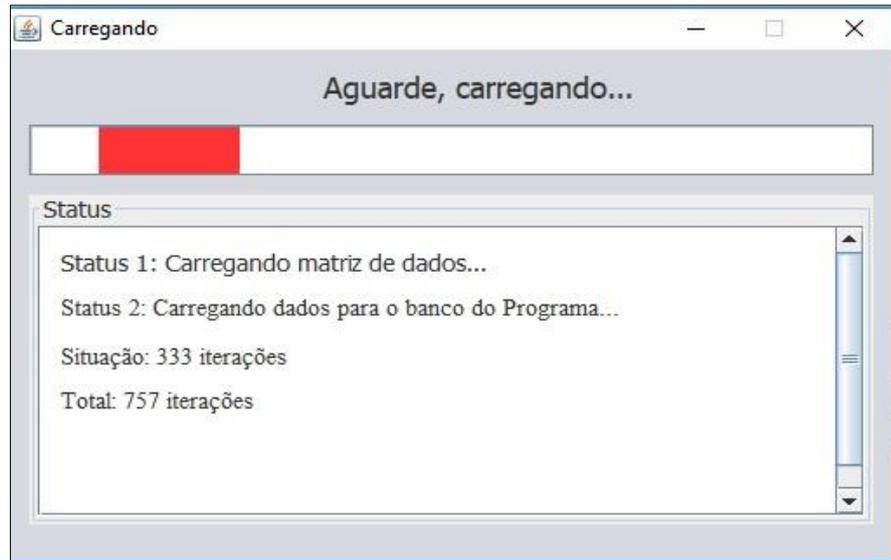


Fonte: Autores (2018)

Motivado pela preocupação da importação de uma grande quantidade de dados advindos do arquivo de texto dos resultados criou-se uma interface de acompanhamento do

referido processo que expõe sequencialmente as ações realizadas para este fim e as iterações atuais e totais para encerramento da atividade atual (ver Figura 21) e, o principal, ela evita que o usuário tente realizar alguma ação no programa que o force desnecessariamente por estar sobrecarregado.

Figura 20 – Interface de acompanhamento do processo de importação de dados



Fonte: Autores (2018)

4.2. Estrutura da entrada de dados

A entrada dos dados para processamento durante a simulação no *software* AMELIA consiste no preenchimento dos dados referentes as fases de voo do foguete tais como são apresentadas na Tabela 2. Esses parâmetros devem ser fornecidos para todas as fases.

Tabela 2 – Parâmetros das fases de voo do foguete

Variável	Descrição	Unidade
tempoInicial	Tempo inicial da fase	s
tempoFinal	Tempo final da fase	s
passoImpressao	Passo de impressão dos resultados da fase	-
tInterpolacao	Opção de interpolação: 0 - tempo de voo; 1 - tempo da fase	-
dinamica	Opção de dinâmica: 0 - movimento no trilho/rampa; 1 - movimento em 6 GDL	-
Sref	Área aerodinâmica de referência	m ²
Lref	Comprimento aerodinâmico de referência	m
Ae	Área de saída da tubeira	m ²

pRef	Pressão de referência para correção do empuxo	Pa
delta1	Incidência do 1º conjunto de empenas	graus
delta2	Incidência do 2º conjunto de empenas	graus
posEmpuxo	Distância entre o nariz do foguete e o ponto de aplicação do empuxo	m
Ixxp	Varição da inercia em X (para atuação do sistema ioiô)	Kgm ² /s
tabCA	Coeficiente de força axial	-
tabCNa	Derivada do coeficiente de força normal	1/rad
tabCmq	Coeficiente de amortecimento em arfagem/guinada	1/rad
tabCld1	Coeficiente de momento de rolamento em relação à incidência das empenas (1º conjunto)	1/rad
tabClp1	Coeficiente de amortecimento em rolamento devido às empenas (1º conjunto)	-
tabCld2	Coeficiente de momento de rolamento em relação à incidência das empenas (2º conjunto)	1/rad
tabClp2	Coeficiente de amortecimento em rolamento devido às empenas (2º conjunto)	-
tabXcp	Posição do centro de pressão em relação ao nariz do foguete	m
tabPosCM	Posição do centro de massa em relação ao nariz do foguete (xcm, ycm, zcm)	m
tabEmpuxo	Empuxo nominal	N
tabMassa	Massa	kg
tabMomInercia	Momentos de inércia (Ix, Iy, Iz)	K.gm ²
tabProdInercia	Produtos de inércia (Ixy, Ixz, Iyz)	K.gm ²
tabMomentoControle	Momento de controle (Mx, My, Mz)	Nm
tabVento	Intensidade do vento (Wx, Wy, Wz)	m/s

Fonte: Adaptado de IAE (2017)

Para a aquisição destes dados desenvolveu-se uma interface bastante intuitiva com o auxílio do recurso de *JTabbedPane*. Ele funciona como um container de painéis capaz de modularizar um formulário em diversas seções e desta forma reduz-se a quantidade de interfaces a serem criadas para o propósito de preenchimento dos parâmetros ou no mínimo evita-se também um superdimensionamento da tela de recebimento de dados.

Os campos para preenchimentos dos valores referentes as fases ficam disponíveis cada uma em sua aba filha do *JTabbedPane* específica. O resultado da criação da tela de dados das fases é visto na Figura 22. Percebe-se que os textos da interface foram implementados sobre *JLabels* e a entrada de dados pelos campos *JTextFields*.

Figura 21 – Interface de obtenção dos dados das fases

The screenshot shows the 'AMELIA 2018a' software interface. The window title is 'AMELIA 2018a'. At the top, there are tabs for 'Fase 1', 'Fase 2', 'Fase 3', 'Fase 4', 'Fase 5', 'Fase 6', 'Fase 7', and 'Fase 8'. The 'Fase 2' tab is selected, and the main area is titled 'Fase 2: Voo 1º estágio'. The interface is divided into three main sections:

- Dados Temporais e Numéricos:** Contains input fields for 'Tempo Inicial', 'Tempo Final', 'Passo de Impressão', 'Op. Tempo de Interp.', and 'Opção de Movimento'.
- Dados do estágio e de condições físicas:** Contains input fields for 'Ae de referência', 'Ce de referência', 'As da tubeira', 'P de referência para E', 'Inc. do 1º conj. de empenas', 'Inc. do 2º conj. de empenas', 'L (nariz e ponto de empuxo)', 'Variação da inércia em X', and 'Intensidade do vento'.
- Dados de arquivos da fase:** Contains input fields for 'Coeficiente de força axial', 'Derivada do coef. de forç...', 'Coef. de amort. ar/igu', 'Coef. de mom. de rol. (1º...', 'Coef. de amor. de rol. (1º...', 'Coef. de mom. de rol. (2º...', 'Coef. de amor. de rol. (2º...', 'Pos. do centro de pres. a...', 'Pos. do centro de m ao n...', 'Empuxo nominal', 'Massa', 'Momentos de inércia', 'Produtos de inércia', and 'Momento de controle'.

At the bottom right, there are three buttons: 'Salvar dados', 'Anterior', and 'Próximo'.

Fonte: Autores (2018)

Foi então criada a interface e funcionalidades relacionadas ao preenchimento de dados relativos dinâmica de voo para 6 GDL (graus de liberdade) e por fim uma interface para recebimento dos parâmetros relativos as condições de lançamento. Nesta última tela é selecionado, por exemplo, qual o modo de lançamento será utilizado para simulação sendo as opções a partir de trilho ou de rampa de lançamento implementados em uma *jComboBox*. A Figura 23 mostra a interface desta sessão de preenchimento de dados.

Figura 22 – Preenchimento das condições de lançamento

The screenshot shows the 'AMELIA 2018a' software interface for launch conditions. The window title is 'AMELIA 2018a'. The interface is divided into two main sections:

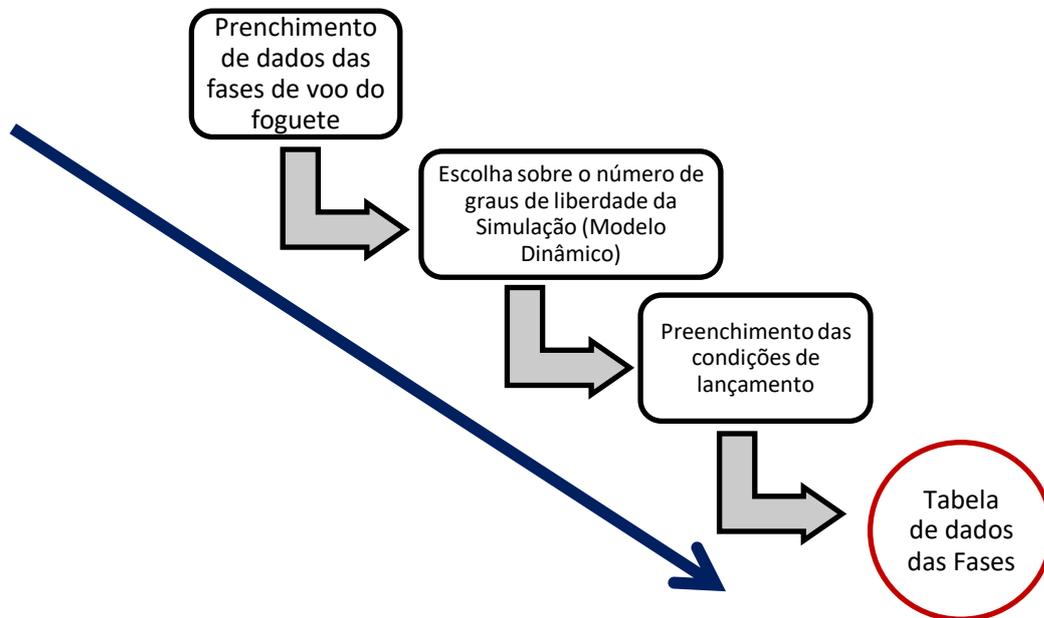
- Modelo da Terra:** Contains input fields for 'Gravidade (m/s²)', 'Raio médio da Terra (m)', and 'grav_L'.
- Modelo Atmosférico:** Contains input fields for 'Tatm (K)', 'R (Cte. Uni. dos Gases) [J/molK]', 'M0 (Massa molar média) [Kg/mol]', 'gamma (Razão entre cal. esp.)', 'Ratm', 'rho0 (Dens. em h=0) [kg/m³]', 'densAtm', 'pAtm', and 'velSom'. There is also a dropdown menu for 'Tipo de Lançamento' with 'Trilho de lançamento' selected.

At the bottom center, there is a 'Salvar' button.

Fonte: Autores (2018)

A sequência de procedimentos de entradas dos dados para alimentar a simulação do AMELIA é resumida na Figura 24:

Figura 23 – Sequência de entrada de dados do *software* AMELIA



Fonte: Autores (2018)

O método *readJTable* foi criado, como o próprio nome sugere, para fornecer a leitura do estado atual da tabela *JTable* e por tanto deve sempre ser chamado após cada alteração feita no banco de dados seja por inserção ou deleção de valores. Este método foi implementado baseado em percorrer a lista de dados por um laço *for* contendo o objeto do tipo *itens* para ser setado a cada iteração. Os dados são extraídos direto do banco de dados pelos métodos *gets*. A estrutura do método de leitura dos valores da tabela é exibida na Figura 25. A disposição dos objetos em linha é recomendada como uma boa prática de programação por facilitar atividades de manutenção quando necessárias.

Figura 24 – Implementação do método readJTable

```

33 public void readJTable(){
34     DefaultTableModel modelo = (DefaultTableModel) jTable1.getModel();
35     itensDAO idao = new itensDAO();
36     modelo.setNumRows(0);
37     for(itens p:idao.read()){
38
39         modelo.addRow(new Object[]{
40             p.getFase(),
41             p.getTempoInicial(),
42             p.getTempoFinal(),
43             p.getPassoImpressao(),
44             p.getInterpolacao(),
45             p.getDinamica(),
46             p.getSref(),
47             p.getLref(),
48             p.getAe(),
49             p.getpRef(),
50             p.getDelta1(),
51             p.getDelta2(),
52             p.getPosEmpuxo(),
53             p.getIxxp(),
54             p.getTabCA(),
55             p.getTabCNa(),
56             p.getTabCmq(),
57             p.getTabCld1(),
58             p.getTabClp1(),
59             p.getTabCld2(),
60             p.getTabClp2(),
61             p.getTabXcp(),
62             p.getTabPosCM(),
63             p.getTabEmpuxo(),
64             p.getTabMassa(),
65             p.getTabMomInercia(),
66             p.getTabProdInercia(),
67             p.getTabMomentoControle(),
68             p.getTabVento(),
69
70         });
71
72     }
73 }

```

Fonte: Autores (2018)

4.3 Interface gráfica

Para o desenvolvimento da interface do *software* AMELIA e todas as suas funcionalidades foi utilizado o Ambiente de Desenvolvimento Integrado NetBeans (apresentado no item 2.7.7). A API (*Application Programming Interface* ou, em português, Interface de Programação de Aplicativos) de interface gráfica utilizada foi o Swing que conta com o recurso *look-and-feel* único em todas as plataformas onde funciona (Windows, Linux, Mac), fato que garante a aplicação ter exatamente a mesma interface em qualquer sistema operacional. O *design* de todos os JFrames exceto as interfaces de acompanhamento são construídas com a aparência Nimbus. A aplicação deste foi garantida pelo simples trecho de código mostrado na Figura 26. Para as interfaces de acompanhamento de processos fez-se preferência visual ao estilo *Windows*.

