sid.inpe.br/mtc-m19/2013/03.20.14.03 -MAN

ENVIRONMENTAL DISTURBANCE MODELS (EDM)-MANUAL DE USO

Valdemir carrara

Manual.

URL do documento original:
<http://urlib.net/8JMKD3MGP7W/3DP8Q6B>

INPE
São José dos Campos
2013

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/6921
Fax: (012) 3208-6919
E-mail: pubtc@sid.inpe.br

CONSELHO DE EDITORAÇÃO E PRESERVAÇÃO DA PRODUÇÃO INTELECTUAL DO INPE (RE/DIR-204):
Presidente:
Marciana Leite Ribeiro - Serviço de Informação e Documentação (SID)
Membros:
Dr. Antonio Fernando Bertachini de Almeida Prado - Coordenação Engenharia e Tecnologia Espacial (ETE)
Drª Inez Staciarini Batista - Coordenação Ciências Espaciais e Atmosféricas (CEA)
Dr. Gerald Jean Francis Banon - Coordenação Observação da Terra (OBT)
Dr. Germano de Souza Kienbaum - Centro de Tecnologias Especiais (CTE)
Dr. Manoel Alonso Gan - Centro de Previsão de Tempo e Estudos Climáticos (CPT)
Drª Maria do Carmo de Andrade Nono - Conselho de Pós-Graduação
Dr. Plínio Carlos Alvalá - Centro de Ciência do Sistema Terrestre (CST)
BIBLIOTECA DIGITAL:
Dr. Gerald Jean Francis Banon - Coordenação de Observação da Terra (OBT)
REVISÃO E NORMALIZAÇÃO DOCUMENTÁRIA:
Marciana Leite Ribeiro - Serviço de Informação e Documentação (SID)
Yolanda Ribeiro da Silva Souza - Serviço de Informação e Documentação (SID)
EDITORAÇÃO ELETRÔNICA:
Maria Tereza Smith de Brito - Serviço de Informação e Documentação (SID)
Luciana Manacero - Serviço de Informação e Documentação (SID)

sid.inpe.br/mtc-m19/2013/03.20.14.03 -MAN

ENVIRONMENTAL DISTURBANCE MODELS (EDM)-MANUAL DE USO

Valdemir carrara

Manual.

URL do documento original:
<http://urlib.net/8JMKD3MGP7W/3DP8Q6B>

INPE
São José dos Campos
2013

Resumo

Este manual apresenta uma biblioteca de funções destinadas ao cômputo de diversos efeitos perturbadores na órbita e atitude de satélites artificiais terrestres de baixa altitude (abaixo de 1000 km de altitude). Foram implementadas funções para cálculo das forças e torques aerodinâmicos, de pressão de radiação solar, do torque de gradiente de gravidade, da aceleração devido ao geopotencial terrestre, aceleração devido ao Sol e à Lua, e, finalmente, ao torque magnético residual. Os efeitos causados pelo arrasto aerodinâmico e pela pressão de radiação dependem da geometria do satélite. Esta geometria pode ser fornecida por meio de elementos planos, coordenadas de vértices, geometrias pré-definidas como cubos, esferas, cones e cilindros, por meio de descrição por arquivo de dados em Nastran, por uma malha armazenada numa estrutura de vértices e polígonos, ou ainda por meio de qualquer combinação destas formas. O modelo aerodinâmico é baseado na teoria cinética dos gases, enquanto que a pressão de radiação solar é baseada na troca de quantidade de movimento dos fótons com a superfície do satélite. O modelo do geopotencial utilizado é o EGM 2008, de ordem 2159. A biblioteca pode ser facilmente integrada a um ambiente de simulação de órbita ou atitude, a programas destinados ao projeto e dimensionamento de sistemas de controle de órbita e atitude, ou podem ser pré-compiladas e integradas ao ambiente Matlab. Adicionalmente, foram produzidas funções para geração de arquivos com a geometria do satélite, quando esta for descrita por meio de triângulos, quadriláteros ou malhas poligonais. Estes arquivos são posteriormente utilizados para visualização da geometria por um programa de computação gráfica tridimensional. Este manual apresenta diversos exemplos de uso, com código fonte, e uma aplicação das funções ao satélite CBERS (China-Brasil Earth Remote Sensing).

Environmental Disturbance Models (EDM) – User's Manual

Abstract

This manual presents a library for computation of several environmental disturbing effects (EDM – Environmental Disturbance Models) on attitude and orbit of artificial satellites at low Earth orbit (below 1000 km altitude). Several functions were created so as to compute the aerodynamic as well as the solar radiation pressure forces and torques, the gravity gradient torque, satellite acceleration due to Earth, Sun and Moon gravitational fields, and the torques due to the satellite residual magnetic moment. The effects due to the aerodynamic drag and due to the radiation pressure depend on the satellite geometry. This can be provided by functions that describe the geometry either by means of a set of flat elements, vertices coordinates, preprogrammed geometries (such as cubes, spheres, cones and cylinders), or by a description data file in Nastran, by a mesh structure stored as a vertex and polygons table, or still by any combination of these forms. The aerodynamic model is based on the kinetic theory of gases, while solar radiation pressure is based on the exchange of momentum of the photons with the satellite external surface. The geopotential model uses a 2159th order EGM 2008 model. The EDM library can be easily integrated into an orbit and attitude simulator, or employed for Attitude and Orbit Control Subsystem (AOCS) design, and can be easily pre-compiled and integrated with Matlab. Additionally, some functions were provided in order to generate a geometry description file that can be used to check the geometry correctness whenever the satellite is described by means of triangles, quadrilaterals or polygonal meshes. A three-dimensional computer graphics program then uses these files so as to render the geometry. This manual presents several application examples of the functions, including the source code of the examples, and a complete application of the functions to CBERS (China-Brazil Earth Remote Sensing).

Sumário

Pág.

1 INTRODUÇÃO 1

2 FUNÇÕES CONJUNTAS 9

3 FUNÇÕES PARA CÁLCULO AERODINÂMICO 13

4 FUNÇÕES PARA CÁLCULO DE PRESSÃO DE RADIAÇÃO 23

5 ESTRUTURAS PARA DESCRIÇÃO DA GEOMETRIA DO SATÉLITE 33

6 FORMATO DO ARQUIVO DE DESCRIÇÃO DA GEOMETRIA 39

7 FUNÇÕES PARA LEITURA DO ARQUIVO COM A DESCRIÇÃO DA GEOMETRIA 45

8 FUNÇÕES PARA CÁLCULO DO TORQUE DE GRADIENTE DE GRAVIDADE 55

9 FUNÇÕES PARA CÁLCULO DO TORQUE MAGNÉTICO RESIDUAL 57

10 FUNÇÕES PARA CÁLCULO DA ACELERAÇÃO LUNI-SOLAR 59

11 FUNÇÕES PARA CÁLCULO DA ACELERAÇÃO DO POTENCIAL GRAVITACIONAL TERRESTRE 61

12 EXEMPLOS DE UTILIZAÇÃO DAS FUNÇÕES 63

12.1 Rotação de geometria 63
12.2 Arrasto aerodinâmico 66
12.3 Pressão de radiação solar 71
12.4 Torque de gradiente de gravidade 76
12.5 Torque magnético 77
12.6 Aceleração Luni-Solar 78
12.7 Aceleração do geopotencial 80

BIBLIOGRAFIA 83

APÊNDICE A 85

APÊNDICE B 91

1 INTRODUÇÃO

O projeto de sistemas para controle de atitude de satélites artificiais é bastante complexo em virtude da alta complexidade dos equipamentos envolvidos no controle e a alta confiabilidade requerida do sistema. É imprescindível, portanto, que o sistema de controle seja projetado de forma a garantir sua funcionalidade em quaisquer circunstâncias. Dentre as diversas ferramentas que o projetista dispõe, destacam-se as ferramentas de simulação, que permitem simular não apenas o comportamento dinâmico do satélite no espaço, mas também o próprio ambiente espacial e sua interação com o satélite. Para dimensionar o sistema de controle, isto é, para estabelecer as características técnicas necessárias aos sensores e atuadores deste sistema, emprega-se usualmente uma simulação acurada da ação do controle sob as principais perturbações encontradas no ambiente espacial. Fica claro, portanto, a importância que um módulo para simulação das perturbações desempenha no projeto de sistemas de controle. Este módulo, denominado de Modelo de Perturbação Ambiental, ou "Environmental Disturbance Models" (EDM), implementa diversas funções para cálculo de perturbações ambientais em satélites, para uso em simuladores e propagadores de atitude e órbita. A presente versão conta com modelos para:

- Força e torque aerodinâmicos
- Força e torque de pressão de radiação solar
- Torque de gradiente de gravidade
- Aceleração luni-solar
- Aceleração do potencial terrestre
- Torque magnético

Este módulo foi desenvolvido de forma a ser facilmente unido ao módulo de simulação de atitude (Carrara, 2011). Em virtude deste aspecto, algumas variáveis de entrada e saída das funções necessitam de um cabeçalho de definição de tipos e protótipos, encontrados em `matrices.h` e `matrices.cpp`. Estes tipos são: `vector3`, `vector6`, `matrix3`, e `quaternion`.