Figura 25 - Aplicação do *design* Nimbus

```

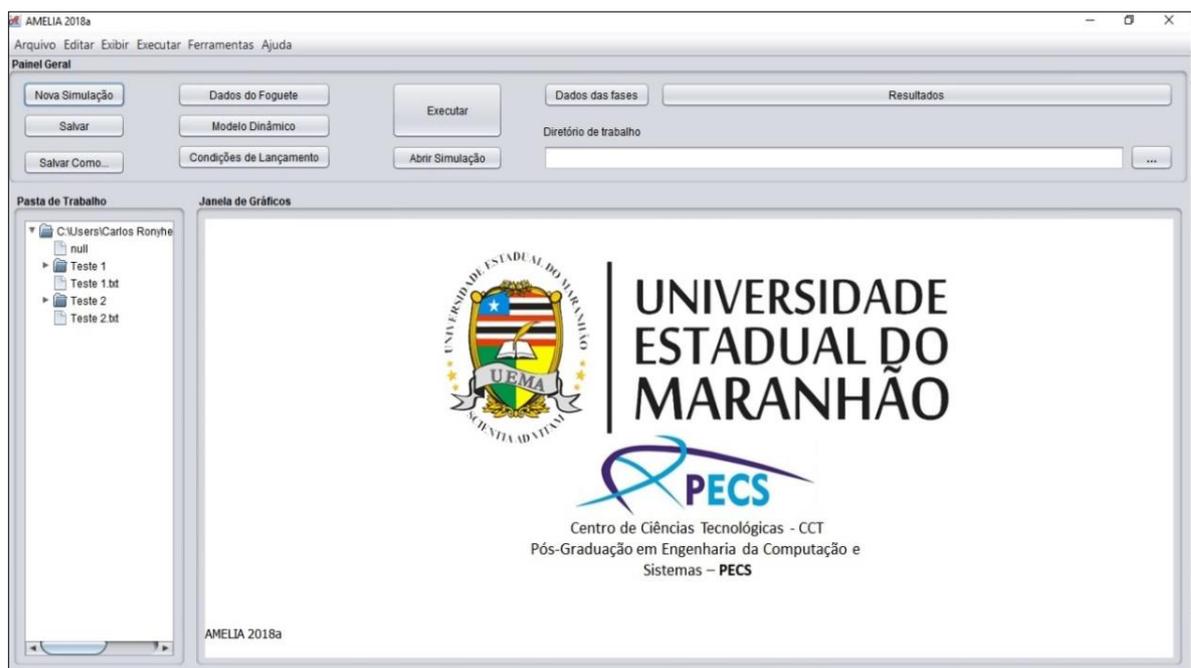
try {
    for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
        if ("Nimbus".equals(info.getName())) {
            javax.swing.UIManager.setLookAndFeel(info.getClassName());
            break;
        }
    }
} catch (ClassNotFoundException ex) {
    java.util.logging.Logger.getLogger(TabeladeResultados.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
} catch (InstantiationException ex) {
    java.util.logging.Logger.getLogger(TabeladeResultados.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
} catch (IllegalAccessException ex) {
    java.util.logging.Logger.getLogger(TabeladeResultados.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
} catch (javax.swing.UnsupportedLookAndFeelException ex) {
    java.util.logging.Logger.getLogger(TabeladeResultados.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
}

```

Fonte: Autores (2018)

Toda a interface do AMELIA foi desenvolvida com a forte preocupação de conciliar os recursos necessários à simulação de trajetória de foguetes em sua configuração mais ampla (contando com 6 graus de liberdade) e ao mesmo tempo garantir uma relação do usuário ao *software* de máxima intuitividade. Soma-se a este processo uma intensa busca pela melhor estética do mesmo. O resultado final da interface principal pode ser visto na Figura 27.

Figura 26 – Interface Principal do AMELIA



Fonte: Autores (2018)

As subdivisões da interface principal visualizadas na Figura 28 foram feitas para facilitar a compreensão de como estão organizados todos os elementos visuais do programa bem como as suas distribuições lógicas na janela.

Figura 27 – Subdivisões lógicas da Interface Principal



Fonte: Autores (2018)

Os elementos visuais encontrados na tela principal são:

- 1 – Painel de funções (verde): contém todas as funcionalidades do programa em uma estrutura de menus e subitens;
- 2 – Painel de acesso rápido (vermelho): dispõe de botões com a maior parte das funcionalidades do *software* alocadas a fácil acesso;
- 3 – Diretório de trabalho (azul escuro): apresenta qual a pasta atual está sendo usada para entrada ou saída de arquivos;
- 4 – Endereço de trabalho (marrom): Exibe qual o caminho do diretório de trabalho. Contém um botão para suprir a necessidade de se trocar a pasta atual por um resultado de busca gerado pela interface da classe *JFileChooser* ou por digitação da *String* referente ao caminho.
- 5 – Botões de entrada (laranja): botões responsáveis por fazerem chamada das interfaces de entradas de valores como dados do foguete ou condições de lançamento;

6 – Tabelas de dados e resultados (azul ciano): tabelas que contêm os dados referentes às fases de voo do foguete e valores gerados pós simulação numérica;

7 – Interface de gráficos (amarelo): painel responsável por receber os gráficos gerados após simulação. Neste componente são evocados 6 gráficos a partir dos resultados obtidos.

4.4 Saídas do *Software* AMELIA

Implementado como uma componente auxiliar da Interface Principal a tela de resultados nasci de uma *JInternalFrame* filha da *JDesktopPane*. O *JDesktopPane* é um painel da área de trabalho da aplicação (como o próprio nome já sugere) que é utilizado para que as janelas filhas sejam abertas apenas dentro da interface principal do sistema, não permitindo que as mesmas transladem para fora desta. Isso é o chamado Menu MDI.

A tabela de resultados (criada em uma *JTable*, ver Figura 29) contém os valores setados do banco de dados que foram inseridos pelo retorno da classe *IntegracaoNumerica*. Esta classe foi construída para receber o conjunto de equações diferenciais, composto pelas equações dinâmicas do movimento e pelas equações diferenciais auxiliares implementadas pelos modelos utilizados na simulação (já apresentados no capítulo 3) e foi concebida utilizando-se a rotina baseada em uma fórmula explícita de Runge-Kutta de quarta ordem. O processo de integração é realizado de modo que, a cada fase da trajetória, sejam utilizados os modelos para aquela fase contidos na tabela de fases da classe *TabeladeDados*.

Figura 28 – Interface de resultados ativa

linha	Título 2	Título 3	Título 4	Título 5	Título 6	Título 7	Título 8	Título 9	Título 10	Título 11	Título 12	Título 13	Título 14	Título 15	Título 16	Título 17	Título 18	Título 19
139308	0.0	0.0	0.0	0.0	-5.10	1.134464	1.4660766	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.10
139309	0.01	0.01	-2.14780...	-4.60598...	-50.9995...	1.134464	1.4660766	0.0	-0.09752...	0.0	0.0	0.0	0.0	0.0	-0.00430...	-0.00923...	0.09698...	50.99951
139310	0.02	0.02	-8.55155...	-1.83388...	-50.9980...	1.134464	1.4660766	0.0	-0.18456...	0.0	0.0	0.0	0.0	0.0	-0.00815...	-0.01748...	0.18355...	50.99807
139311	0.03	0.03	-1.75960...	-3.77349...	-50.9960...	1.134464	1.4660766	0.0	-0.21529...	0.0	0.0	0.0	0.0	0.0	-0.00951...	-0.02039...	0.21411...	50.99603
139312	0.04	0.04	-2.67231...	-5.73078...	-50.9939...	1.134464	1.4660766	0.0	-0.18829...	0.0	0.0	0.0	0.0	0.0	-0.00831...	-0.01783...	0.18726...	50.99398
139313	0.05	0.05	-3.33816...	-7.15870...	-50.9924...	1.134464	1.4660766	0.0	-0.10353...	0.0	0.0	0.0	0.0	0.0	-0.00457...	-0.00980...	0.10296...	50.99248
139314	0.06	0.06	-3.50197...	-7.51000...	-50.9921...	1.134464	1.4660766	0.0	0.03900...	0.0	0.0	0.0	0.0	0.0	0.00172...	0.00369...	-0.03878...	50.99211
139315	0.07	0.07	-2.90755...	-6.23526...	-50.9934...	1.134464	1.4660766	0.0	0.24010...	0.0	0.0	0.0	0.0	0.0	0.01050...	0.02274...	-0.23879...	50.99345
139316	0.08	0.08	-1.22705...	-2.76910...	-50.9971...	1.134464	1.4660766	0.0	0.50178...	0.0	0.0	0.0	0.0	0.0	0.02216...	0.04753...	-0.49304...	50.99710
139317	0.09	0.09	1.57818...	3.38443...	-51.0035...	1.134464	1.4660766	0.0	0.80186...	0.0	0.0	0.0	0.0	0.0	0.0354229	0.07596...	-0.79747...	51.00355
139318	0.1	0.1	5.82030...	0.00124...	-51.0131...	1.134464	1.4660766	0.0	1.120211	0.0	0.0	0.0	0.0	0.0	0.04948...	0.10612...	-1.11407...	51.01310
139319	0.11	0.11	0.00114...	0.00246...	-51.0258...	1.134464	1.4660766	0.0	1.4451537	0.0	0.0	0.0	0.0	0.0	0.06384...	0.13690...	-1.437237	51.02586
139320	0.12	0.12	0.00185...	0.00398...	-51.0418...	1.134464	1.4660766	0.0	1.7709716	0.0	0.0	0.0	0.0	0.0	0.0782338	0.16777...	-1.76127...	51.04185
139321	0.13	0.13	0.00271...	0.00581...	-51.0610...	1.134464	1.4660766	0.0	2.0968692	0.0	0.0	0.0	0.0	0.0	0.09263...	0.19864...	-2.08537...	51.06106
139322	0.14	0.14	0.00371...	0.00795...	-51.0835...	1.134464	1.4660766	0.0	2.4228196	0.0	0.0	0.0	0.0	0.0	0.1070296	0.22952...	-2.40954...	51.08356
139323	0.15	0.15	0.00485...	0.01040...	-51.10928...	1.134464	1.4660766	0.0	2.7488497	0.0	0.0	0.0	0.0	0.0	0.12143...	0.26041...	-2.73739...	51.10928
139324	0.16	0.16	0.00614...	0.01316...	-51.1382...	1.134464	1.4660766	0.0	3.0749505	0.0	0.0	0.0	0.0	0.0	0.1358379	0.29130...	-3.05810...	51.13823
139325	0.17	0.17	0.00757...	0.01623...	-51.1704...	1.134464	1.4660766	0.0	3.4011222	0.0	0.0	0.0	0.0	0.0	0.15024...	0.32220...	-3.38249...	51.17044
139326	0.18	0.18	0.00914...	0.01951...	-51.2058...	1.134464	1.4660766	0.0	3.7272947	0.0	0.0	0.0	0.0	0.0	0.16465...	0.35311...	-3.70894...	51.20586
139327	0.19	0.19	0.01088...	0.02329...	-51.2445...	1.134464	1.4660766	0.0	4.053878	0.0	0.0	0.0	0.0	0.0	0.17907...	0.38402...	-4.03147...	51.24456
139328	0.2	0.2	0.01272...	0.02729...	-51.2865...	1.134464	1.4660766	0.0	4.3800622	0.0	0.0	0.0	0.0	0.0	0.19349...	0.41494...	-4.35606...	51.28651
139329	0.21	0.21	0.01473...	0.03159...	-51.3317...	1.134464	1.4660766	0.0	4.7065173	0.0	0.0	0.0	0.0	0.0	0.2079134	0.44587...	-4.68073...	51.33170
139330	0.22	0.22	0.01688...	0.03621...	-51.3801...	1.134464	1.4660766	0.0	5.0330433	0.0	0.0	0.0	0.0	0.0	0.2223379	0.47680...	-5.00547...	51.38013
139331	0.23	0.23	0.01918...	0.04113...	-51.4318...	1.134464	1.4660766	0.0	5.3595403	0.0	0.0	0.0	0.0	0.0	0.23676...	0.50774...	-5.33027...	51.43181
139332	0.24	0.24	0.02162...	0.04636...	-51.4874...	1.134464	1.4660766	0.0	5.6862082	0.0	0.0	0.0	0.0	0.0	0.25119...	0.53869...	-5.655159	51.48744
139333	0.25	0.25	0.02420...	0.05190...	-51.5449...	1.134464	1.4660766	0.0	6.0130471	0.0	0.0	0.0	0.0	0.0	0.26563...	0.56964...	-5.980107	51.54491
139334	0.26	0.26	0.02693...	0.05775...	-51.6063...	1.134464	1.4660766	0.0	6.3398577	0.0	0.0	0.0	0.0	0.0	0.28006...	0.6006061	-6.30512...	51.60634
139335	0.27	0.27	0.02980...	0.06391...	-51.6710...	1.134464	1.4660766	0.0	6.666738	0.0	0.0	0.0	0.0	0.0	0.2945074	0.63157...	-6.63021...	51.67101
139336	0.28	0.28	0.03282...	0.07038...	-51.7389...	1.134464	1.4660766	0.0	6.993939	0.0	0.0	0.0	0.0	0.0	0.30895...	0.66254...	-6.95537...	51.73894
139337	0.29	0.29	0.03598...	0.07717...	-51.8101...	1.134464	1.4660766	0.0	7.3207131	0.0	0.0	0.0	0.0	0.0	0.32339...	0.69352...	-7.28060...	51.81012
139338	0.3	0.3	0.0392912	0.08426...	-51.8845...	1.134464	1.4660766	0.0	7.6478072	0.0	0.0	0.0	0.0	0.0	0.33784...	0.72451...	-7.60591...	51.88455
139339	0.31	0.31	0.04274...	0.09166...	-51.9622...	1.134464	1.4660766	0.0	7.9749726	0.0	0.0	0.0	0.0	0.0	0.3522995	0.75550...	-7.93128...	51.96224
139340	0.32	0.32	0.04633...	0.09937...	-52.0431...	1.134464	1.4660766	0.0	8.302209	0.0	0.0	0.0	0.0	0.0	0.36675...	0.78650...	-8.25672...	52.04318
139341	0.33	0.33	0.05007...	0.10739...	-52.12738...	1.134464	1.4660766	0.0	8.6295167	0.0	0.0	0.0	0.0	0.0	0.3812144	0.81751...	-8.58224...	52.12738
139342	0.34	0.34	0.05396...	0.11572...	-52.21483...	1.134464	1.4660766	0.0	8.9568955	0.0	0.0	0.0	0.0	0.0	0.39567...	0.84853...	-8.90782...	52.21483
139343	0.35	0.35	0.05799...	0.12436...	-52.3055...	1.134464	1.4660766	0.0	9.2843456	0.0	0.0	0.0	0.0	0.0	0.41014...	0.87955...	-9.233485	52.30553

Fonte: Autores (2018)

A partir de então cada gráfico é gerado por uma classe específica para si onde são configurados os dados e a estética que ele apresentará. As classes geradoras foram chamadas genericamente de GeradorDeGrafico e tem o acréscimo no nome pela relação de conjuntos que ele plota (por exemplo: GeradorDeGraficoAltitudeTempo).