Historicamente, esta biblioteca de funções iniciou-se na década de 1980, quando o autor deste documento implentou em Fortran um programa para cálculo de forças e torques ambientais atuantes no satélite que se tornaria o SCD1, como parte da sua dissertação de mestrado (CARRARA, 1982), sob orientação do prof. Nellore Srinivasan Venkataraman, que, na época, havia sido recentemente contratado pelo INPE. Ainda naquela década, o pacote foi aperfeiçoado durante uma estadia na empresa SPAR do Canadá. Na nova versão (CARRARA, 1988), também em Fortran, foi incorporada uma descrição da geometria por meio de arquivo em Nastran, e a adição de um programa para cômputo da variação de quantidade de momento linear ou angular provocada pelas perturbações ambientais. Os resultados obtidos com este programa permitiriam projetar sistemas de controle com maior nível de precisão. Este programa continuou em uso até 1992, quando os computadores pessoais (PC) passaram a substituir os computadores de grande porte. Embora houvesse a disponibilidade de compiladores Fortran para PC, não houve a necessária migração daquele pacote até recentemente, quando se decidiu estudar e calcular o centro de pressões do satélite CBERS. A dificuldade de conversão do pacote para execução em PC, aliada ao fato de que havia a necessidade de uma revisão dos parâmetros e do cálculo efetuado por diversas funções fez com que se buscasse por uma nova implementação em C, para que ficasse compatível com o simulador de atitude implementado também em C pelo autor (CARRARA, 2007). Nesta versão, além do cômputo dos efeitos aerodinâmicos e de pressão de radiação, foram

incorporados modelos para as demais forças e perturbações, como o gradiente de gravidade, geopotencial, magnético, e aceleração luni-solar.

O cálculo das forças e torques aerodinâmicos e de pressão de radiação solar depende da descrição da geometria do satélite. Esta descrição é baseada no conceito de representação por fronteira, ou *b-reps* (de "boundary-representations") que compreendem descrições da superfície por meio de equações matemáticas ou por meio de polígonos planos. A representação adotada aqui foi a de polígonos planos, semelhante àquela do OpenGl. A superfície, então, é descrita por meio de polígonos definidos pelos seus vértices. Superfícies curvas são aproximadas por subdivisões em pequenos triângulos ou quadriláteros planos. As forças elementares, isto é, que agem num elemento de área plano são computadas em função da normal ao elemento **n** e da sua área d*A* (funções `edm_aerodynamic_plane` e `edm_solar_prsr_plane`). Como pode ser visto na Figura 1.1, sobre o elemento plano de área d*A* pode ser construído um sistema de eixos *x-y-z* tal que uma dada direção **v** (velocidade do satélite ou direção do Sol) situa-se no plano *yz*. É neste sistema que as componentes da força elementar são obtidas. Não há restrição sobre a direção de **v**, e convenciona-se que **n** representa a normal externa. As funções `edm_set_environ_properties`, `edm_set_drag_properties`, e `edm_set_solar_rad_properties` permitem estabelecer o valor do vetor **v**. Tanto a normal quanto a área podem ser obtidas a partir das coordenadas dos vértices de um quadrilátero ou de um triângulo. As funções `edm_aerodynamic_triangle`, `edm_aerodynamic_quad`, `edm_solar_prsr_triangle` e `edm_solar_prsr_quad` calculam as forças elementares a partir dos vértices, tal que **n** segue a regra da mão direita na seqüência dos vértices fornecidos às funções. A direção da normal pode ser selecionada, caso necessário, por meio da função `edm_set_surface_side`. Seja por meio do vetor normal e da área, ou então por meio dos vértices, a força aerodinâmica e de pressão de radiação necessita ser integrada (acumulada) sobre toda a superfície externa do satélite. Ressalta-se que a integração deve ser avaliada ao longo do tempo, para permitir a correta avaliação dos seus efeitos sobre a órbita e atitude. Cabe ao usuário efetuar esta integração, já que ela é dependente da geometria de cada satélite em particular. Para facilitar esta integração foram implementadas diversas funções que permitem configurar e armazenar a geometria do satélite em estruturas construídas especialmente para isso. As funções `edm_aerodynamic_mesh` e `edm_solar_prsr_mesh` integram a força e o torque a partir da estrutura que armazena os vértices, as faces e as características superficiais do satélite. Estas três formas distintas de cálculo são mostradas na Figura 1.2. De certa forma, elas não são independentes, uma vez que as funções de nível superior utilizam as funções de nível inferior. Finalmente, foram implementadas funções para cálculo das forças e torques com integração analítica em geometrias simples, como esfera (`edm_aerodynamic_sphere`, `edm_solar_prsr_sphere`), cilindro (`edm_aerodynamic_cylinder`, `edm_solar_prsr_cylinder`), cone (`edm_solar_prsr_cone`), e paralelepípedo (`edm_aerodynamic_box`, `edm_solar_prsr_box`). A força aerodinâmica no cone (`edm_aerodynamic_cone`), é realizada por uma integral numérica, já que não existe solução analítica para seu cálculo. Para estas geometrias as forças são calculadas para uma incidência externa, isto é, as faces internas não são consideradas no cálculo. Há ainda a possibilidade de cálculo das forças e torques num setor esférico (`edm_aerodynamic_sphere_sector`, `edm_solar_prsr_ sphere_sector`), com integração numérica simples (somatória) num setor definido por longitudes e latitudes de início e fim do setor, permitindo também a configuração do número de divisões do setor. Setores esféricos são geralmente empregados para modelar geometricamente antenas parabólicas. Todas estas funções são representadas graficamente na Figura 1.3.

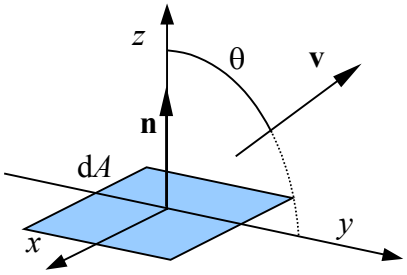

Fig. 1.1 – Sistema de coordenadas fixado ao elemento de área.

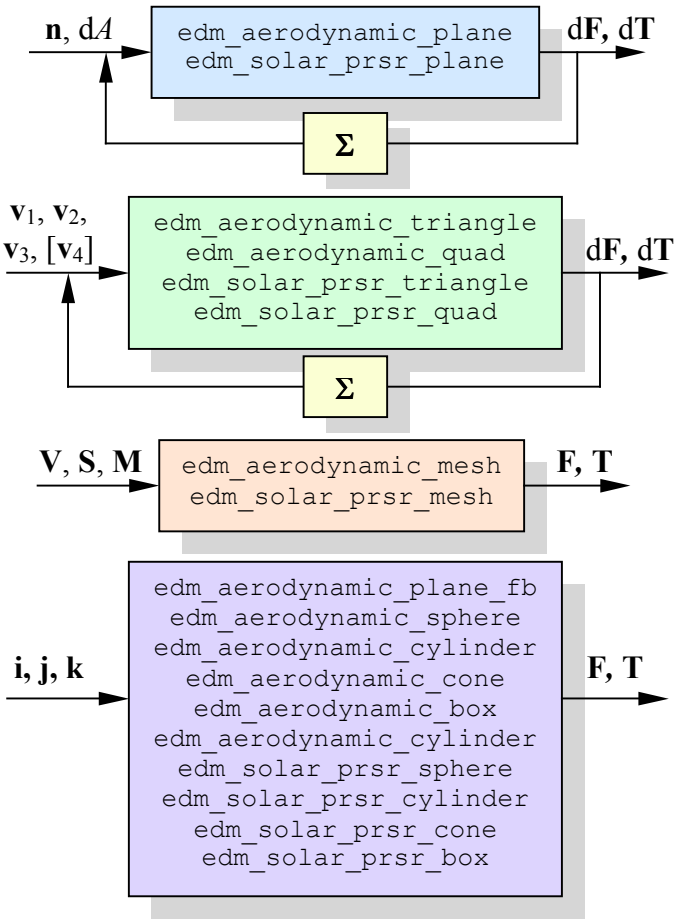

Fig. 1.2 – Funções para cálculo das forças e torques aerodinâmicos e de pressão de radiação solar.