A plotagem dos gráficos é realizada com a chamada da biblioteca *JFreeChart* acessando os dados advindos das linhas e colunas da *JTable* da interface de resultados. Esse acesso é garantido com um laço *for* que varre da primeira a última linha de um par de colunas específicas aos conjuntos que se deseja plotar no gráfico. Na sequência esses são enviados pela variável séries (instaciadora da classe *XYSeries*) ao dataset da classe *XYSeriesCollection* sendo, portanto, os gráficos gerados como linhas. A Figura 30 mostra como são feitas as criações das variáveis instaciadoras das classes da biblioteca *JFreeChart* e como esse mecanismo de varredura dos dados trabalha.

Figura 29 – Coleta dos dados para plotagem

```

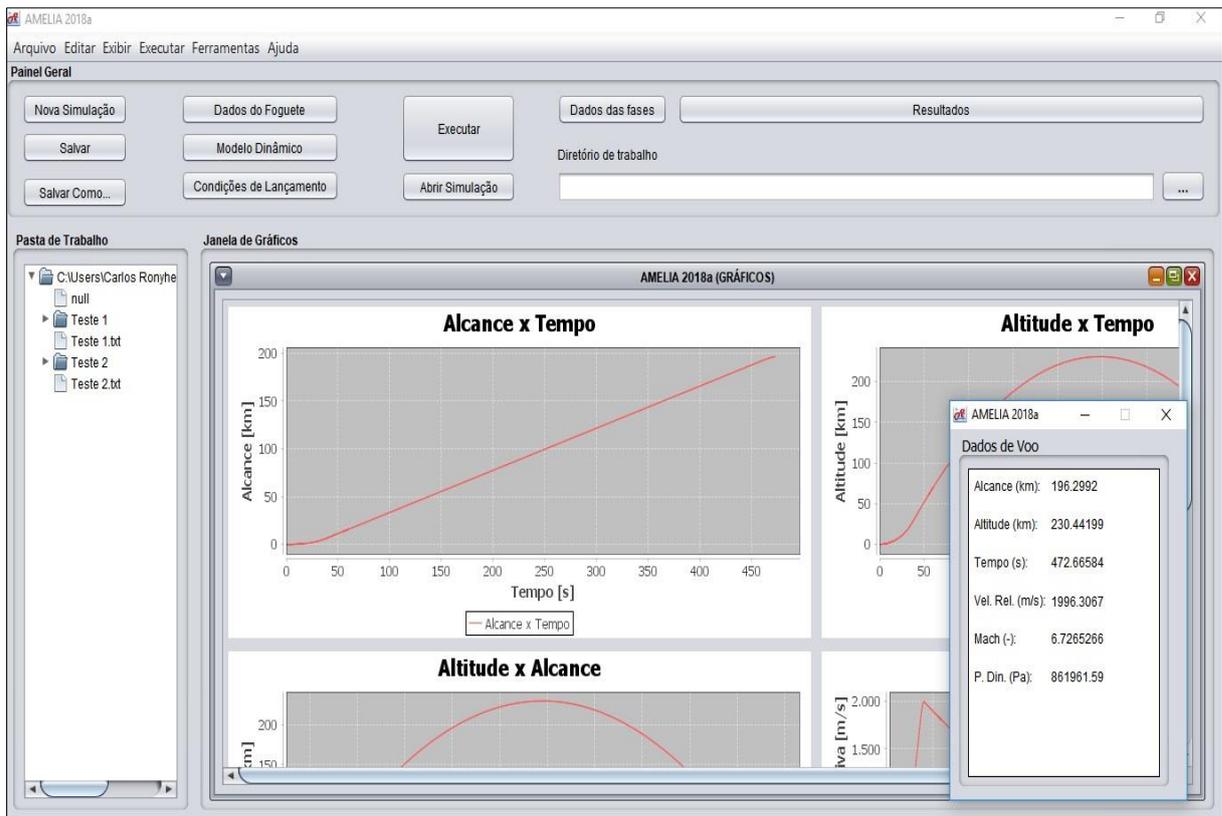
37 public JPanel orientacaoPanel(){
38
39     XYSeriesCollection dataset = new XYSeriesCollection();
40     XYSeries series1 = new XYSeries("Altitude x Alcance");
41     TabelaDeResultados tabela2 = new TabelaDeResultados();
42     itens item = new itens();
43     itensDAO idao = new itensDAO();
44
45     for(int linx = 0; linx<tabela2.jtabela2.getRowCount(); linx++){
46
47         series1.add(Double.parseDouble((String) tabela2.jtabela2.getValueAt(linx,4))/1000 ,
48                 Double.parseDouble((String) tabela2.jtabela2.getValueAt(linx,18))/1000);
49
50     }
51
52     dataset.addSeries(series1);
53
54     criaGrafico = ChartFactory.createXYLineChart("Altitude x Alcance",
55         "Alcance [km]", "Altitude [km]", dataset, PlotOrientation.VERTICAL, true, true, false);
56

```

Fonte: Autores (2018)

Uma segunda interface (além da *JInternalFrame*) foi construída para exibir os resultados da simulação. Ela contém apenas os máximos valores das colunas de Tempo, alcance, altitude, velocidade relativa, Número de Mach e pressão dinâmica. A Interface de resultados com a sua *JInternalFrame* ativada mostrando os gráficos e a interface de dados de voo chamada são exibidas na Figura 31.

Figura 30 – Interface Principal com a *JInternalFrame* ativa



Fonte: Autores (2018)

Uma apresentação mais detalhada das imagens das interfaces que compõe o *software* AMELIA pode ser visualizada no apêndice A deste trabalho.

5 RESULTADOS E DISCUSSÕES

Este capítulo é destinado a apresentar os resultados obtidos com o AMELIA, simulador de voo de foguetes desenvolvido neste trabalho, quando configurado para atender a simulação de voo de um veículo suborbital e não controlado e que nos permite fazer a verificação dos resultados obtidos por meio de comparação com resultados do mesmo veículo em outras duas ferramentas de simulação discutidas nos itens 2.4.1 e 2.4.3 respectivamente: o ROSI e o RTS.

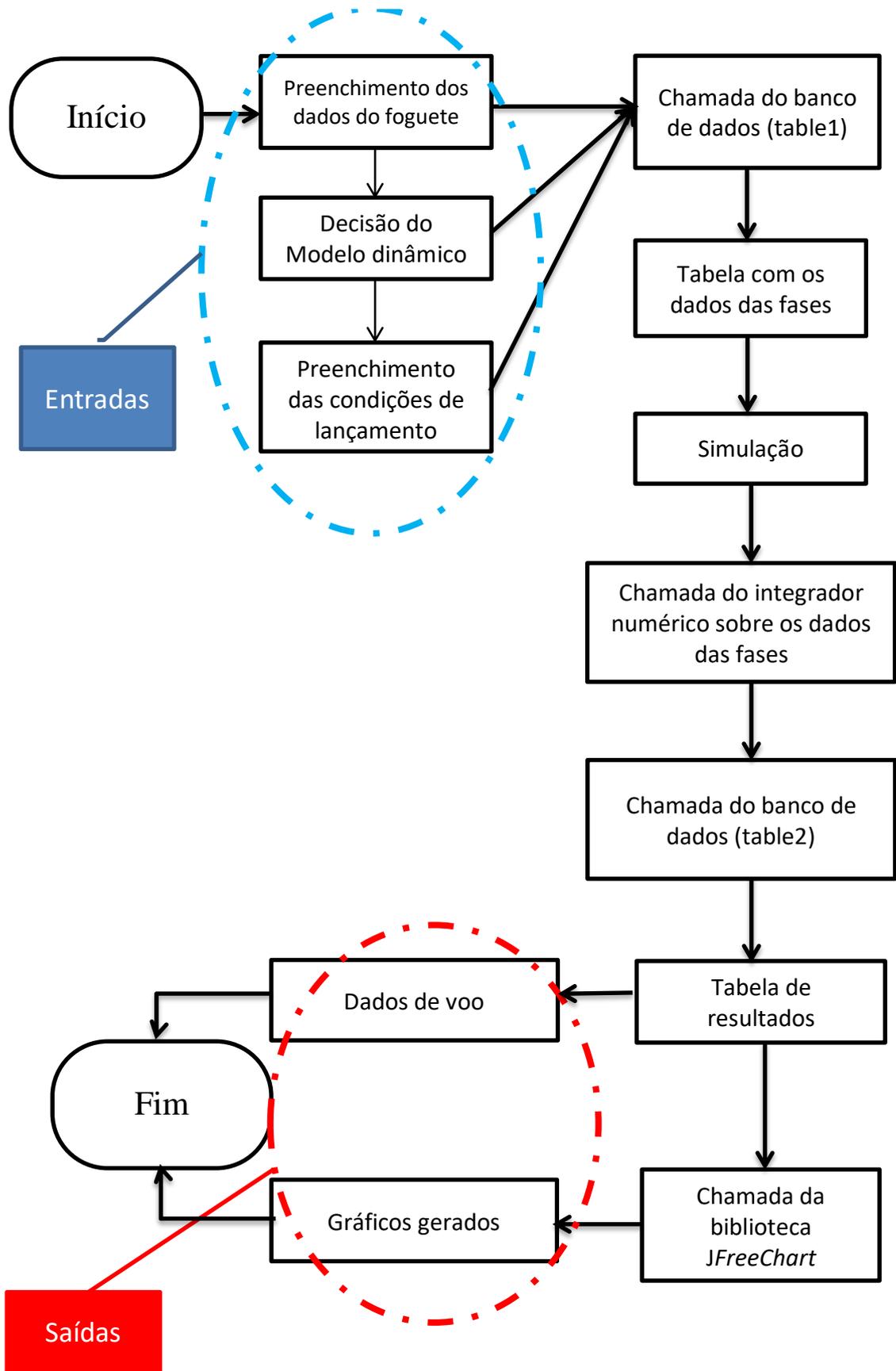
Como já apresentado este *software* foi desenvolvido contando como um dos requisitos preferenciais a característica de portabilidade entre sistemas operacionais. A simulação que será mostrada foi executada em um computador com sistema operacional Windows 10 PRO, com processador Intel® Core i5, 2.20 GHz, e 8 GB de memória RAM. Porém, antes de se iniciar a análise dos resultados de simulação apresenta-se a estrutura de funcionamento do AMELIA explorada para viabilizar estes resultados.

5.1 Mecanismos de funcionamento

Desde sua concepção à fase final de programação o presente simulador contou como um dos requisitos fundamentais a facilidade de utilização pelo usuário e a distribuição lógica de seus componentes nas telas. Dessa forma, todas as etapas de funcionamento do AMELIA contam com interfaces gráficas e o usuário somente interage realizando o preenchimento dos dados com entrada de valores via *JTextField*s ou através de busca de arquivos de textos que contenham dados necessários a simulação. O mecanismo de funcionamento do AMELIA se resume nos seguintes procedimentos listados abaixo e visualizados na Figura 32:

- 1 – Entrada dos dados do foguete, modelo dinâmico e condições de lançamento;
- 2 – Os dados são salvos no banco de dados na *table1* do banco;
- 3 – A exibição dos dados é feita pela comunicação do banco com a *JTable* da interface de Dados das fases;
- 4 – A simulação é feita por um botão de mesmo nome posicionado na Interface Principal que chama a rotina integradora e exporta os valores de resultados para o banco em sua *jtable2* e que podem ser vistos na *JTable* contida na Interface de Resultados.
- 5 – Da Interface de Resultados os dados são extraídos para as rotinas que evocam a biblioteca *JFreeChart* criando os gráficos exibidos na *JInternalFrame* da Interface Principal e também para Janela de Dados de Voo.

Figura 31 – Fluxograma de funcionamento do AMELIA



5.2 Simulação de veículo suborbital não controlado

Neste item são apresentados os resultados da simulação para 6 parâmetros de voo do VSB-30 (Veículo de Sondagem Brasileiro-30) com missão definida em prover uma carga útil em tempo mínimo de micro gravidade. Trata-se de um veículo lançador suborbital, não controlado, constituído por dois estágios a propulsão sólida. Ele tem a capacidade de transportar cargas úteis de até 400 kg para realização de experimentos na faixa de 270 km de altitude (IAE, 2017). Além dos estágios propulsivos, fazem parte do veículo um sistema de amortecimento do movimento de rolamento, conhecido como yo-yo. Este foguete é lançado a partir de trilho de lançamento.

As fases de voo do VSB-30 estão divididas nas seguintes atividades:

1 – Lançamento: realizado a partir de trilhos ganhando movimento restrito ao caminho disponível pelo próprio trilho.

2 – Desprendimento do trilho e voo de primeiro estágio: momento em que o foguete ganha liberdade de movimento no espaço tridimensional em 6 graus de liberdade. Nesta fase ainda são acionados os motores de indução de rolamento que aumentam a velocidade de rolamento significativamente.

3 – Separação do primeiro estágio e voo do segundo estágio: após a queima do motor desse estágio sua estrutura é separada do restante do foguete e então é iniciada a ignição e queima do motor de segundo estágio. Fase em que a estabilidade aerodinâmica é responsável pela estabilidade do movimento do veículo.

4 – Primeira fase balística: após a queima do motor de segundo estágio o foguete segue movimento sem propulsão alguma (voo balístico).

5 – Sistema yo-yo: nesta etapa é acionado o sistema yo-yo com finalidade de amenizar o rolamento do veículo adquirido nas fases anteriores.

6 – Segunda fase balística: após a atuação do yo-yo sua estrutura juntamente com a estrutura do segundo estágio são alijadas e a partir de então a carga útil segue em voo livre atingindo a altitude máxima (apogeu).

7 – Reentrada: uma vez atingido o apogeu o veículo realiza reentrada na atmosfera e segue em direção ao local de impacto.

A Tabela 3 apresenta o resumo destas fases e como elas ocorrem em cada um dos três simuladores que se comparou.

Tabela 3 – Modelamento para simulação do VSB-30 nos simuladores ROSI, RTS e AMELIA

Fase	ROSI	RTS	AMELIA
Lançamento	Dinâmica para trilhos; Massa variável; Motor com tubeira fixa; Aerodinâmica linear.	Dinâmica para trilhos; Massa variável; Motor com tubeira fixa; Aerodinâmica linear.	Dinâmica para trilhos; Massa variável; Motor com tubeira fixa; Aerodinâmica linear.
Voo do 1º estágio	Dinâmica com 6 GDL; Massa variável; Motor com tubeira fixa exist. Indução de rolamento; Aerodinâmica linear.	Dinâmica com 6 GDL; Massa variável; Motor com tubeira fixa exist. Indução de rolamento; Aerodinâmica linear.	Dinâmica com 6 GDL; Massa variável; Motor com tubeira fixa exist. Indução de rolamento; Aerodinâmica linear.
Voo do 2º estágio	Dinâmica com 6 GDL; Massa variável; Motor com tubeira fixa; Aerodinâmica linear.	Dinâmica com 6 GDL; Massa variável; Motor com tubeira fixa; Aerodinâmica linear.	Dinâmica com 6 GDL; Massa variável; Motor com tubeira fixa; Aerodinâmica linear.
1ª fase balística	Dinâmica com 6 GDL; Massa constante; Sem propulsão; Aerodinâmica linear.	Dinâmica com 6 GDL; Massa constante; Sem propulsão; Aerodinâmica linear.	Dinâmica com 6 GDL; Massa constante; Sem propulsão; Aerodinâmica linear.
Atuação do yo-yo	Dinâmica com 6 GDL; Massa constante; Sem propulsão; Aerodinâmica linear; Sistema yo-yo	Dinâmica com 6 GDL; Massa constante; Sem propulsão; Aerodinâmica linear; Sistema yo-yo	Dinâmica com 6 GDL; Massa constante; Sem propulsão; Aerodinâmica linear; Sistema yo-yo
2ª fase balística	Dinâmica com 6 GDL; Massa constante; Sem propulsão; Aerodinâmica linear.	Dinâmica com 6 GDL; Massa constante; Sem propulsão; Aerodinâmica linear.	Dinâmica com 6 GDL; Massa constante; Sem propulsão; Aerodinâmica linear.
Reentrada da carga útil	Dinâmica com 3 GDL; Massa constante; Sem propulsão; Apenas força de arrasto;	Dinâmica com 6 GDL; Massa constante; Sem propulsão; Apenas força de arrasto;	Dinâmica com 6 GDL; Massa constante; Sem propulsão; Apenas força de arrasto;

Fonte: Adaptado de Silveira (2014)

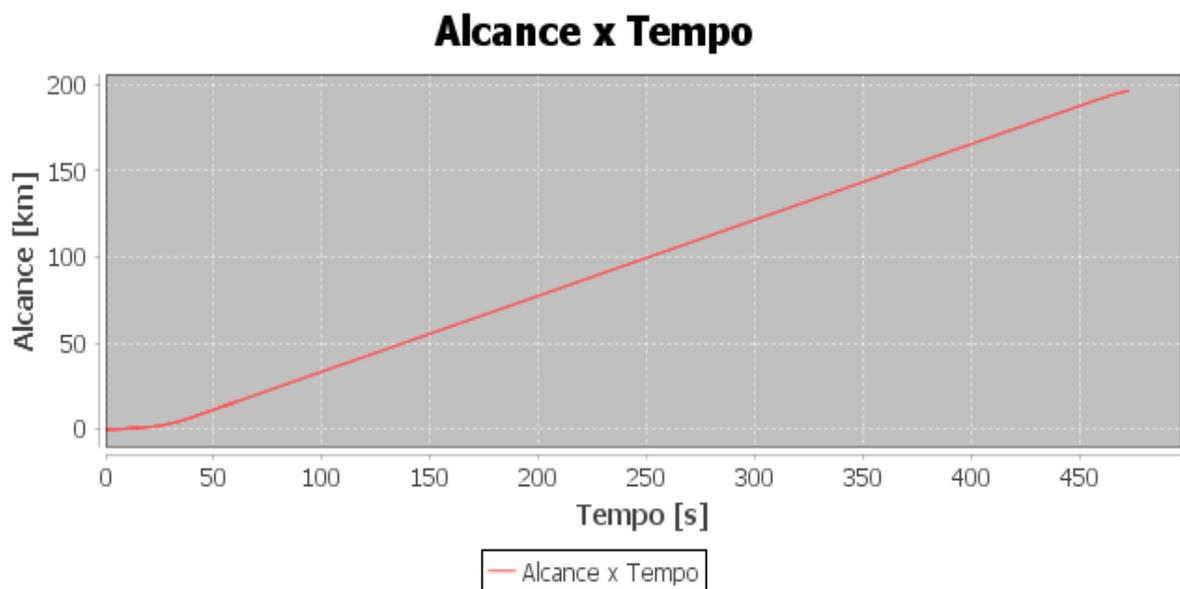
A sequência de exibição dos resultados foi construída em duas etapas: a primeira expõe os resultados obtidos para cada parâmetro pelo AMELIA e a segunda é a verificação dos resultados feita pela comparação em um gráfico que une as três curvas (do ROSI, RTS e AMELIA) na mesma plotagem evidenciando as diferenças de precisão dos *softwares* como ação necessária para o processo de validação do AMÉLIA.

5.2.1 Resultados de alcance

A Figura 33 mostra os resultados do alcance em relação ao tempo com o AMELIA e a Figura 34 expõe as comparações obtidas com as três ferramentas. Torna-se nítido que para os primeiros instantes do voo, as três retas são praticamente coincidentes, enquanto a diferença entre uma com as outras duas aumenta conforme a distância percorrida pelo veículo aumenta. Esse erro nos resultados aparece naturalmente como função das diferenças entre os modelos implementados no AMELIA e no RTS com relação aos do ROSI.

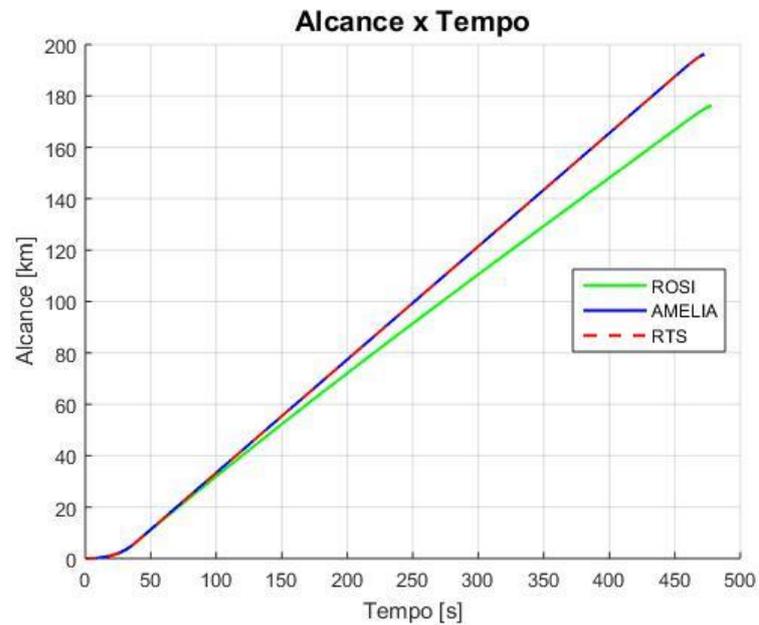
Esses erros somam-se ao longo do processo de integração da trajetória. Embora nessa escala de representação gráfica não seja visível há uma diferença dos resultados entre o AMELIA e o RTS a partir da ordem de 10^{-4} (ver Figura 35) resultado da rotina integradora utilizada no AMELIA. Vale-se lembrar que o RTS utiliza a função padrão do MATLAB® ode45 para solução das equações diferenciais.

Figura 32 – Gráfico do Alcance x Tempo para o VSB-30 no simulador AMELIA



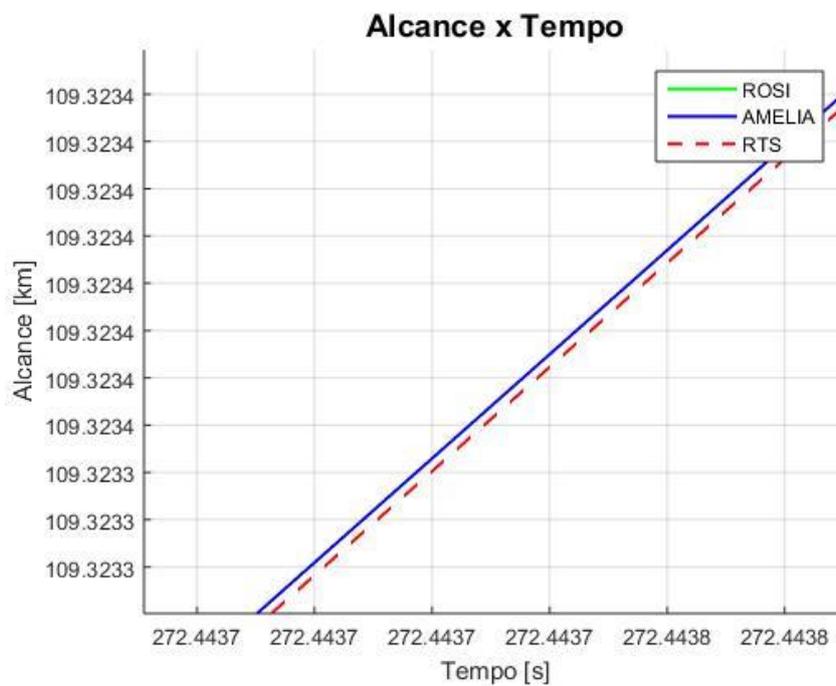
Fonte: Autores (2018)

Figura 33 – Gráfico de comparação do Alcance x Tempo para o VSB-30 nas plataformas AMELIA, RTS e ROSI



Fonte: Autores (2018)

Figura 34 – Comparação do Alcance x Tempo para o VSB-30 no AMELIA e no RTS



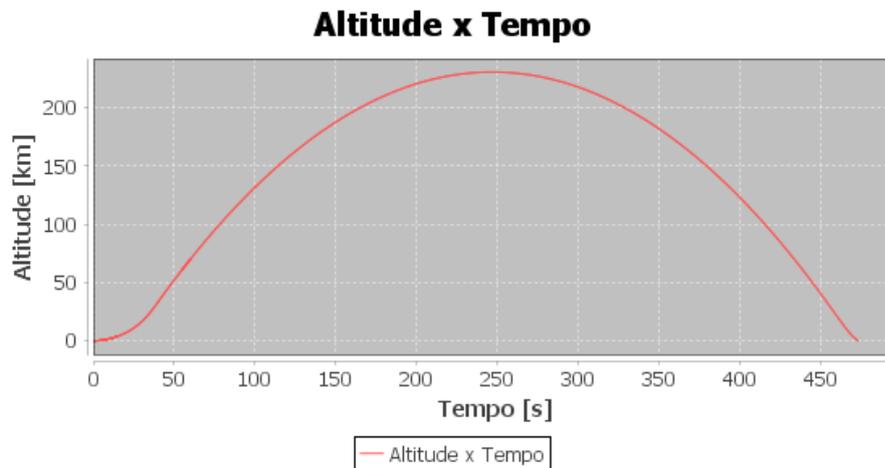
Fonte: Autores (2018)

Vale-se ressaltar que a terceira reta, do ROSI, não consta na Figura 35 em função da ampliação dada na exibição adotada não permitir que ele apareça.

5.2.2 Resultados de altitude

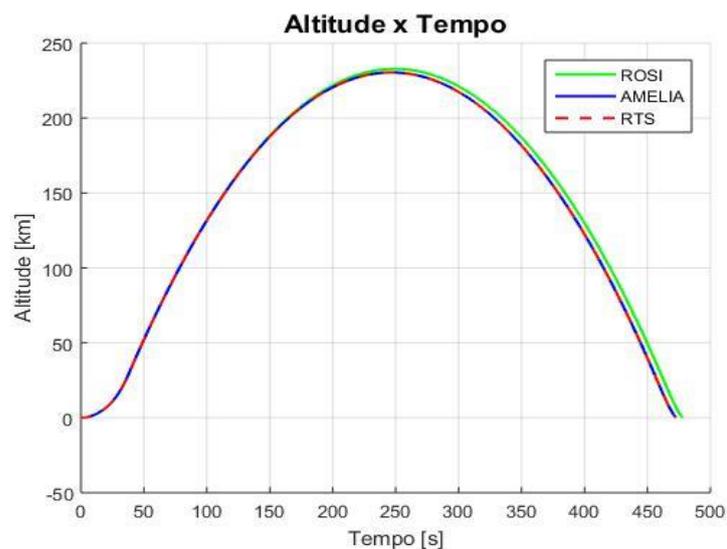
Podem-se visualizar os resultados da simulação de altitude em função do tempo realizados pelo AMELIA na Figura 36 e na Figura 37 a comparação do mesmo parâmetro entre os três *softwares* já mencionados.

Figura 35 – Gráfico da Altitude x Tempo para o VSB-30 no simulador AMELIA



Fonte: Autores (2018)

Figura 36 – Gráfico de comparação Altitude x Tempo para o VSB-30 nas plataformas AMELIA, RTS e ROSI



Fonte: Autores (2018)

Dada a proximidade dos valores entre as três simulações realiza-se uma comparação mais precisa através da análise da Tabela 4 que expõe os resultados máximos da

altitude para cada *software*. Desta tabela percebe-se que o valor do AMELIA é mais aproximado ao valor do RTS numa margem de erro em 10^{-7} . A comparação dos tempos para atingir o apogeu também é comparada na Tabela 4.