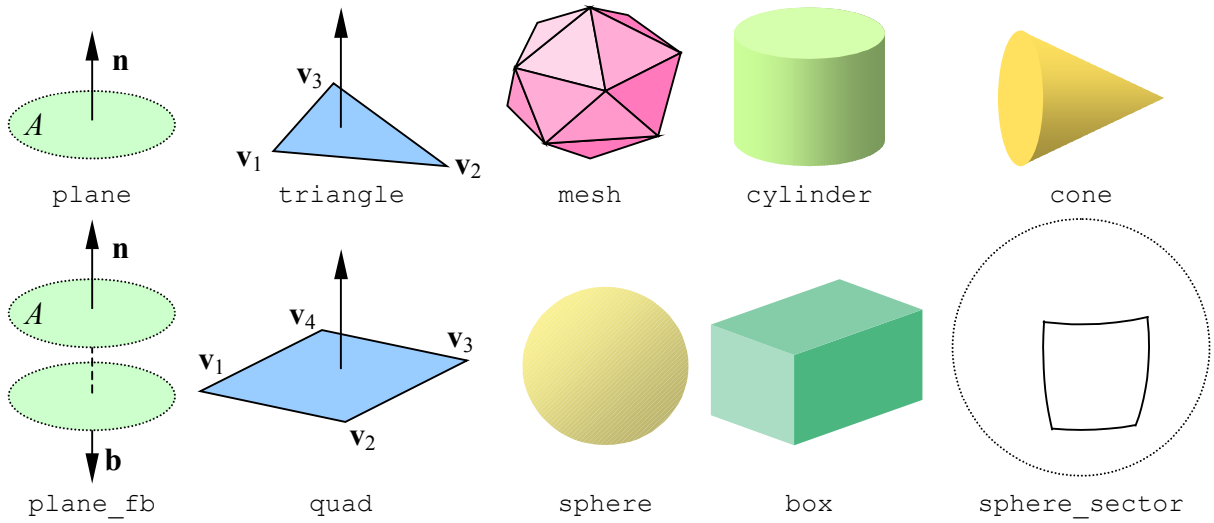

Fig. 1.3 – Geometrias implementadas para cálculo de forças e torques ambientais

Não é intenção descrever os aspectos teóricos ou o equacionamento utilizado nos cálculos das forças e torques ambientais aqui. Toda a teoria pode ser encontrada facilmente na literatura específica (CARRARA, 1982 e 2013, NASA, 1969, SINGER, 1974, WERTZ, 1978). Menciona-se, contudo, que o modelo aerodinâmico emprega a teoria cinética dos gases rarefeitos, a pressão de radiação utiliza modelos simplificados de reflexão, o torque de gradiente de gravidade baseia-se na assimetria de massa presente no satélite e o torque magnético é dado pelo produto vetorial entre o momento magnético e o campo magnético terrestre. Para se obter as equações das forças e torques aerodinâmicos e de pressão de radiação, foi necessário adotar a hipótese de que cada elemento de área seja receba um fluxo de moléculas livre (provindas de todas as direções) ou de radiação solar de uma única fonte de luz. Isto significa que o equacionamento é válido apenas nos objetos convexos, isto é, nos quais nenhum elemento de área consegue "ver" o outro. Esferas, cones, cilindro e paralelepípedos são todos convexos. Um toróide, por outro lado, é côncavo. Um cúbico com painéis laterais é também côncavo em virtude destes painéis. Esta restrição invalidaria a aplicação da teoria na quase totalidade dos satélites existentes, não fosse o fato de que o efeito provocado pela "sombra" ou pelo "reflexo" de uma superfície sobre outras superfícies ser pequeno. Portanto, sempre que a geometria do satélite for côncava, a teoria pode ainda ser utilizada, porém deve-se estar atento de que o cálculo será apenas aproximado e conterá erros.

Tanto o cálculo aerodinâmico quanto o da pressão de radiação dependem da temperatura do elemento de superfície. Na verdade, a força aerodinâmica depende de

$$\Gamma = \sqrt{\frac{T_w}{T_i}}\,, \tag{1.1}$$

onde T_w é a temperatura absoluta do elemento e T_i a temperatura absoluta da atmosfera no local onde se encontra o satélite. A temperatura T_i pode ser obtida em modelos da termosfera (CARRARA, 1990). Em algumas situações é preferível fornecer diretamente o valor já calculado de Γ a ter que especificar individualmente a temperatura de cada elemento. Para manter o cálculo coerente, algumas regras devem, portanto, ser obedecidas:

1) O valor de Γ atribuído pela função `edm_set_drag_surface` (variável `sq_temp_ratio`) não é afetado quando T_i for fornecido pela função `edm_set_environ_properties` (variável `atmo_temperature`). Neste caso T_i é considerado para o cálculo da razão de velocidades (parâmetro `speed_ratio` da função `edm_set_drag_properties`).

2) Caso a temperatura do elemento T_w seja conhecida, então cabe ao utilizador calcular Γ com base na definição, desde que T_i seja também conhecido. Novamente, Γ deve ser atribuído pela função `edm_set_drag_surface`.

3) Quando for usada a descrição da geometria por meio de arquivo (veja-se o Capítulo 6), a temperatura do elemento deverá necessariamente ser provida no arquivo, e, neste caso, o cálculo de Γ será realizado internamente e levará em conta o valor de T_i atribuído previamente pela função `edm_set_environ_properties` ou então seu valor "default".

Para esclarecer um pouco mais o motivo deste arranjo de parâmetros, considera-se o fato de que o cálculo do arrasto depende das seguintes variáveis: coeficientes de troca de quantidade de movimento, σ_n e σ_t, razão de temperaturas Γ, direção da velocidade do satélite $\hat{\mathbf{u}}$, pressão aerodinâmica p e razão de velocidades s. Os três últimos podem ser calculados a partir de:

$$\hat{\mathbf{u}} = \frac{\mathbf{u}}{u}, \text{ com } u = |\mathbf{u}| \tag{1.2}$$

$$p = \frac{1}{2}\rho \mathbf{u}^2, \tag{1.3}$$

$$s = u\sqrt{\frac{M}{2RT_i}}, \tag{1.4}$$

onde ρ é a densidade local da atmosfera, $\mathbf{u}$ é o vetor velocidade do satélite, u o seu módulo, M é a massa molecular média local da atmosfera, e R é a constante universal dos gases ($R \approx 8314$ J/K kgmol). Estas relações são utilizadas pela função `edm_set_environ_properties` que repassa o cálculo para `edm_set_drag_properties`. A Tabela 1.1 identifica a correspondência entre as variáveis matemáticas e os parâmetros utilizados nas funções. Portanto, a temperatura local da atmosfera é usada tanto na avaliação de Γ quanto na razão de velocidades s. Porém, estes dois parâmetros podem ser fornecidos de forma independente, para assegurar flexibilidade de uso. Em resumo, há duas formas de se calcular o arrasto: pelo fornecimento das temperaturas T_w e T_i, ou então pela definição direta de Γ e s.

O cálculo das forças e torques de pressão de radiação segue um procedimento semelhante ao do aerodinâmico. Para o cálculo das forças de pressão de radiação, é necessário que se conheça o vetor de direção do Sol e a pressão de radiação solar. Se $\mathbf{r}_s$ for o vetor que representa a posição do centro do Sol no sistema de coordenadas fixado ao satélite, então a direção do Sol é a normalização deste vetor, enquanto que a pressão de radiação p_s é dada pela relação entre a constante solar S no ponto onde se encontra o satélite e a velocidade da luz no vácuo, c, ou seja:

$$\hat{\mathbf{r}}_s = \frac{\mathbf{r}_s}{r_s}, \quad r_s = |\mathbf{r}_s|, \tag{1.5}$$

$$p_s = \frac{S}{c}. \tag{1.6}$$

Tabela 1.1 –Variáveis e parâmetros utilizados no cálculo aerodinâmico

Variável	Parâmetro
σ_n	`sigma_normal`
σ_t	`sigma_tang`
Γ	`sq_temp_ratio`
$\hat{\mathbf{u}}$	`un_sat_vel`
$\mathbf{u}$	`sat_velocity`
p	`aerodynamic_pressure`
s	`speed_ratio`
ρ	`atmo_density`
M	`mean_molecular_weight`
T_i	`atmo_temperature`

A pressão de radiação varia com a distância da Terra ao Sol ao longo do ano em cerca de 7%. A constante solar média (quando a distância Terra-Sol é igual a uma unidade astronômica, r_{1UA} = 149,6 10^9 metros) é de cerca de 1353 W/m^2, e com isso a pressão de radiação nesta distância fica p_{1UA} = 4,5131 10^{-6} N/m^2. A pressão de radiação a uma distância r_s do Sol fica então:

$$p_s = p_{1UA}\left(\frac{r_{1UA}}{r_s}\right)^2 \tag{1.7}$$

A direção do Sol e a pressão de radiação podem ser fornecidas alternativamente por meio de duas funções: `edm_set_solar_rad_properties` e `edm_set_solar_rad_pressure`. A função `edm_set_solar_rad_pressure` permite que os valores de $\hat{\mathbf{r}}_s$ e p_s sejam estabelecidos para os cálculos que se seguirem. Já a função `edm_set_solar_rad_properties` tem como argumento a posição do Sol no sistema do satélite, $\mathbf{r}_s$, e calcula o versor direção do Sol e a pressão de radiação a partir das equações mostradas acima. Quando se deseja obter o coeficiente de pressão de radiação, e não a força, deve-se estabelecer que a pressão de radiação seja unitária. A Tabela 1.2 apresenta as variáveis empregadas no cálculo da pressão de radiação. A temperatura do elemento de superfície do satélite, T_w, é utilizada no cálculo da força devido à emissão de radiação pelo satélite, de acordo com a lei de Stefan-Boltzmann.

Foram criadas funções para atribuir características para os elementos de área a serem utilizados no cálculo das forças e torques, funções para recuperar estes valores, funções para selecionar o lado da superfície considerada no cálculo, funções para atribuir valores aos parâmetros ambientais, para calcular as forças e torques num elemento triangular e num quadrilátero. As funções contendo integrações analíticas das forças e torques em geometrias simples são mais rápidas do que uma integração numérica, mas para isso admitem que as

propriedades da superfície (coeficientes de acomodação e reflexão, além da temperatura) não variem ao longo da geometria, o que impede simulações mais realistas. Qualquer que seja a geometria do satélite, pode-se usar uma ou mais funções para a integração das forças ambientais, desde que esta geometria possa ser decomposta naquelas já implementadas nas funções, como cilindros, esferas e paralelepípedos. Deve-se, contudo, garantir que o ponto de atuação dos torques oriundos das geometrias básicas coincida em todos os cálculos individuais.