Tabela 4 – Comparação dos valores máximos de altitude

SIMULADOR	APOGEU	UNIDADE	TEMPO	UNIDADE
AMELIA	230,4419900000000	Km	247	s
RTS	230,4419896966828	Km	247	s
ROSI	232,0061000000000	Km	229	s

Fonte: Autores (2018)

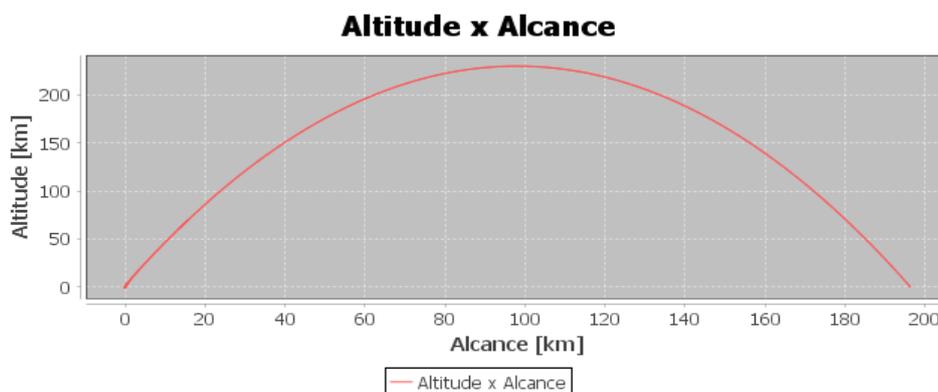
Para encerrar as comparações de distâncias percorridas em alcance e altitude plotam-se os gráficos nas Figuras 38 e 39 para se evidenciar a propagação dos erros obtidos pelos modelos e rotinas integradoras de cada *software*. Dentre as diferenças entre as equações do RTS e AMELIA (são modelos idênticos) com os do ROSI, pode ser citado o método de correção do valor do empuxo nos modelos propulsivos durante a cauda de empuxo do motor, que corresponde aos instantes de final de queima. Enquanto o AMELIA e o RTS utilizam a Equação 72 o ROSI utiliza o modelo 73 mostrados abaixo.

$$T_{cor} = T_{ref} + (p_{ref} - p_{atm})A_e \quad (72)$$

$$T_{cor} = T_{ref} + \frac{T_{ref}}{T_{tail}}(p_{ref} - p_{atm})A_e \quad (73)$$

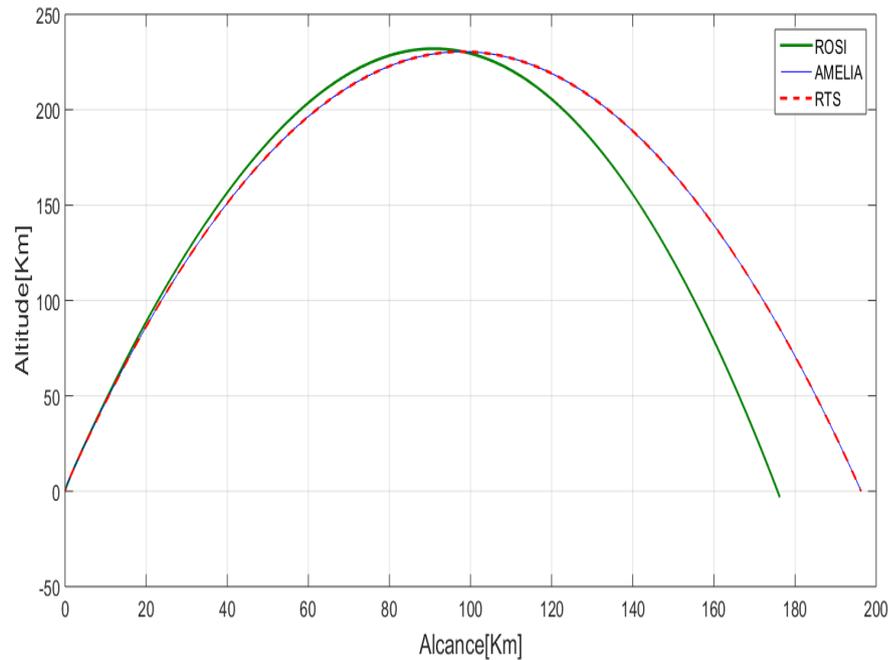
T_{tail} é o valor do empuxo no início da cauda de empuxo.

Figura 37 – Gráfico da Altitude x Alcance para o VSB-30 no simulador AMELIA



Fonte: Autores (2018)

Figura 38 – Gráfico de comparação do Altitude x Alcance para o VSB-30 nas plataformas AMELIA, RTS e ROSI



Fonte: Autores (2018)

5.2.3 Resultados da velocidade relativa

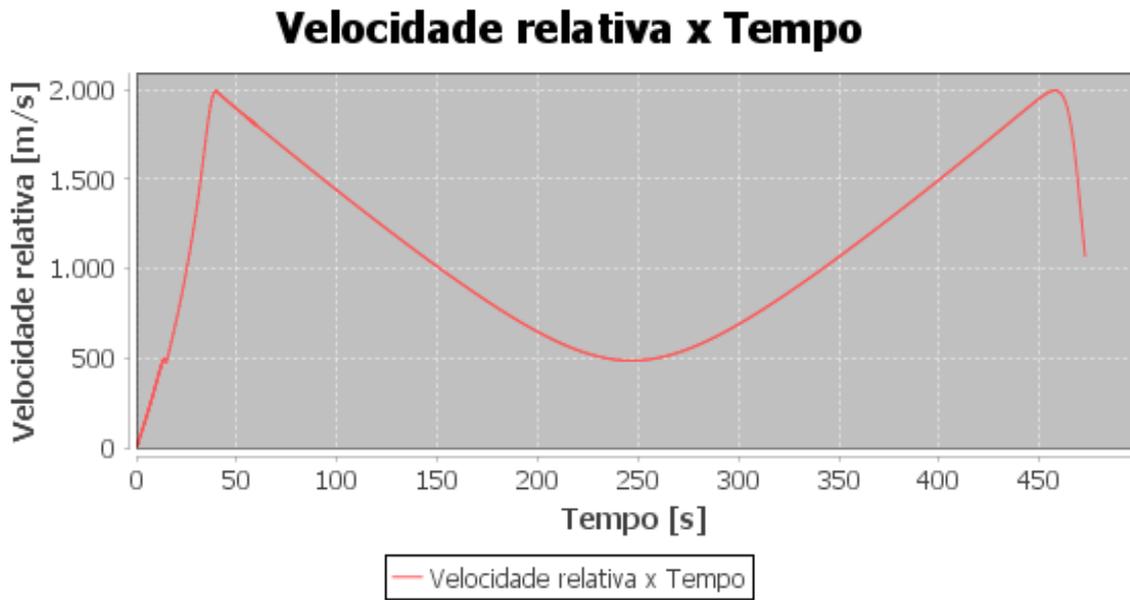
Através da análise das Figuras 40 e 41 nota-se uma diferença pequena nos resultados da velocidade relativa para os três simuladores que pode ser justificada devido aos diferentes modelos aerodinâmicos utilizados por eles. Para este parâmetro os modelos adotados pelo AMELIA e o RTS são os mesmos o que proporcionam a um erro na ordem de 10^{-4} gerado pela rotina integradora. Enquanto o ROSI por trabalhar com modelos diferentes apresenta maior contraste em seu valor de velocidade relativa máxima como seguem apresentados esses dados na Tabela 5. Os tempos para atingir essas velocidades são os mesmos entre o AMELIA e o RTS, porém diferentes do parâmetro temporal do ROSI.

Tabela 5 – Comparação dos valores máximos de velocidade relativa

SIMULADOR	VELOCIDADE	UNIDADE	TEMPO	UNIDADE
AMELIA	1996,306700000000	m/s	247	s
RTS	1996,306733867666	m/s	458	s
ROSI	1992,985000000000	m/s	458	s

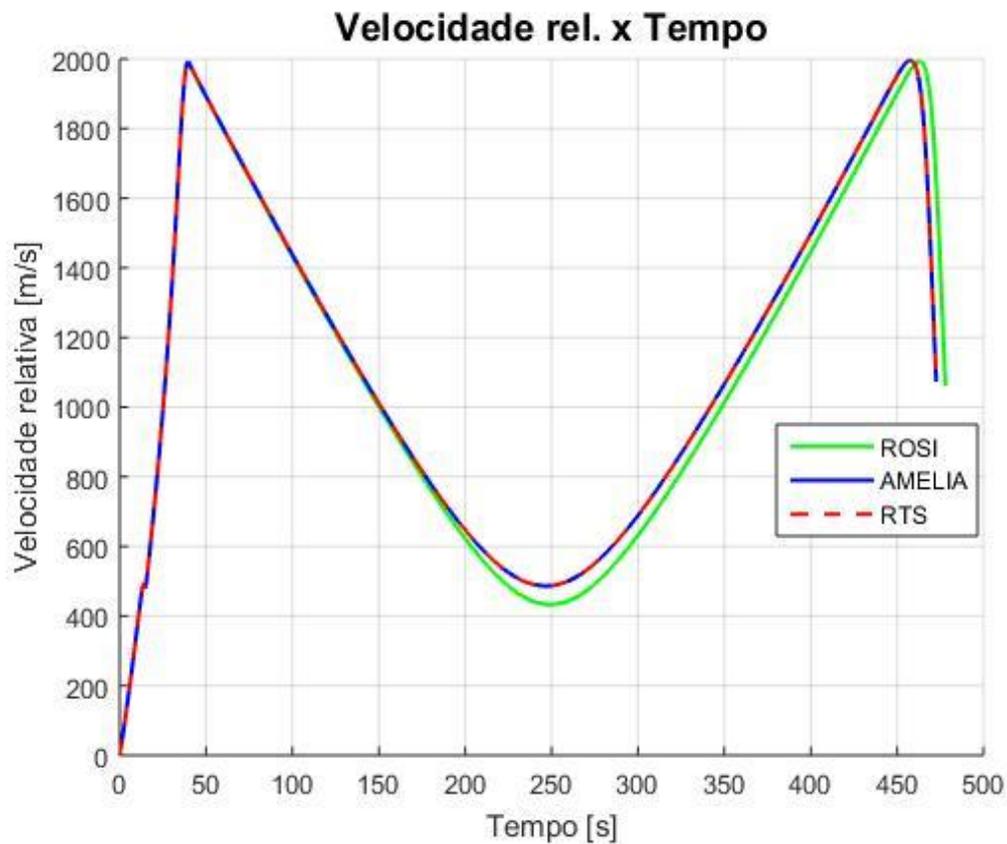
Fonte: Autores (2018)

Figura 39 – Gráfico da Velocidade Rel. x Tempo para o VSB-30 no simulador AMELIA



Fonte: Autores (2018)

Figura 40 – Gráfico de comparação da Velocidade Relativa x Tempo para o VSB-30 nas plataformas AMELIA, RTS e ROSI



Fonte: Autores (2018)

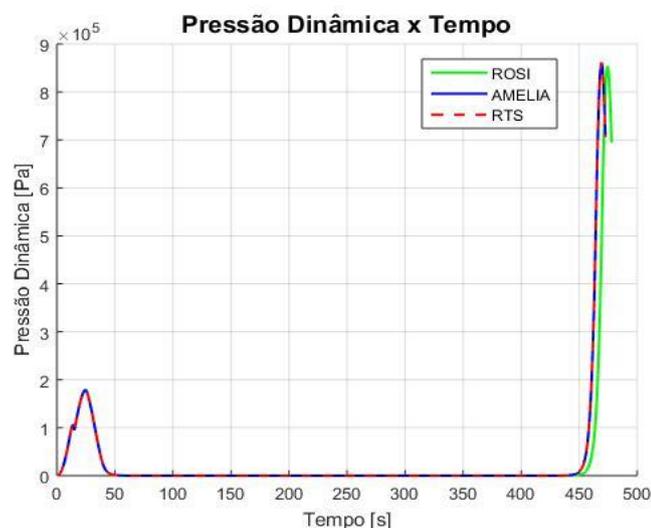
5.2.4 Resultados da pressão dinâmica

Os resultados desse parâmetro são muito próximos nas três ferramentas ocorrendo variações mais acentuadas a partir dos 450 s para o qual o AMELIA assume seu valor máximo em 861961,590 Pa, o ROSI em 853937,9 Pa e o RTS em 861961,587 (fato que nos leva ao erro de 10^{-2} em relação ao AMELIA). As diferenças encontradas nos valores de pressão dinâmica para a dupla AMELIA e RTS em relação ao ROSI ocorrem pelos métodos de correção de empuxo que consequentemente geram valores de velocidade relativa diferenciados (como já expostos no item 5.2.2) e que também afetam por consequência o parâmetro de pressão dinâmica como visto nas Figuras 42 e 43.

Figura 41 – Gráfico da Pressão Dinâmica x Tempo para o VSB-30 no simulador AMELIA



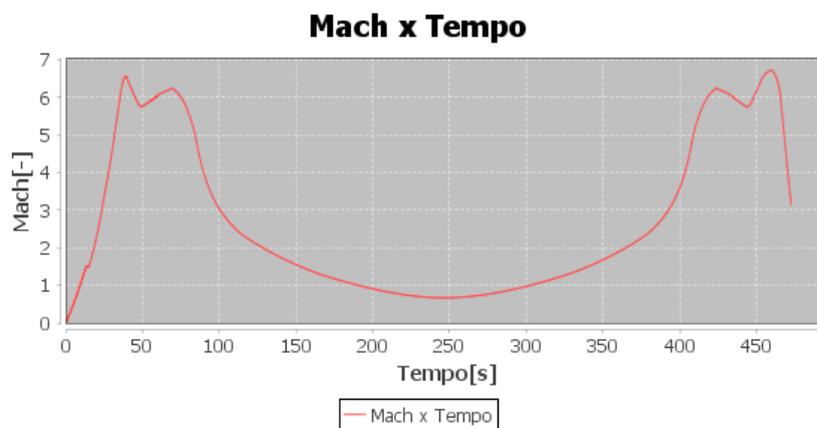
Figura 42 – Gráfico de comparação da Pressão Dinâmica x Tempo para o VSB-30 nas plataformas AMELIA, RTS e ROSI



5.2.5 Resultados do Número de Mach

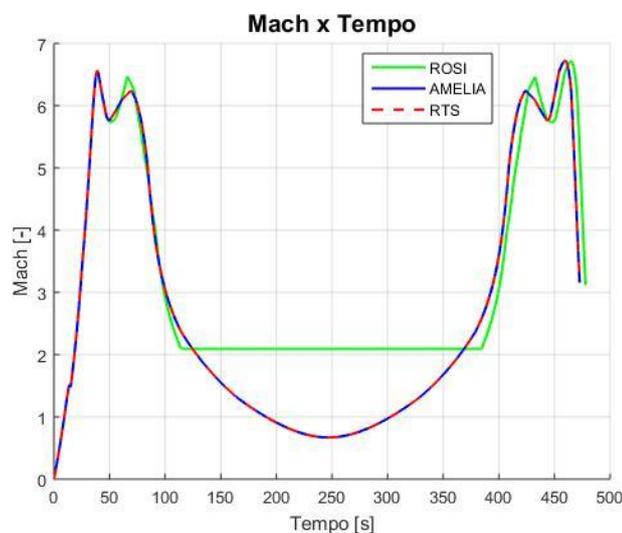
As diferenças encontradas dos simuladores para o número de Mach vistas nas Figuras 44 e 45 são geradas pelos diferentes modelos atmosféricos empregados em cada ferramenta: o RTS e o AMELIA utilizam a atmosfera padrão *U. S. Standard 1976* (comentado no item 3.1), enquanto o ROSI utiliza a *U. S. Standard 1962*. O valor relativamente constante do número de Mach gerado pelo ROSI no intervalo que compreende instantes após os 100 s até proximidades dos 400 s ocorre devido a uma simplificação no modelo atmosférico dessa plataforma, que não calcula a velocidade do som para altitudes maiores do que 150 km.

Figura 43 – Gráfico do Número de Mach x Tempo para o VSB-30 no simulador AMELIA



Fonte: Autores (2018)

Figura 44 – Gráfico de comparação do Número de Mach x Tempo para o VSB-30 nas plataformas AMELIA, RTS e ROSI

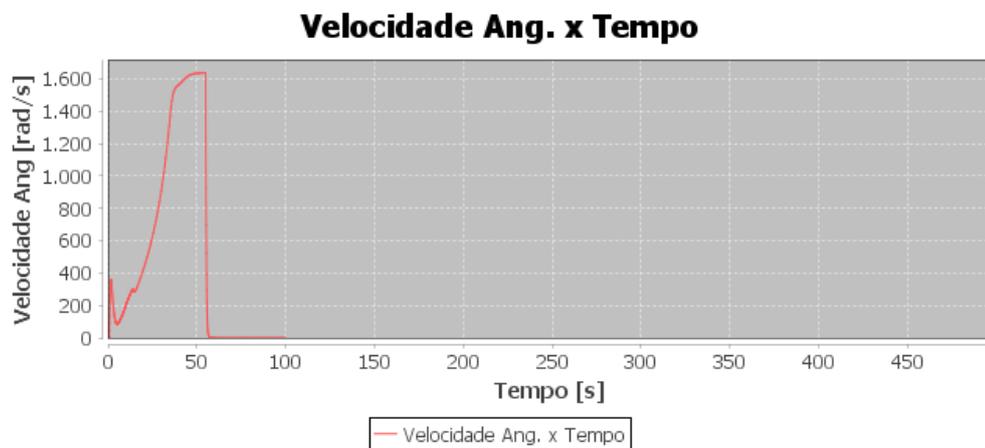


Fonte: Autores (2018)

5.2.6 Resultados da velocidade de rolamento

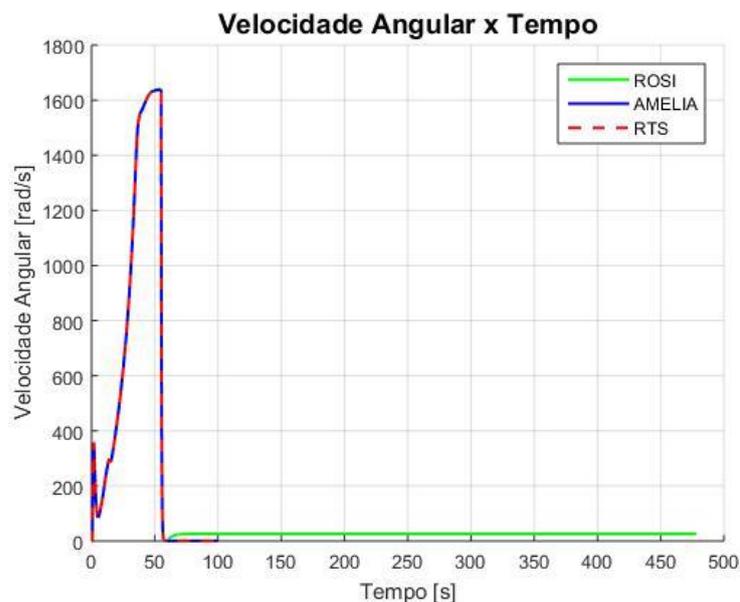
Por último são apresentados aqui os resultados para a velocidade angular do VSB-30 em função do tempo de voo. As três simulações são relativamente coincidentes com uma pequena variação causada pelo ROSI não considerar a variação do tensor de inércia do veículo no momento aparente de Coriolis fato adotado nos simuladores AMELIA e RTS. O comportamento desse parâmetro pode ser visualizado nos gráficos das Figuras 46 e 47.

Figura 45 – Gráfico da Velocidade Angular x Tempo para o VSB-30 no simulador AMELIA



Fonte: Autores (2018)

Figura 46 – Gráfico da comparação da Velocidade Angular x Tempo para o VSB-30 nas plataformas AMELIA, RTS e ROSI



Fonte: Autores (2018)

6 CONSIDERAÇÕES FINAIS

Este projeto apresenta o desenvolvimento de uma plataforma para simulação de voo de foguetes atendendo o cumprimento de movimentos em 6 GDL (graus de liberdade). Para esse propósito foi realizada uma ampla pesquisa da literatura e estudo dos *softwares* já validados nesse mercado com o objetivo de elevar ao máximo possível o nível de precisão da simulação computacional proposta e de versatilidade do produto gerado.

A etapa de revisão da literatura realizada em primeiro momento na pesquisa permitiu a identificação de diversos modelos matemáticos adequados para a simulação de voo de foguetes e a adoção de quase todos os que vêm sendo submetidos a testes desde o STVLS ao RTS no Brasil, compreendendo modelos dinâmicos, modelos dos subsistemas do veículo e modelos ambientais.

Na sequência, a adoção de uma arquitetura de programação flexível, ao qual foi possível implementar por meio da estratégia de programação modular onde os modelos dos módulos do veículo, ou subsistemas, bem como os modelos dinâmicos e ambientais são implementados separadamente e encontram-se disponíveis em uma biblioteca possibilitaram uma maior amplitude de utilização deste *software* por permitir a combinação de diferentes tipos de veículos além da possibilidade de se adicionar em código novos modelos à biblioteca existente conforme a necessidade do usuário.

Adquirir excelência no desenvolvimento de tecnologias aeroespaciais e com ênfase no domínio de simulação de voo de foguetes (foco deste trabalho) figura hoje uma necessidade vital para que se possa ter liberdade e autonomia em relação às nações protagonistas desse mercado, principalmente pelo fato do simulador, produto final deste trabalho, funcionar livre de outros *softwares* em especial programas prioritários.

Seguindo a estrutura de desenvolvimento da plataforma proposta e as simulações comparativas já realizadas (comparações com os programas RTS e ROSI) foi alcançado um simulador com o nível de precisão requerido para simulações computacionais deste âmbito. O mesmo apresenta considerada flexibilidade quanto a diferentes veículos e missões de foguetes, e ao mesmo tempo integrou todas as suas funcionalidades a uma interface gráfica intuitiva e que dá acesso fácil a elas. Além de tudo isso, dada à ênfase de criar este livro texto como uma fonte de referência sobre programação de um simulador em linguagem de programação Java espera-se que o mesmo se torne uma referência didática sobre desenvolvimento de simuladores de trajetórias de foguetes e um veículo para difusão de conhecimentos pertinentes ao setor aeroespacial.

Sobre a utilização do simulador AMELIA como fonte primária de resultados para o Programa Espacial Brasileiro deve ser lembrado que ele foi desenvolvido como fruto de um trabalho de pesquisa de mestrado e ainda necessita de um maior número de utilizações e testes para o completo processo de verificação e validação de suas funções, mesmo que até o presente momento ele tenha obtido uma boa aprovação e recomendação quando comparado seus resultados com os de outras duas ferramentas já utilizadas nesse âmbito.

Além disso, este *software* não apresenta um fim em seu estado atual, haja visto, que ainda existem significativas melhorias que podem ser implementadas a ele uma vez que no cenário atual o RTS continua como sendo a ferramenta computacional de maior potencial para a simulação do VLM dada sua grande abrangência em relação tanto ao AMELIA quanto os demais simuladores disponíveis no IAE. Entretanto, a proposta deste trabalho de desenvolver uma plataforma de simulação baseada principalmente no modelo de software do RTS, mas com ênfase em ser livre de outros *softwares* para seu funcionamento (como o MATLAB® no caso do RTS) deve ter seu reconhecimento como um avanço muito engajado em vista da política de autonomia internacional.

Desta forma, lista-se abaixo uma série de sugestões para trabalhos futuros que podem significativamente elevar as possibilidades a serem exploradas quanto à simulação de voo de veículos lançadores.

6.1 Sugestões de trabalhos futuros

A simulação realizada neste trabalho foi conduzida com um veículo de propulsão sólida e desta forma facilitada pelas curvas de variação de massa e empuxo já serem dadas como parâmetros de entrada para a simulação. Como proposta de trabalho futuro pode-se se explorar a simulação de veículo a propelente líquido e, para tanto, modelos de propriedades de massa, propulsivos e de controle devem ser programados e incorporados à biblioteca de modelos.

O atual estado do simulador é carente de uma biblioteca de sensores e atuadores além de modelos de previsão de perturbações na trajetória e equações que considerem rigorosamente a mudança na atitude do veículo em face da separação de algum estágio.

A inserção de modelos para prever com rigorosa precisão o ponto de impacto e elementos orbitais já contidos no RTS se faz necessária em vista de aproximar ainda mais a amplitude de atuação dessas ferramentas.

Embora o AMELIA tenha capacidade de indicar restrições da trajetória como valores máximos de pressão dinâmica, há características de grande importância em simulação de voo de veículos lançadores que se fazem necessárias ao mesmo como a capacidade de localização de pontos de impacto de estruturas alijadas, estimativa da influência da excentricidade do centro de massa e as assimetrias do veículo.

Por fim, indica-se a criação da versatilidade de ampliação do número de fases de voo de foguetes (até o presente momento limitadas a 8 fases no máximo).

REFERÊNCIAS

AMARAL; H. M. C. **Análise e Métodos Numéricos em Engenharia**. São Luís - MA: Ed. UEMA. 2012

ABDUL KADIR, A.; Xu, X.; HAMMERLE, E. – **Virtual Machine Tools and Virtual Machining – A Technological Review. Robotics and Computer-Integrated Manufacturing**, vol. 27, no. 3, pp. 494-508, Jun. 2011.

AGÊNCIA ESPACIAL BRASILEIRA. **Centros de Lançamento**. 2017. Disponível em: <<http://www.aeb.gov.br/programa-espacial-brasileiro/infraestrutura-de-solo/centros-de-lancamento/>> Acesso em: 30 maio de 2018.

AGÊNCIA ESPACIAL BRASILEIRA. **Programa Nacional de Atividades Espaciais, PNAE: 2012-2021**. Brasília: Ministério da Ciência, Tecnologia e Inovação, Agência Espacial Brasileira, 2012.

AGÊNCIA ESPACIAL BRASILEIRA. **Veículos lançadores**. 2012. Disponível em: <<http://www.aeb.gov.br/programa-espacial/veiculos-lancadores/>>. Acesso em: 16 outubro 2017.

BALDESI, G.; TOSO, M. **European Space Agency's launcher multibody dynamics simulator used for system and subsystem level analyses**. CEAS Space Journal, v.3, p.27–48, jun. 2012.

BATE, R. R; MUELLER, D. D; WHITE, J. E. **Fundamentals of astrodynamics**. New York: Dover Publications, 1971.

BAZZO, W. A. PEREIRA, L. T. V. **Introdução à engenharia: conceitos, ferramentas e comportamentos**. Florianópolis: Ed. da UFSC, 2006.

BRASILEIRO, R. N. **Determinação preliminar de órbita de veículos espaciais a partir de observações dos radares do Centro de Lançamento de Alcântara**. Dissertação (Programa de Pós-graduação em Engenharia Aeronáutica e Mecânica). São José dos Campos, 2007.

CARRARA, V. **Cinemática e dinâmica de satélites artificiais**. São José dos Campos: INPE, 2012.

CECHTICKY, V.; MONTALTO, G.; PASETTI, A.; SALERNO, N. **The AOCS framework**. In: **International ESA Conference on Spacecraft Guidance, Navigation and Control Systems**. Frascati, 2002.

CHAPRA; S. C. **Métodos numéricos aplicados com o MATLAB: para engenheiros e cientistas**, Mc Graw Hill – Bookman, 2013.

CLARO, D. B.; SOBRAL; J. B. M. **Programação em Java**. Copyleft Pearson Education. Florianópolis, SC. 2008.

CORNELISSE, J.; SCHÖYER, H.; WAKKER, K. **Rocket propulsion and spaceflight dynamics**. Londres: Pitman, 1979.

DUARTE, R. N. **Simulação Computacional: Análise de uma célula de manufatura em lotes do setor de autopeças**. Dissertação (Mestrado). Itajubá: Universidade Federal de Itajubá, 2003.

GILBERT, D. **Welcome To JFreeChart!** Disponível em: < <http://www.jfree.org/jfreechart/>>. Acessado em 20 de abril de 2018.

GOULD, H.; TOBOCHNIK, J. **An introduction to computer simulation methods: applications to physical systems**. New York: Addison-Wesley, 1996.

HARREL, C. R.; MOTT, J. R. A.; BATEMAN, R. E.; BOWDEN, R. G.; GOOG, T. J. **Simulação: Otimizando os Sistemas**. 2 ed. São Paulo: Belge Engenharia e Sistemas Ltda, IMAM, 2002.

HOFFMANN, L. T. **Estudo de simuladores computacionais aplicados ao ciclo de desenvolvimento de plataformas orbitais**. São José dos Campos: INPE, 2009.

HOFFMANN, L. T.; PERONDI, L. F. **Estudo de simuladores computacionais aplicados ao ciclo de desenvolvimento de plataformas orbitais**. In: Workshop em Engenharia e Tecnologia Espaciais, 1. (WETE), 2010, São José dos Campos. Anais... São José dos Campos: INPE, 2010.