Tabela 1.2 –Variáveis e parâmetros utilizados no cálculo da pressão de radiação

Variável	Parâmetro
e	`specular_coef`
d	`diffuse_coef`
ε	`emmitance`
T_w	`surf_temperature`
$\hat{\mathbf{r}}_s$	`un_sun_direction`
$\mathbf{r}_s$	`sun_vector`
p_{1UA}	`rad_pressure_au`

A integração analítica da força aerodinâmica num cilindro resulta numa expressão que depende das funções modificadas de Bessel do primeiro tipo, I_0 e I_1. O código para estas funções foi obtido a partir de uma versão em C de Jean-Pierre Moreau (MOREAU, 2012), baseado em algoritmos apresentados em livros de análise numérica. Este código foi incorporado à biblioteca EDM. As propriedades da atmosfera podem ser obtidas a partir de diversos modelos disponíveis na literatura. Recomendam-se as funções disponíveis em Carrara (1990, 2012), pois a interface é plenamente compatível com esta biblioteca. A posição do Sol, da Lua e o valor do campo geomagnético podem ser recuperados da biblioteca contida no pacote `orbit.cpp`, cuja documentação pode ser vista em Carrara (2007). A aceleração devida ao geopotencial terrestre é calculada usando o modelo Earth Gravitational Model 1996 (EGM, 2012), codificado por Kuga em Fortran e convertido para C por este autor para uso na biblioteca EDM.

Na descrição que segue, os valores "default" dos argumentos das funções são indicados entre barras, como /0/, /5/, /3.14/, etc. Caso a função para atribuição de valores não seja utilizada pelo programador, prevalece o valor "default" para a variável. As funções e seus parâmetros são descritos nas seções que seguem.

O Capítulo 2 irá apresentar algumas funções que são comuns ao cálculo aerodinâmico e da pressão de radiação. O Capítulo 3 aborda as funções desenvolvidas para o arrasto atmosférico, enquanto que o Capítulo 4 abrange as funções associadas com a pressão de radiação. As estruturas desenvolvidas para armazenar geometrias do tipo malha são apresentadas no capítulo seguinte. O Capítulo 6 irá mostrar o formato de armazenagem de geometria em arquivos e os comandos associados à geometria, ao passo que as funções implementadas para efetuar a leitura deste arquivo são descritas no Capítulo 7. Os Capítulos 8 a 11 apresentam, respectivamente, as funções para cálculo do torque de gradiente de gravidade, o torque magnético residual, da aceleração luni-solar e da aceleração devido ao geopotencial. Finalmente, no Capítulo 12, são apresentados exemplos de utilização de cada um dos modelos e funções. O Apêndice A trás o código de visualização da geometria em POV (POVRAY, 2013), e o Apêndice B trás a relação das funções desenvolvidas, separadas por finalidade.

2 FUNÇÕES CONJUNTAS

Agrupam-se aqui algumas funções que servem tanto para a configuração de parâmetros utilizados no cálculo das forças aerodinâmicas quanto à pressão de radiação.

- **`void edm_set_surface_side (unsigned char surfaceside);`**

Esta função permite selecionar o lado do elemento superficial que será empregado no cálculo da força e torque agindo sobre ele. Esta função afeta o cálculo realizado pelas funções `edm_aerodynamic_triangle`, `edm_aerodynamic_quad`, `edm_solar_prsr_triangle`, `edm_solar_prsr_quad`, `edm_aerodynamic_sphere_sector` e `edm_solar_prsr_sphere_sector`.

Parâmetros de entrada:

`surfaceside`

Variável que pode assumir os valores:

1 ou FRONT ou OUTSIDE: neste caso o lado considerado será o mesmo lado da normal externa ao elemento, que é computado com base no produto vetorial das arestas, e segue a regra da mão direita.

2 ou BACK ou INSIDE: neste caso o lado considerado será oposto à normal do elemento.

0 ou FRONT_BACK ou OUTSIDE_INSIDE: o cálculo da força e torque elementar irá considerar ambas as faces do elemento.

Qualquer valor diferente de 0, 1 ou 2 será considerado como sendo 1.

Valor default: /1/

Parâmetros de saída:

Não há.

- **`unsigned char edm_get_surface_side();`**

Esta função informa o lado correntemente empregado no cálculo da força e torque agindo sobre o elemento superficial.

Parâmetros de entrada:

Não há.

Parâmetros de saída:

`char edm_get_surface_side`

Variável que assume os valores:

1 ou FRONT ou OUTSIDE: neste caso o lado considerado é o da normal externa ao elemento.

2 ou BACK ou INSIDE: neste caso o lado considerado é oposto ao vetor normal do elemento.

0 ou FRONT_BACK ou OUTSIDE_INSIDE: o cálculo da força e torque elementar considera ambas as faces do elemento.

- **`double edm_erf(double x);`**

A função `edm_erf` calcula a função erro do argumento `x`.

Parâmetros de entrada:

`x`

Argumento da função erro

Parâmetros de saída:

`edm_erf`

Função erro ("error function").

- **`double BESSI0 (double X);`**

Cálculo da função modificada de Bessel de ordem 0, $I_0(x)$.

Parâmetros de entrada:

`X`

Argumento da função de Bessel (`double`)

Parâmetros de saída:

`BESSI0`

Valor da função de Bessel modificada de ordem 0 avaliada no argumento `X`.

- **`double BESSI1 (double X);`**

Cálculo da função modificada de Bessel de ordem 1, $I_1(x)$.

Parâmetros de entrada:

`X`

Argumento da função de Bessel (`double`)

Parâmetros de saída:

`BESSI1`

Valor da função de Bessel modificada de ordem 1 avaliada no argumento `X`.

- **`double BESSI (int N, double X);`**

Cálculo da função modificada de Bessel de ordem n, $I_n(x)$.

Parâmetros de entrada:

`X`

Argumento da função de Bessel (`double`)

Parâmetros de saída:

`BESSI`

Valor da função de Bessel modificada de ordem n avaliada no argumento `X`.

3 FUNÇÕES PARA CÁLCULO AERODINÂMICO

As funções descritas aqui se destinam ao fornecimento de parâmetros ambientais, características superficiais, descrição da geometria e cálculo das forças e torques aerodinâmicos em diversas geometrias.

- **`void edm_set_drag_surface (double sigma_normal, double sigma_tang, double sq_temp_ratio);`**

  Esta função irá atribuir as características atuais da superfície, a serem utilizadas no cálculo subseqüente das forças e torques aerodinâmicos que agem nos elementos de área. Esta função deve ser empregada quando somente forças aerodinâmicas forem calculadas, ou quando houver alteração das características aerodinâmicas do elemento de superfície.

  Parâmetros de entrada:

  `sigma_normal`

  Coeficiente de troca de quantidade de movimento das moléculas da atmosfera com a superfície do satélite na direção normal à superfície, σ_n. /1.0/

  `sigma_tang`

  Coeficiente de troca de quantidade de movimento das moléculas da atmosfera com a superfície do satélite na direção normal à superfície, σ_t. /1.0/

  `sq_temp_ratio`

  Raiz quadrada da relação entre a temperatura absoluta da superfície, T_w, e a temperatura absoluta local da atmosfera, T_{loc}, ambas em graus Kelvin. /0.5/

  Parâmetros de saída:

  Não há.

- **`vector3 edm_get_drag_surface();`**

  Esta função irá recuperar as características aerodinâmicas atuais da superfície.

  Parâmetros de entrada:

  Não há.

  Parâmetros de saída:

  `edm_get_drag_surface`

  Variável do tipo `vector3` contendo, respectivamente, o coeficiente de troca de quantidade de movimento na direção normal (`sigma_normal`), o coeficiente de troca de quantidade de movimento das moléculas da atmosfera com a superfície na direção tangencial (`sigma_tang`) e a raiz quadrada da relação de temperaturas (`sq_temp_ratio`).

- **`void edm_set_environ_properties (vector3 sat_velocity, double atmo_density, double atmo_temperature, double mean_molecular_weight);`**

A função `edm_set_environ_properties` atribui os parâmetros ambientais necessários para o cálculo das forças e torques aerodinâmicos. Esta função sobrescreve os valores previamente atribuídos pela função `edm_set_drag_properties`.

Parâmetros de entrada:

`sat_velocity`

Vetor do tipo `vector3`, contendo a velocidade do satélite em relação à atmosfera, em coordenadas do sistema de eixos fixado ao satélite, em metros por segundo.

`atmo_density`

Densidade local da atmosfera, em kg/m^3. /10^{-12}/

`atmo_temperature`

Temperatura local da atmosfera em graus Kelvin. /300/

`mean_molecular_weight`

Massa molecular média da atmosfera, em moles (kg/kgmol). /16/

Parâmetros de saída:

Não há.

- **`void edm_set_drag_properties (vector3 un_sat_vel, double aerodynamic_pressure, double speed_ratio);`**

Esta função permite atribuir valores aos parâmetros ambientais utilizados no cálculo das forças e torques aerodinâmicos. Esta função pode modificar valores previamente atribuídos pela função `edm_set_environ_properties`.