HUMBLE, R. W.; HENRY, G. N.; LARSON, W. J. **Space Propulsion Analysis and Design**. McGraw-Hill Professional, 1 edition. 1995.

KAPLAN, M. H. **Launch Vehicle Systems Design and Engineering, A Three Day Presentation Focused on Current Developments in the Launch Vehicle Industry, Lecture handout**, São José dos Campos, 1995.

LAW, A.M.; KELTON, W.D. **Simulation Modelling & Analysis**. 3. ed. Nova Iorque: McGraw-Hill Books, 2000.

LUTZ, H.; WENDET, W. **Taschenbuch der Regelungstechnik: mit MATLAB und Simulink**. Verlag Harri Deutsch, 2010.

MARTIN, A. C. M.; CARVALHO, M. M. de. **Avaliação do uso da simulação virtual no processo de desenvolvimento de produtos**. São Paulo, 2005.

NETBEANS. **NetBeans IDE Features**. Disponível em: <<http://netbeans.org/features/index.html>>. Acessado em 17 de fevereiro de 2018.

OGATA, K. **Engenharia de controle moderno**. Prentice-Hall, 5. ed, 2004.

OLIVEIRA, C. R. S. **Proposta de desenvolvimento de um foguete de baixo empuxo**. Monografia (Graduação em Engenharia Mecânica) – Universidade Estadual do Maranhão, São Luís, 2016.

PALMERIO, A. F. **Introdução à engenharia de foguetes**. São José dos Campos - SP: SindCT, Instituto Tecnológico de Aeronáutica (ITA), 2017.

RIBEIRO, M. V. F. Metodologia de projeto e validação de motores foguete propelente sólido. Dissertação de Mestrado em Engenharia Mecânica, Publicação UFSCar, Departamento de Engenharia Mecânica, Escola de Engenharia de São Carlos da Universidade de São Paulo, São Carlos, SP, 2013.

SALGADO, M. C. V. **Agregação individual em decisão em grupo – estudo de caso: avaliação da realização do voo tecnológico do veículo lançador de satélites VLS-1.** 2008. Dissertação (Mestrado Profissionalizante em Engenharia Aeroespacial) - Instituto Tecnológico de Aeronáutica (ITA), São José dos Campos, 2008.

SHIGLEY, Joseph E.; Mischke, Charles R.; Budynas, Richard G. **Projeto de Engenharia Mecânica**, 7. ed. Porto Alegre: Bookman, 2005.

SARMA, I.; PRASAD, U.; VATHSAL, S. **Computer simulation methods for launch vehicle mission and control problems. Proceedings of the Indian Academy of Sciences Section C: Engineering Sciences.** Springer India, v.1, n.4, p. 423–440, 1978. ISSN0250-5983.

SILVA, J. A. da. **Manual de utilização do programa de cálculo de trajetória do VLS-STVLS.** São José dos Campos: IAE, 1997.

Silva, J. F. V. da. **JFloat: Uma biblioteca de ponto flutuante para a linguagem Java com suporte a arredondamento direcionado.** Dissertação de Mestrado. Universidade Federal do Rio Grande do Norte. Natal/RN. 2007.

SILVEIRA, G. **Desenvolvimento de uma ferramenta computacional para simulação de voo de veículos lançadores.** 2014. Dissertação (Mestrado em Mecânica Espacial e Controle) - Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2014.

SUTTON, G. P.; BIBLARZ, O. **Rocket Propulsion Elements.** 8.ed. Hoboken: John Wiley & Sons, 2010.

TEWARI, A. Atmospheric and space flight dynamics: modeling and simulation with MATLAB and Simulink. Boston, MA: Birkhauser, 2007. (Modeling and simulation in science, engineering and technology).

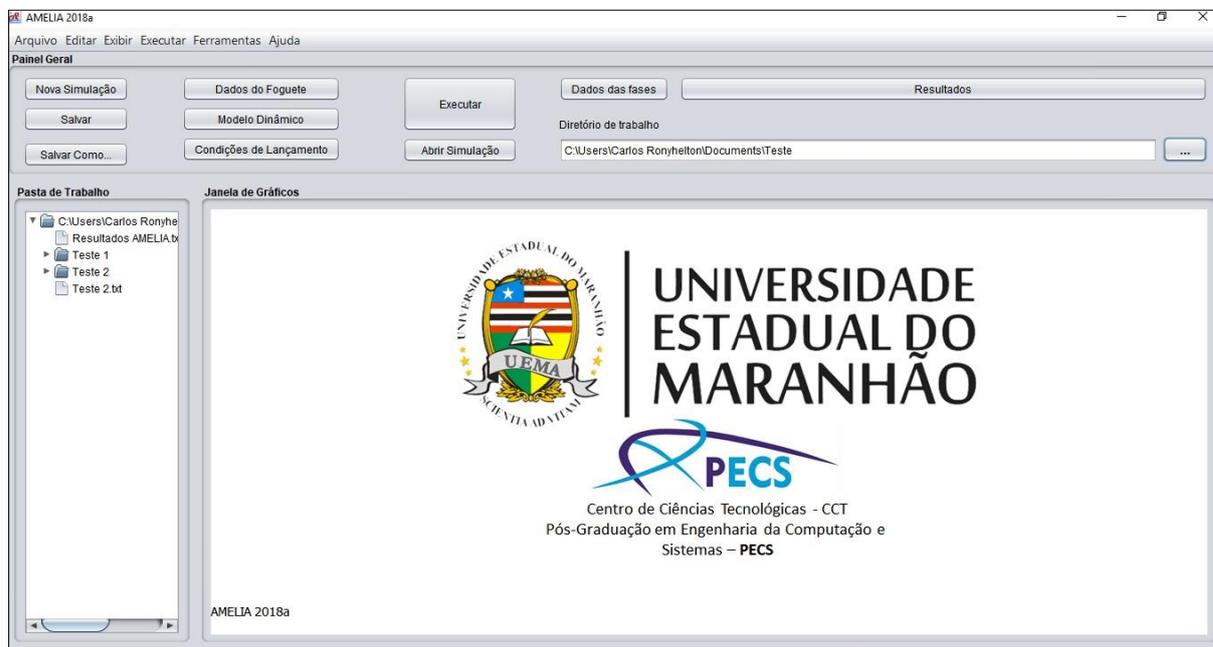
TURNER, A. J. The development and use of open-source spacecraft simulation and control software for education and research. In: SMC-IT 2006: Second IEEE International Conference on Space Mission Challenges for Information Technology. Washington, 2006.

YAMANAKA, F., Análise de Falhas em Modelo Representativo de Sistema Elétrico Proposto para Plataforma de Lançamento de Veículos Espaciais. Dissertação de Mestrado em Engenharia Mecânica – Instituto Tecnológico de Aeronáutica, Brasil, 2006.

ZIPFEL, P. Modeling and simulation of aerospace vehicle dynamics. 2 Ed. Reston,VA: AIAA, 2007.

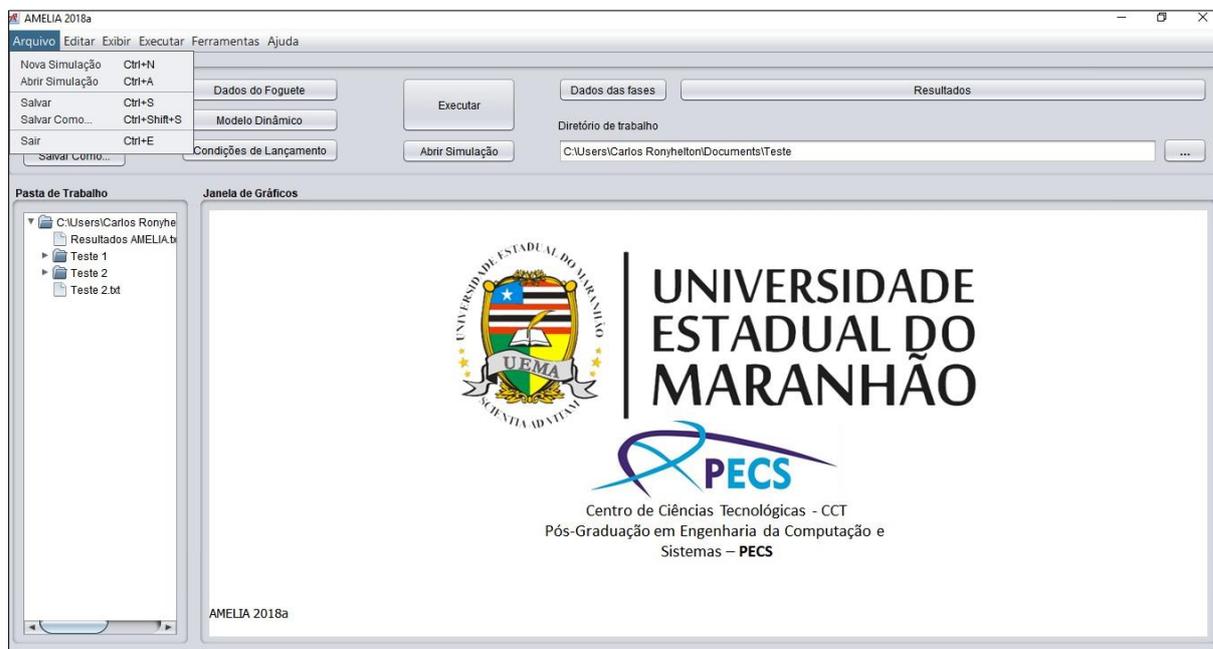
APÊNDICE A - INTERFACES DA PLATAFORMA AMELIA

Figura A.1 – Interface Principal do AMELIA



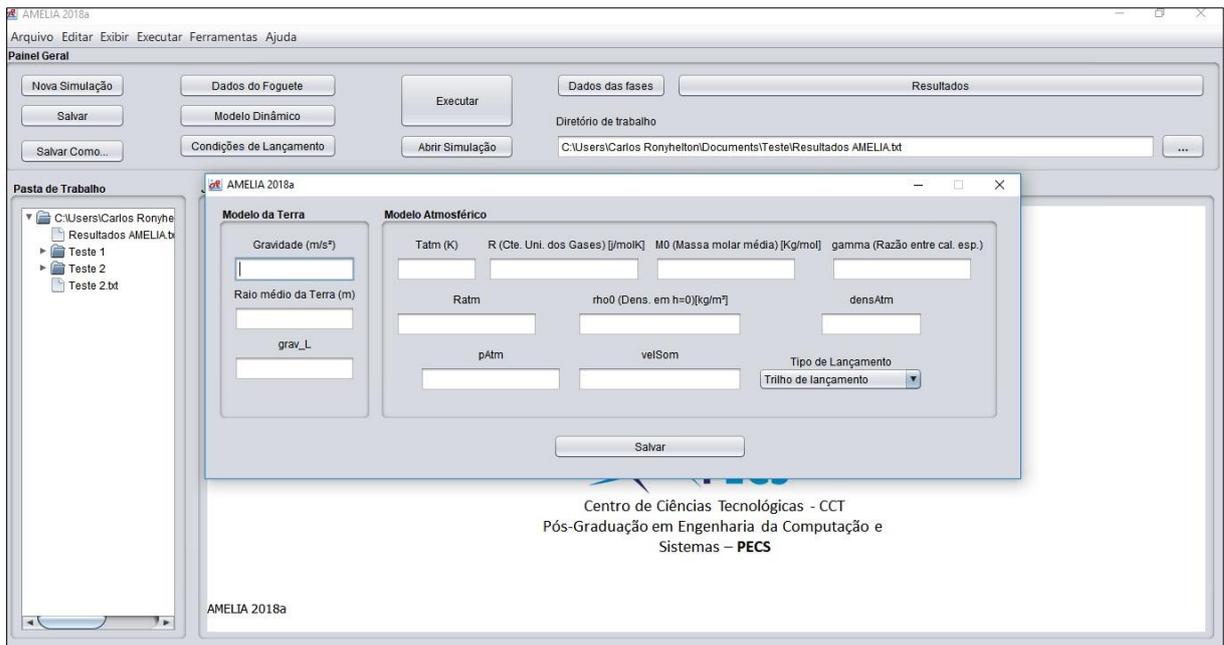
Fonte: Autores (2018)

Figura A.2 – Interface Principal com menu e teclas de atalho exibidos



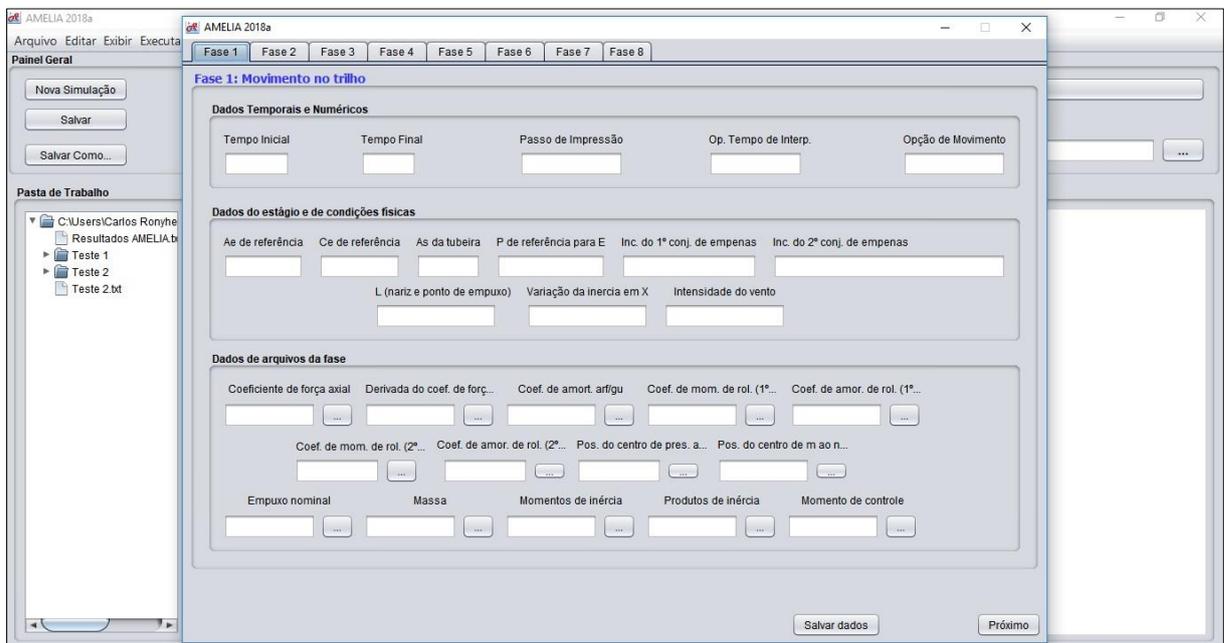
Fonte: Autores (2018)

Figura A.3 - Entrada de dados das condições lançamento



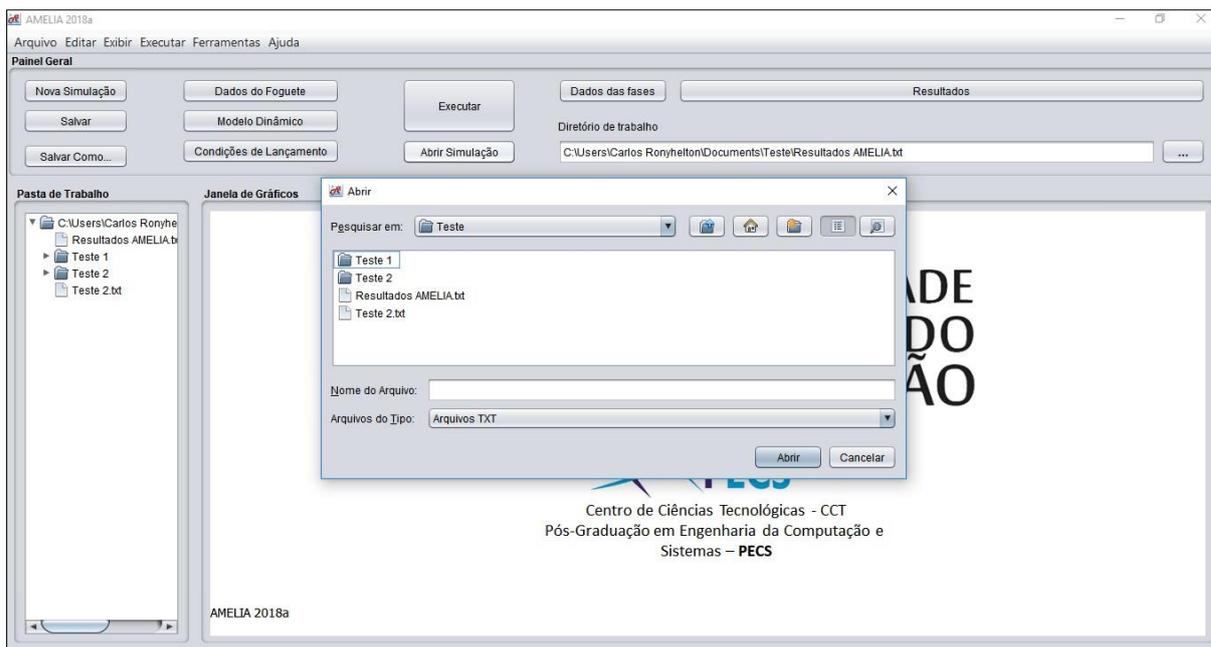
Fonte: Autores (2018)

Figura A.4 - Entrada de dados do foguete



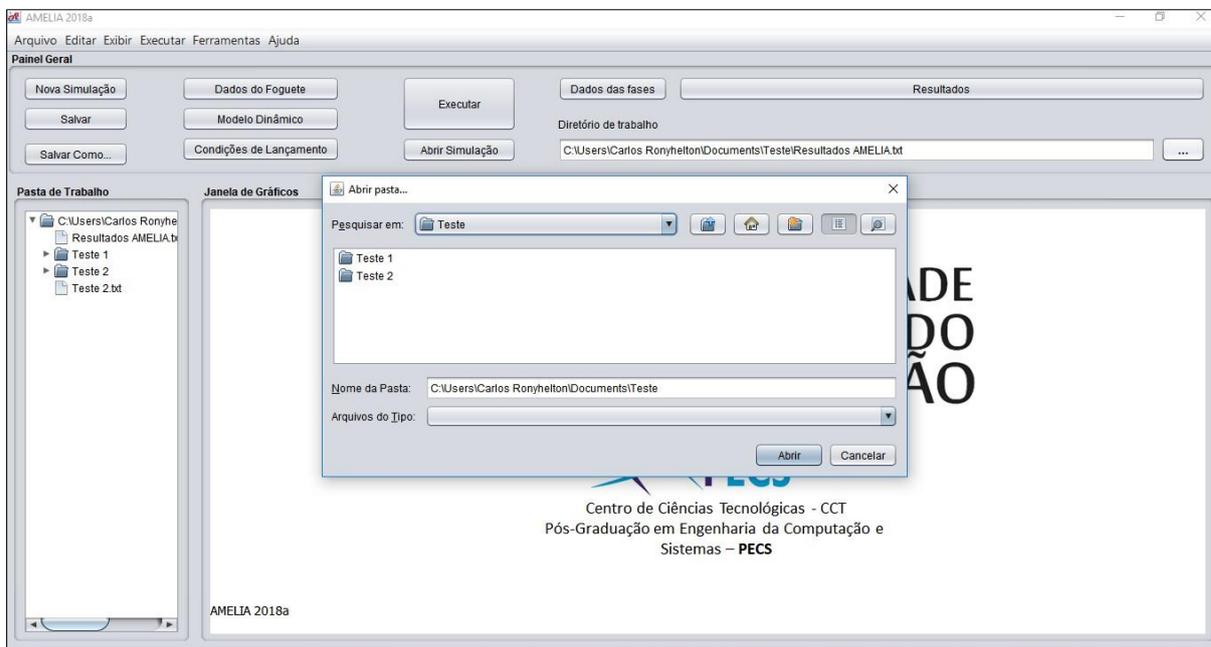
Fonte: Autores (2018)

Figura A.5 – Abrindo nova simulação



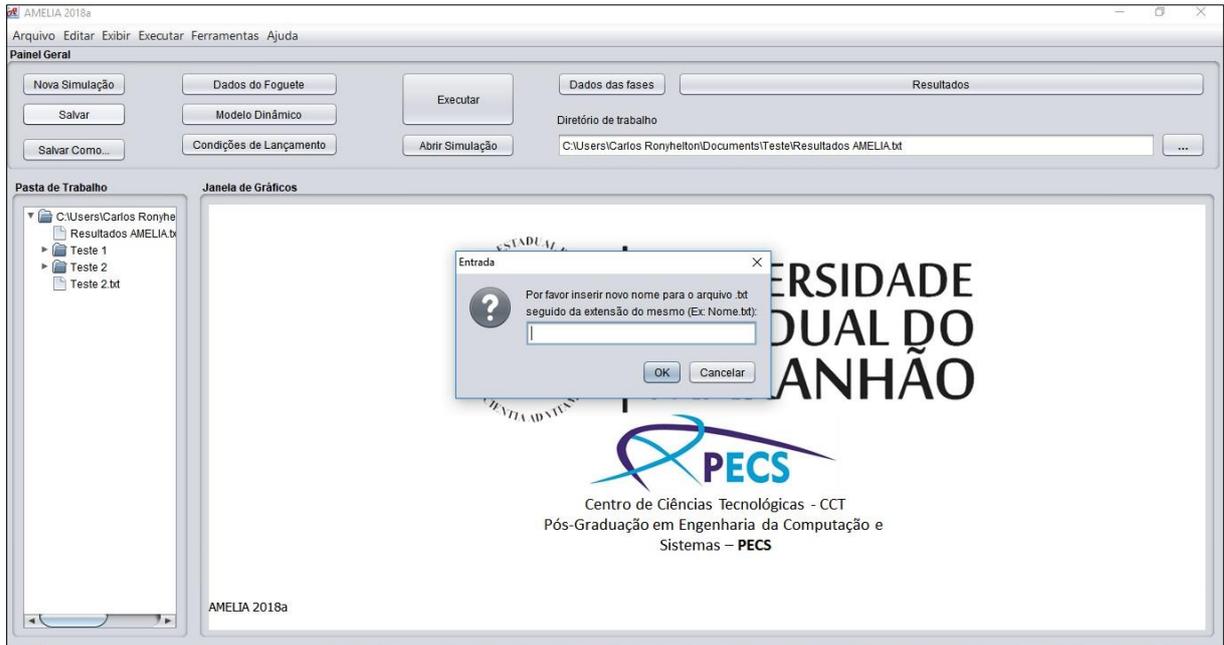
Fonte: Autores (2018)

Figura A.6 – Abrindo nova pasta de trabalho



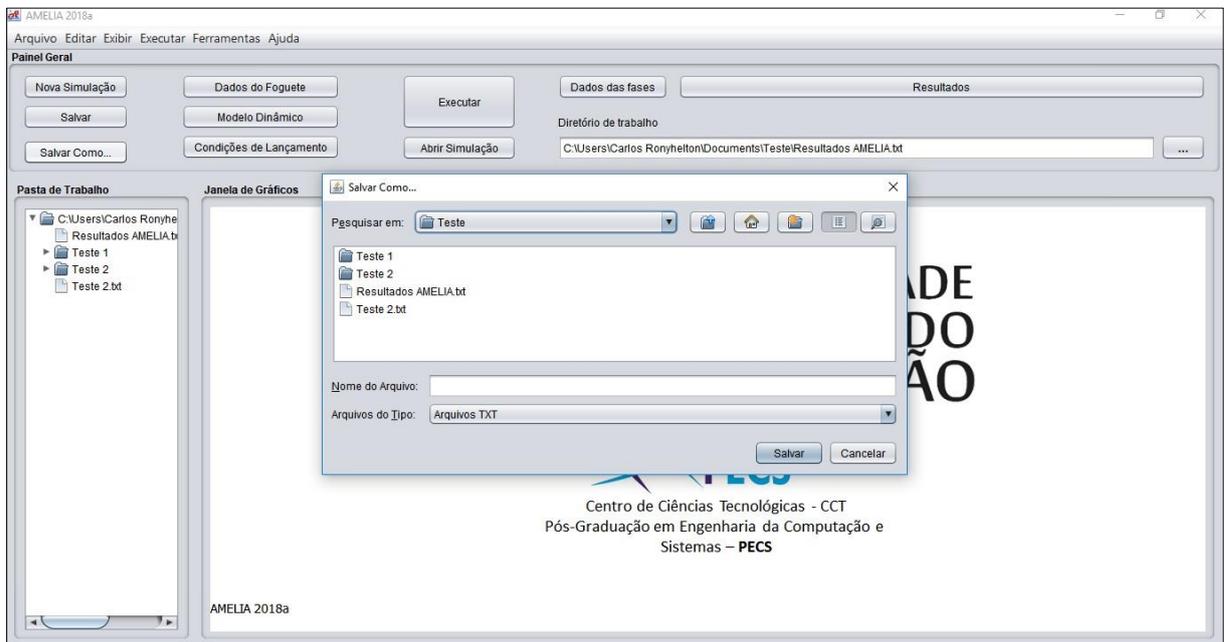
Fonte: Autores (2018)

Figura A.7 – Salvando dados de simulação (Botão Salvar)



Fonte: Autores (2018)

Figura A.8 – Salvando dados de simulação (Botão Salvar como...)



Fonte: Autores (2018)

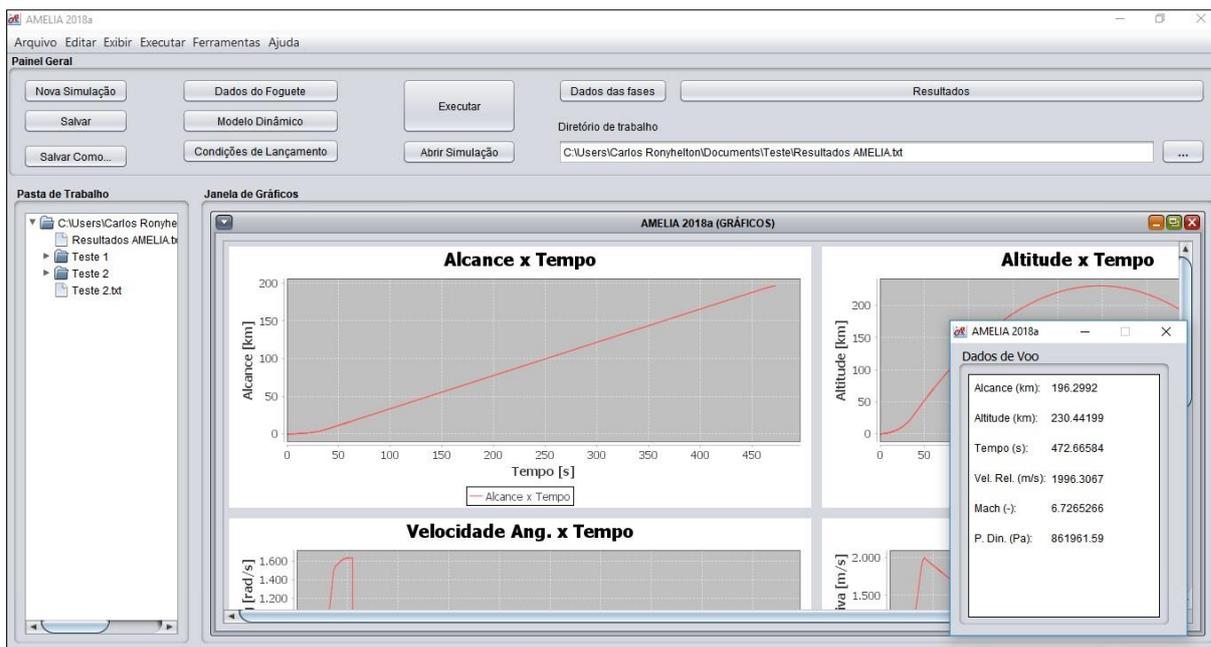
Figura A.9 – Carregando dados de simulação anterior



Fonte: Autores (2018)

Figura A.10 – Deletando dados de simulação



Figura A.11 – Saídas do *Software* AMELIA

Fonte: Autores (2018)