Parâmetros de entrada:

`un_sat_vel`

Vetor do tipo `vector3`, contendo a direção (vetor unitário) da velocidade do satélite em relação à atmosfera, em coordenadas do sistema de eixos fixado ao satélite. /1, 0, 0/

`aerodynamic_pressure`

Pressão aerodinâmica no local, em N/m^2. É dada pelo produto entre a metade da densidade atmosférica, em kg/m^3, pelo quadrado da velocidade do satélite em relação à atmosfera, em m/s. /2.45 10^{-5}/

`speed_ratio`

Razão de velocidades *s*. É dada pela relação entre a velocidade do satélite e a velocidade mais provável das moléculas. /4/

Parâmetros de saída:

Não há.

- **`vector6 edm_get_drag_properties();`**

Esta função recupera os parâmetros ambientais correntes.

Parâmetros de entrada:

Não há.

Parâmetros de saída:

`edm_get_drag_properties`

Vetor do tipo `vector6` contendo, no primeiro vetor, a direção (vetor unitário) da velocidade do satélite em relação à atmosfera, em coordenadas do sistema de eixos fixado ao satélite, atribuída pela função `edm_set_environ_properties` ou `edm_set_drag_properties`, e, no segundo vetor, a pressão aerodinâmica no local, em N/m^2, a razão de velocidades, *s*, e um valor nulo na terceira componente do segundo vetor.

- **`vector3 edm_aerodynamic_plane (vector3 un_normal, double area);`**

A função `edm_aerodynamic_plane` calcula a força aerodinâmica agindo sobre um elemento de superfície do satélite, em função da sua área e da direção do vetor normal. A força é calculada apenas do lado da normal. As propriedades da superfície devem ser configuradas por meio da função `edm_set_drag_surface`. As propriedades do ambiente são configuráveis por intermédio das funções `edm_set_environ_properties` ou `edm_set_drag_properties`. Assume-se que a linha de ação da força passe pelo baricentro ou centro de pressões da superfície plana.

Parâmetros de entrada:

`un_normal`

Vetor do tipo `vector3` com a direção do vetor normal unitário ao elemento de superfície, em coordenadas do sistema fixado ao satélite.

`area`

Área do elemento, em m^2.

Parâmetros de saída:

`edm_aerodynamic_plane`

Vetor do tipo `vector3` contendo a força no elemento em N, no sistema de coordenadas fixado ao satélite.

- **`vector6 edm_aerodynamic_plane_fb (vector3 un_normal, double length, double area);`**

A função `edm_aerodynamic_plane_fb` calcula a força aerodinâmica agindo sobre dois elementos de superfície, de áreas iguais e normais opostas. A força é calculada em ambos os lados da superfície, o que significa que esta superfície constitui uma área simétrica. O parâmetro `length` informa a distância que separa os dois lados da superfície. Este comprimento é utilizado no cálculo d o torque, cujo valor retorna no segundo vetor de saída. O torque é referido ao ponto mediano do comprimento `length` que une os dois baricentros dos elementos de superfície. As propriedades da superfície devem ser configuradas por meio da função `edm_set_drag_surface`. As propriedades do ambiente são configuráveis por intermédio das funções `edm_set_environ_properties` ou `edm_set_drag_properties`.

<u>Parâmetros de entrada:</u>

`un_normal`

Vetor do tipo `vector3` com a direção do vetor normal unitário ao elemento de superfície, em coordenadas do sistema fixado ao satélite.

`length`

Distância entre as duas superfícies, em m.

`area`

Área do elemento, em m^2.

<u>Parâmetros de saída:</u>

`edm_aerodynamic_plane_fb`

Vetor do tipo `vector6` contendo a força no elemento em N, no sistema de coordenadas fixado ao satélite, e o torque em Nm, referido ao ponto mediano da distância que separa os elementos.

- **`vector6 edm_aerodynamic_triangle (vector3 vertex_1, vector3 vertex_2, vector3 vertex_3);`**

A função `edm_aerodynamic_triangle` calcula a força e o torque aerodinâmico agindo sobre um elemento de área dado pelos vértices de um triângulo. A normal externa ao elemento será calculada pelo produto vetorial $(V_2 - V_1) \times (V_3 - V_1)$ (regra da mão direita), na qual V_i é o vetor de coordenadas do vértice i ($i = 1, 2, 3$). A face utilizada no cálculo irá depender do valor atribuído pela função `edm_set_surface_side`. Por "default" o cálculo será efetuado apenas na face externa. A linha de ação da força passa pelo baricentro do triângulo. As propriedades da superfície devem ser configuradas por meio da função `edm_set_drag_surface`. As propriedades do ambiente são configuráveis por intermédio das funções `edm_set_environ_properties` ou `edm_set_drag_properties`.

Parâmetros de entrada:

`vertex_1`

Vetor do tipo `vector3` com a posição do vértice 1 em coordenadas do sistema fixado ao satélite, em metros.

`vertex_2`

Vetor do tipo `vector3` com a posição do vértice 2 em coordenadas do sistema fixado ao satélite, em metros.

`vertex_3`

Vetor do tipo `vector3` com a posição do vértice 3 em coordenadas do sistema fixado ao satélite, em metros.

Parâmetros de saída:

`edm_aerodynamic_triangle`

Vetor do tipo `vector6` contendo, no primeiro vetor, a força no elemento em N, e o torque no elemento, em Nm.

- **`vector6 edm_aerodynamic_quad (vector3 vertex_1, vector3 vertex_2, vector3 vertex_3, vector3 vertex_4);`**

A função `edm_aerodynamic_quad` calcula a força e o torque aerodinâmico agindo sobre um quadrilátero definido pelos seus vértices. O quadrilátero será dividido internamente em dois triângulos, formados pelos vértices 1, 2, 3 e 1, 3, 4. Deve-se assegurar que o quadrilátero seja planar. A face utilizada no cálculo irá depender do valor atribuído pela função `edm_set_surface_side`. Por "default" o cálculo será efetuado apenas na face externa. A linha de ação da força passa pelo baricentro do quadrilátero. As propriedades da superfície devem ser configuradas por meio da função `edm_set_drag_surface`. As propriedades do ambiente são configuráveis por intermédio das funções `edm_set_environ_properties` ou `edm_set_drag_properties`.

Parâmetros de entrada:

`vertex_1`

Vetor do tipo `vector3` com a posição do vértice 1 em coordenadas do sistema fixado ao satélite, em metros.

`vertex_2`

Vetor do tipo `vector3` com a posição do vértice 2 em coordenadas do sistema fixado ao satélite, em metros.

`vertex_3`

Vetor do tipo `vector3` com a posição do vértice 3 em coordenadas do sistema fixado ao satélite, em metros.

`vertex_4`

Vetor do tipo `vector3` com a posição do vértice 4 em coordenadas do sistema fixado ao satélite, em metros.

Parâmetros de saída:

`edm_aerodynamic_triangle`

Vetor do tipo `vector6` contendo, no primeiro vetor, a força no elemento em N, e o torque no elemento, em Nm.

- **`vector6 edm_aerodynamic_mesh (mesh s_mesh, vertex s_vert, material s_mat);`**

Esta função integra a força e torque aerodinâmicos agindo num satélite descrito por uma tabela de vértices, tabela de polígonos e por uma tabela de materiais. Estas tabelas são descritas adiante (veja-se as funções para a descrição da geometria).

Parâmetros de entrada:

`s_mesh`

Estrutura do tipo `mesh`, que armazena a tabela de triângulos do satélite.

`s_vert`

Estrutura do tipo `vertex`, que armazena a tabela de vértices do satélite.

`s_mat`

Estrutura do tipo `material`, que armazena a tabela de propriedades das superfícies do satélite.

Parâmetros de saída:

`edm_aerodynamic_mesh`

Vetor do tipo `vector6` contendo, no primeiro vetor, a força aerodinâmica integrada no satélite em N, e o torque aerodinâmico integrado, no segundo vetor, em Nm.

- **`vector3 edm_aerodynamic_sphere (double radius);`**

Esta função obtém a força aerodinâmica agindo numa esfera cujas propriedades da superfície devem ser configuradas por meio da função `edm_set_drag_surface`. As propriedades do ambiente são configuráveis por intermédio das funções `edm_set_environ_properties` ou `edm_set_drag_properties`. Assume-se que toda a superfície da esfera possui as mesmas propriedades (coeficientes e temperatura). A integral é realizada analiticamente, portanto é mais rápida e precisa do que uma subdivisão da esfera em elementos planos. Em virtude da simetria da esfera, a força resultante terá direção contrária à direção do vetor velocidade e o torque será nulo.

Parâmetros de entrada:

`radius`

Raio da esfera, em m.

Parâmetros de saída:

`edm_aerodynamic_sphere`

Vetor do tipo `vector3` contendo a força na esfera em N, no sistema de coordenadas fixado ao satélite.

- **`vector6 edm_aerodynamic_sphere_sector (double radius, vector3 ang_meridian, vector3 ang_paralel);`**

Esta função obtém a força aerodinâmica agindo num setor esférico cujas propriedades da superfície devem ser configuradas por meio da função `edm_set_drag_surface`. As propriedades do ambiente são configuráveis por intermédio das funções `edm_set_environ_properties` ou `edm_set_drag_properties`. Esta função subdivide o setor esférico em elementos pequenos, e efetua uma integração numérica (somatória) na força e torque que agem sobre os elementos.

Parâmetros de entrada:

`radius`

Raio da esfera, em m.

`ang_meridian`

Vetor do tipo `vector3` contendo, respectivamente, a longitude em graus do meridiano inicial do setor esférico, a longitude em graus do meridiano final, e o número de divisões para definição dos elementos de área entre os meridianos limites. Não há limites para estes parâmetros, mas recomenda-se que estejam compreendidos entre –180 e 360°. Além disso deve-se assegurar que a longitude inicial seja menor que a longitude final.

`ang_paralel`

Vetor do tipo `vector3` contendo, respectivamente, a latitude em graus do paralelo inicial do setor esférico, a latitude em graus do paralelo final, e o número de divisões para definição dos elementos de área entre os paralelos limites. Os paralelos devem estar compreendidos entre –90 e 90°, e a latitude inicial deve ser menor do que a latitude final.

Parâmetros de saída:

`edm_aerodynamic_sphere_sector`

Vetor do tipo `vector3` contendo a força no setor esférico em N, no sistema de coordenadas fixado ao satélite.

- **`vector6 edm_aerodynamic_cylinder (double radius, vector3 center_bottom, vector3 center_top);`**

Esta função obtém a força e o torque aerodinâmicos que agem num cilindro de raio `radius`. Os centros das faces inferior e superior do cilindro devem ser fornecidas em relação ao sistema do satélite, e com isso a altura do cilindro pode ser facilmente calculada. As forças nas faces superior e inferior do cilindro não são calculadas, apenas aquelas que agem sobre a superfície cilíndrica. Utiliza-se uma integral analítica da força e do torque, e, portanto, este cálculo é mais rápido e preciso do que uma integral numérica. As propriedades da superfície devem ser configuradas por meio da função `edm_set_drag_surface`. As propriedades do ambiente são configuráveis por intermédio das funções `edm_set_environ_properties` ou `edm_set_drag_properties`. Assume-se que toda a superfície do cilindro possui as mesmas propriedades (coeficientes e temperatura).

Parâmetros de entrada:

`radius`

Raio da esfera, em m.

`center_bottom`

Posição do centro da face inferior do cilindro, em metros, referida ao sistema do satélite.

`center_top`

Posição do centro da face superior do cilindro, em metros, referida ao sistema do satélite.

Parâmetros de saída:

`edm_aerodynamic_cylinder`

Vetor do tipo `vector6` contendo a força em N, e o torque em Nm, integrado no cilindro, no sistema de coordenadas fixado ao satélite.

- **`vector6 edm_aerodynamic_cone (double radius_bottom, vector3 center_bottom, double radius_top, vector3 center_top, int n_div)`**

A função `edm_solar_prsr_cone` obtém a força e o torque devido à pressão aerodinâmica que age num cone ou num cone truncado. As coordenadas dos centros das faces inferior e superior do cone devem ser fornecidas no sistema do satélite, e com isso a altura do cone pode ser facilmente calculada. As forças nas faces opostas do cone não são calculadas, apenas aquelas que agem sobre a superfície cônica. O cálculo é realizado por meio da divisão da superfície cônica em elementos planos, e, portanto, é necessário fornecer o número de divisões que constituem os elementos de área trapezoidais. As propriedades da superfície devem ser configuradas por meio da função `edm_set_set_drag_surface`. As propriedades do ambiente são configuráveis por intermédio das funções `edm_set_environ_properties` ou `edm_set_drag_properties`. Assume-se que toda a superfície externa do cone possua as mesmas propriedades (coeficientes e temperatura).

Parâmetros de entrada:

`radius_bottom`

Raio da base do cone, em m.

`center_bottom`

Posição do centro da base do cone, em metros, referida ao sistema do satélite.

`radius_top`

Raio da face do topo do cone, em m. Se `radius_top` for nulo, então a geometria resultante será um cone.

`center_top`

Posição do centro da face superior do cone, em metros, referida ao sistema do satélite.

Parâmetros de saída:

`edm_aerodynamic_cone`

Vetor do tipo `vector6` contendo a força aerodinâmica em N, e o torque em Nm, integrado no cone, no sistema de coordenadas fixado ao satélite.

- **`vector6 edm_aerodynamic_box (vector3 bottom_corner, vector3 top_corner);`**

Esta função obtém a força e o torque aerodinâmicos que agem num paralelepípedo definido pelas coordenadas de dois cantos opostos. As faces são assumidas como paralelas aos eixos cartesianos. As propriedades da superfície (coeficientes e temperatura) devem ser configuradas por meio da função `edm_set_drag_surface`, e são constantes para as 6 faces. As propriedades do ambiente são configuráveis por intermédio das funções `edm_set_environ_properties` ou `edm_set_drag_properties`.

Parâmetros de entrada:

`bottom_corner`

Posição de um dos vértices do paralelepípedo, nas coordenadas *x*, *y* e *z*, em metros, referida ao sistema do satélite.

`top_corner`

Posição dos vértices oposto ao vértice `bottom_corner`, nas coordenadas *x*, *y* e *z*, em metros, referida ao sistema do satélite.

Parâmetros de saída:

`edm_aerodynamic_box`

Vetor do tipo `vector6` contendo a força em N, e o torque em Nm, integrado no paralelepípedo, no sistema de coordenadas fixado ao satélite.

4 FUNÇÕES PARA CÁLCULO DE PRESSÃO DE RADIAÇÃO

Nesta seção são descritas funções destinadas ao fornecimento de parâmetros ambientais, características superficiais, descrição da geometria e cálculo das forças e torques de pressão de radiação solar.

- **`void edm_set_solar_rad_surface (double specular_coef, double diffuse_coef, double emmitance, double temperature);`**

  Esta função irá atribuir as características reflexivas da superfície, a serem utilizadas no cálculo subseqüente das forças e torques de pressão de radiação que agem nos elementos de área. Esta função deve ser empregada quando somente as forças de pressão de radiação forem calculadas.

  Parâmetros de entrada:

  `specular_coef`

  Coeficiente de reflexão especular da superfície, *e*, isto é, a relação média entre a radiação incidente e aquela que é refletida na forma especular. /0.5/

  `diffuse_coef`

  Coeficiente de reflexão difusa da superfície, δ, isto é, a relação média entre a radiação incidente e aquela que é refletida na forma difusa. /0.5/

  `emmitance`

  Emitância da superfície, ε, isto é, a relação entre a radiação térmica emitida pela superfície e aquela emitida por um corpo negro à mesma temperatura. /1.0/

  `temperature`

  Temperatura absoluta da superfície, em graus Kelvin, a ser empregada nos cálculos. /300.0/

  Parâmetros de saída:

  Não há.

- **`vector3 edm_get_solar_rad_surface ();`**

  Esta função permite recuperar o valor corrente das características reflexivas da superfície a serem utilizadas no cômputo das forças e torques de pressão de radiação.

  Parâmetros de entrada:

  Não há.

  Parâmetros de saída:

  `edm_get_solar_rad_surface`

Vetor do tipo `vector3` contendo respectivamente o coeficiente de reflexão especular da superfície, *e*, o coeficiente de reflexão difusa da superfície, δ, e a emitância da superfície, ε.

- **`double edm_get_surface_temperature ();`**

Esta função permite recuperar o valor corrente da temperatura da superfície a ser empregada no cálculo das forças e torques de pressão de radiação.

Parâmetros de entrada:

Não há.

Parâmetros de saída:

`edm_get_surface_temperature`

Variável do tipo `double` com a temperatura absoluta da superfície, em graus Kelvin.

- **`void edm_set_solar_rad_properties (vector3 sun_vector);`**

Esta função irá atribuir as características ambientais para cálculo das forças e torques de pressão de radiação que agem nos elementos de área. É uma alternativa à função `edm_set_solar_rad_pressure`. Ambas definem os mesmos parâmetros e não devem ser utilizadas concomitantemente.

Parâmetros de entrada:

`sun_vector`

Vetor do tipo `vector3` contendo a posição do centro do Sol no sistema de coordenadas fixado ao satélite, em metros.

Parâmetros de saída:

Não há.

- **`void edm_set_solar_rad_pressure (vector3 un_sun_vec, double rad_prsr);`**

Esta função irá atribuir as características ambientais para cálculo das forças e torques de pressão de radiação que agem nos elementos de área. É uma alternativa à função `edm_set_solar_rad_properties`. Ambas definem os mesmos parâmetros e não devem ser utilizadas concomitantemente.

Parâmetros de entrada:

`un_sun_vec`

Vetor do tipo `vector3` contendo a direção contrária à incidência da luz solar, ou seja, a direção do Sol, no sistema de coordenadas fixado ao satélite.

`rad_prsr`

Variável do tipo `double` contendo a pressão de radiação. Normalmente esta pressão é obtida pela relação entre a potência solar corrigida para a distância Terra-Sol (por volta de 1353 W/m^2) e a velocidade da luz no vácuo (cerca de 300000 km/s).

Parâmetros de saída:

Não há.

- **`vector6 edm_get_solar_rad_properties ();`**

Esta função permite recuperar as propriedades atuais da pressão de radiação.

Parâmetros de entrada:

Não há.

Parâmetros de saída:

`edm_get_solar_rad_properties`

Vetor do tipo `vector6` contendo, no primeiro vetor (`._1`) o versor de direção do Sol, no sistema de referência do satélite, e, nas componentes do segundo vetor, respectivamente a distância Terra-Sol em unidades astronômicas, a pressão de radiação solar local, em N/m^2, e um valor nulo.

- **`vector3 edm_solar_prsr_plane (vector3 un_normal, double area);`**

A função `edm_solar_prsr_plane` calcula a força de pressão de radiação solar agindo sobre um elemento de superfície do satélite, em função da sua área e da direção do vetor normal. A força é calculada apenas do lado da normal. As propriedades da superfície devem ser configuradas por meio da função `edm_set_solar_rad_surface`. As propriedades do ambiente são configuráveis por intermédio da função `edm_set_solar_rad_properties`. Assume-se que a linha de ação da força passe pelo baricentro ou centro de pressões da superfície plana.

Parâmetros de entrada:

`un_normal`

Vetor do tipo `vector3` com a direção do vetor normal unitário ao elemento de superfície, em coordenadas do sistema fixado ao satélite.

`area`

Área do elemento, em m^2.

Parâmetros de saída:

`edm_solar_prsr_plane`

Vetor do tipo `vector3` contendo a força de pressão de radiação no elemento, em N, no sistema de coordenadas fixado ao satélite.

- **`vector6 edm_solar_prsr_triangle (vector3 vertex_1, vector3 vertex_2, vector3 vertex_3);`**

A função `edm_solar_prsr_triangle` calcula a força e o torque de pressão de radiação solar agindo sobre um elemento de área dado pelos vértices de um triângulo. A normal externa ao elemento será calculada pelo produto vetorial $(V_2 - V_1) \times (V_3 - V_1)$ (regra da mão direita), na qual V_i é o vetor de coordenadas do vértice i (i = 1, 2, 3). A face utilizada no cálculo irá depender do valor atribuído pela função `edm_set_surface_side`. Por "default" o cálculo será efetuado apenas na face externa. As propriedades da superfície devem ser configuradas por meio da função `edm_set_solar_rad_surface`. As propriedades do ambiente são configuráveis por intermédio da função `edm_set_solar_rad_properties`. A linha de ação da força passa pelo baricentro do triângulo.

<u>Parâmetros de entrada:</u>

`vertex_1`

Vetor do tipo `vector3` com a posição do vértice 1 em coordenadas do sistema fixado ao satélite, em metros.

`vertex_2`

Vetor do tipo `vector3` com a posição do vértice 2 em coordenadas do sistema fixado ao satélite, em metros.

`vertex_3`

Vetor do tipo `vector3` com a posição do vértice 3 em coordenadas do sistema fixado ao satélite, em metros.

<u>Parâmetros de saída:</u>

`edm_solar_prsr_triangle`

Vetor do tipo `vector6` contendo, no primeiro vetor, a força no elemento em N, e o torque no elemento, em Nm.

- **`vector6 edm_solar_prsr_quad (vector3 vertex_1, vector3 vertex_2, vector3 vertex_3, vector3 vertex_4);`**

A função `edm_solar_prsr_quad` calcula a força e o torque de pressão de radiação solar agindo sobre um quadrilátero definido pelos seus vértices. O quadrilátero será dividido internamente em dois triângulos, formados pelos vértices 1, 2, 3 e 1, 3, 4. Deve-se assegurar que o quadrilátero seja planar. A linha de ação da força passa pelo baricentro do quadrilátero. A face utilizada no cálculo irá depender do valor atribuído pela função `edm_set_surface_side`. Por "default" o cálculo será efetuado apenas na face externa. As propriedades da superfície devem ser configuradas por meio da função `edm_set_solar_rad_surface`. As propriedades do ambiente são configuráveis por intermédio da função `edm_set_solar_rad_properties`.

Parâmetros de entrada:

`vertex_1`

Vetor do tipo `vector3` com a posição do vértice 1 em coordenadas do sistema fixado ao satélite, em metros.

`vertex_2`

Vetor do tipo `vector3` com a posição do vértice 2 em coordenadas do sistema fixado ao satélite, em metros.

`vertex_3`

Vetor do tipo `vector3` com a posição do vértice 3 em coordenadas do sistema fixado ao satélite, em metros.

`vertex_4`

Vetor do tipo `vector3` com a posição do vértice 4 em coordenadas do sistema fixado ao satélite, em metros.

Parâmetros de saída:

`edm_solar_prsr_quad`

Vetor do tipo `vector6` contendo, no primeiro vetor, a força no elemento em N, e o torque no elemento, em Nm.

- **`vector6 edm_solar_prsr_mesh (mesh s_mesh, vertex s_vert, material s_mat);`**

Esta função integra a força e torque de pressão de radiação solar agindo num satélite descrito por uma tabela de vértices, tabela de polígonos e por uma tabela de materiais. Estas tabelas são descritas adiante (veja-se as funções para a descrição da geometria).

Parâmetros de entrada:

`s_mesh`

Estrutura do tipo `mesh`, que armazena a tabela de triângulos do satélite.

`s_vert`

Estrutura do tipo `vertex`, que armazena a tabela de vértices do satélite.

`s_mat`

Estrutura do tipo `material`, que armazena a tabela de propriedades das superfícies do satélite.

Parâmetros de saída:

`edm_aerodynamic_mesh`

Vetor do tipo `vector6` contendo, no primeiro vetor, a força aerodinâmica integrada no satélite em N, e o torque aerodinâmico integrado, no segundo vetor, em Nm.

- **`vector3 edm_solar_prsr_sphere (double radius);`**

Esta função obtém a força devido à pressão de radiação solar agindo numa esfera cujas propriedades da superfície devem ser configuradas por meio da função `edm_set_solar_rad_surface`. A intensidade da radiação e a direção do Sol são estabelecidas pelas funções `edm_set_solar_rad_properties` ou `edm_set_solar_rad_pressure`. Assume-se que toda a superfície da esfera possui as mesmas propriedades (coeficientes e temperatura). A integral é realizada analiticamente, portanto é mais rápida e precisa do que uma subdivisão da esfera em elementos planos. Em virtude da simetria da esfera, a força resultante terá direção contrária à direção do Sol e o torque será nulo.

Parâmetros de entrada:

`radius`

Raio da esfera, em m.

Parâmetros de saída:

`edm_solar_prsr_sphere`

Vetor do tipo `vector3` contendo a força na esfera em N, no sistema de coordenadas fixado ao satélite.

- **`vector6 edm_solar_prsr_sphere_sector (double radius, vector3 ang_meridian, vector3 ang_paralel);`**

Esta função obtém a força devido à pressão de radiação solar que age num setor esférico cujas propriedades da superfície devem ser configuradas por meio da função `edm_set_solar_rad_surface`. A intensidade da radiação e a direção do Sol são estabelecidas pelas funções `edm_set_solar_rad_properties` ou `edm_set_solar_rad_pressure`. Esta função subdivide o setor esférico em elementos pequenos, e efetua uma integração numérica (somatória) na força e torque que agem sobre os elementos.

Parâmetros de entrada:

`radius`

Raio da esfera, em m.

`ang_meridian`

Vetor do tipo `vector3` contendo, respectivamente, a longitude em graus do meridiano inicial do setor esférico, a longitude em graus do meridiano final, e o número de divisões para definição dos elementos de área entre os meridianos limites. Não há limites para estes parâmetros, mas recomenda-se que estejam

compreendidos entre –180 e 360°. Além disso deve-se assegurar que a longitude inicial seja menor que a longitude final.

`ang_paralel`

Vetor do tipo `vector3` contendo, respectivamente, a latitude em graus do paralelo inicial do setor esférico, a latitude em graus do paralelo final, e o número de divisões para definição dos elementos de área entre os paralelos limites. Os paralelos devem estar compreendidos entre –90 e 90°, e a latitude inicial deve ser menor do que a latitude final.

Parâmetros de saída:

`edm_solar_prsr_sphere_sector`

Vetor do tipo `vector3` contendo a força no setor esférico em N, no sistema de coordenadas fixado ao satélite.

- **`vector6 edm_solar_prsr_cylinder (double radius, vector3 center_bottom, vector3 center_top);`**

Esta função obtém a força e o torque devido à pressão de radiação solar que agem num cilindro de raio `radius`. Os centros das faces inferior e superior do cilindro devem ser fornecidos em relação ao sistema do satélite, e com isso a altura do cilindro pode ser facilmente calculada. As forças nas faces superior e inferior do cilindro não são calculadas, apenas aquelas que agem sobre a superfície cilíndrica. Utiliza-se uma integral analítica da força e do torque, e, portanto, este cálculo é mais rápido e preciso do que uma integral numérica. As propriedades da superfície devem ser configuradas por meio da função `edm_set_solar_rad_surface`. A intensidade da radiação e a direção do Sol são estabelecidas pelas funções `edm_set_solar_rad_properties` ou `edm_set_solar_rad_pressure`. Assume-se que toda a superfície do cilindro possui as mesmas propriedades (coeficientes e temperatura).

Parâmetros de entrada:

`radius`

Raio do cilindro, em m.

`center_bottom`

Posição do centro da face inferior do cilindro, em metros, referida ao sistema do satélite.

`center_top`

Posição do centro da face superior do cilindro, em metros, referida ao sistema do satélite.

Parâmetros de saída:

`edm_solar_prsr_cylinder`

Vetor do tipo `vector6` contendo a força em N, e o torque em Nm, integrado no cilindro, no sistema de coordenadas fixado ao satélite.

- **`vector6 edm_solar_prsr_cone (double radius_bottom, vector3 center_bottom, double radius_top, vector3 center_top)`**

A função `edm_solar_prsr_cone` obtém a força e o torque devido à pressão de radiação solar que agem num cone ou num cone truncado. As coordenadas dos centros das faces inferior e superior do cone devem ser fornecidas no sistema do satélite, e com isso a altura do cone pode ser facilmente calculada. As forças nas faces opostas do cone não são calculadas, apenas aquelas que agem sobre a superfície cônica. Utiliza-se uma integral analítica da força e do torque, e, portanto, este cálculo é mais rápido e preciso do que uma integração numérica. As propriedades da superfície devem ser configuradas por meio da função `edm_set_solar_rad_surface`. A intensidade da radiação e a direção do Sol são estabelecidas pelas funções `edm_set_solar_rad_properties` ou `edm_set_solar_rad_pressure`. Assume-se que toda a superfície externa do cone possui as mesmas propriedades (coeficientes e temperatura).

<u>Parâmetros de entrada:</u>

`radius_bottom`

Raio da base do cone, em m.

`center_bottom`

Posição do centro da base do cone, em metros, referida ao sistema do satélite.

`radius_top`

Raio da face do topo do cone, em m. Se `radius_top` for nulo, então a geometria resultante será um cone.

`center_top`

Posição do centro da face superior do cone, em metros, referida ao sistema do satélite.

<u>Parâmetros de saída:</u>

`edm_solar_prsr_cone`

Vetor do tipo `vector6` contendo a força em N, e o torque em Nm, integrado no cone, no sistema de coordenadas fixado ao satélite.

- **`vector6 edm_solar_prsr_box (vector3 bottom_corner, vector3 top_corner);`**

Esta função obtém a força e o torque devidos à pressão de radiação solar que agem num paralelepípedo definido pelas coordenadas de dois cantos opostos. As faces são assumidas como paralelas aos eixos cartesianos. As propriedades da superfície (coeficientes e temperatura) devem ser configuradas por meio da função `edm_set_solar_rad_surface`, e são constantes para as 6 faces. A intensidade da radiação e a direção do Sol são

estabelecidas pelas funções `edm_set_solar_rad_properties` ou `edm_set_solar_rad_pressure`.

Parâmetros de entrada:

`bottom_corner`

Posição de um dos vértices do paralelepípedo, nas coordenadas *x*, *y* e *z*, em metros, referida ao sistema do satélite.

`top_corner`

Posição dos vértices oposto ao vértice `bottom_corner`, nas coordenadas *x*, *y* e *z*, em metros, referida ao sistema do satélite.

Parâmetros de saída:

`edm_aerodynamic_box`

Vetor do tipo `vector6` contendo a força em N, e o torque em Nm, integrado no paralelepípedo, no sistema de coordenadas fixado ao satélite.

5 ESTRUTURAS PARA DESCRIÇÃO DA GEOMETRIA DO SATÉLITE

A necessidade de construir funções que permitissem manipular, ler e armazenar informações a respeito da geometria e das características superficiais dos satélites fez com que fossem naturalmente criadas estruturas que armazenassem estes dados. Embora elas tenham sido criadas com a intenção de tornar o processo de descrição da geometria mais automatizado, ainda assim elas permitem que seus constituintes sejam individualmente acessados e modificados. As estruturas permitem armazenar a tabela de vértices, a tabela de triângulos e a tabela de características superficiais do satélite. A memória necessária para armazenar cada uma das três estruturas deve ser alocada durante a execução do programa. Foram implementadas funções específicas para alocar e liberar a memória de armazenagem da geometria.

Uma vez que o satélite pode possuir apêndices articulados, a tabela de vértices terá que ser ajustada de forma a prover este movimento, pois os vértices, do ponto de vista do sistema de coordenadas fixado ao corpo principal do satélite, giram ao redor de um eixo. As funções para cálculo da força e torque numa malha poligonal ("mesh") não efetuam esta transformação, cabendo, portanto, ao usuário efetuá-la, caso seja necessário. Para identificar a qual apêndice um determinado triângulo pertence e com isso efetuar a transformação, um vetor de índices denominado de componente do triângulo, `syst_pt[]`, foi incluído na tabela de triângulos. A identificação do componente, isto é, o valor numérico armazenado neste vetor deverá necessariamente coincidir com o valor atribuído ao apêndice pela função `set_body_k`, caso seja empregado o simulador de atitude **`attit_pro`**. Por exemplo, seja o apêndice 2 formado pelos vértices $\mathbf{V}_{15}$ a $\mathbf{V}_{23}$. Na tabela de vértices, o valor de `syst_pt[`*i*`]` deverá ser igual a 2, para os vértices de 15 a 23. Igualmente, a função `set_body_k` deverá definir as características de inércia, posição e orientação do apêndice 2 compatíveis com a geometria descrita pelos vértices. No Capítulo 12 é apresentado um exemplo de transformação da geometria para girar o painel do satélite CBERS.

Deve-se mencionar que o cálculo da força de arrasto e pressão de radiação não leva em conta eventuais reflexões duplas caso o satélite possua geometria côncava, nem tampouco verifica sombreamento de superfícies por outras. Por isso não é necessário efetuar translações na tabela de vértices, mas somente a devida rotação do apêndice.

As tabelas de vértices, triângulos e materiais possuem um vetor de identificadores, respectivamente `vtx_id[]`, `trg_id[]`, e `mat_id[]`. Estes identificadores, de fato, não são utilizados no cálculo das forças e torques, mas fazem associações entre as estruturas e servem para atribuir os valores dos membros `mat_pt[]` e `trg_tab[]` na estrutura que define os triângulos, quando a geometria for fornecida, por exemplo, por meio de arquivos (Veja-se o próximo capítulo).

Deve ser ressaltado que as propriedades da atmosfera, incluindo a temperatura, (função `edm_set_environ_properties`) devem ser previamente estabelecidos antes de se promover o cálculo das forças e torques aerodinâmicos, já que a estrutura de materiais não armazena a razão de temperaturas, mas sim a temperatura absoluta dos elementos.

As seguintes operações aritméticas foram estendidas e implementadas em estruturas do tipo `vertex`:

- A pré ou pós soma ou diferença de um vetor do tipo `vector3` com uma variável do tipo `vertex` promove uma translação correspondente de todos os vértices da tabela.
- O produto de uma matriz do tipo `matrix3` por uma variável do tipo `vertex` promove uma transformação (rotação) correspondente de todos os vértices da tabela.
- O pré ou pós-produto de uma variável do tipo `double` ou `int` por uma variável do tipo `vertex` promove uma variação de escala correspondente de todos os vértices da tabela.

A seguir fornece-se uma descrição das estruturas implementadas.

- **`vecint3`**

Vetor com 3 variáveis inteiras (`int`), nas componentes `._1`, `._2`, e `._3`.

- **`vertex`**

Tabela de vértices, com 3 membros:

`numb_vtx`

Variável inteira contendo o número de vértices da tabela.

`vtx_id[]`

Vetor de inteiros contendo a identificação de cada vértice.

`vtx_tab[]`

Vetor de variáveis do tipo `vector3` contendo as coordenadas dos vértices, em metros.

- **`mesh`**

Tabela de triângulos, com 6 membros:

`numb_trg`

Variável inteira contendo o número de triângulos da superfície do satélite.

`trg_id[]`

Vetor de inteiros contendo um número de identificação de cada triângulo.

`syst_pt[]`

Vetor do tipo `unsigned char` contendo a informação sobre a estrutura à qual pertence o triângulo: corpo principal do satélite (0) ou cada um dos eventuais apêndices articulados (seqüencialmente a partir de 1). Veja-se a Seção 3.6, Propriedades do Satélite, de Carrara (2007).

`side_fc[]`

Vetor do tipo `unsigned char` contendo a informação sobre a direção da normal (conforme código estabelecido pela função `edm_set_surface_side`).

`mat_pt[]`

Vetor do tipo inteiro contendo um apontador para o identificador de material da superfície.

`trg_tab[]`

Vetor de variáveis do tipo `vecint3` contendo apontadores para os índices dos 3 vértices que compõem o triângulo.

- **`material`**

Tabela com as características da superfície, com 8 membros:

`numb_mat`

Número de materiais presentes na tabela.

`mat_id[]`

Vetor de inteiros contendo um número de identificação de cada material.

`s_nor[]`

Vetor do tipo `double` contendo o coeficiente de troca de quantidade de movimento na direção normal (Veja-se `edm_set_drag_surface`).

`s_tan[]`

Vetor do tipo `double` contendo o coeficiente de troca de quantidade de movimento na direção tangencial (Veja-se `edm_set_drag_surface`).

`espec[]`

Vetor do tipo `double` contendo o coeficiente de reflexão especular da superfície (Veja-se `edm_set_solar_rad_surface`).

`diffuse[]`

Vetor do tipo `double` contendo o coeficiente de reflexão difusa da superfície (Veja-se `edm_set_solar_rad_surface`).

`emissiv[]`

Vetor do tipo `double` contendo a emitância ou emissividade da superfície (Veja-se `edm_set_solar_rad_surface`).

`sur_tp[]`

Vetor do tipo `double` contendo a temperatura absoluta da superfície em Kelvin (Veja-se `edm_set_solar_rad_surface`).

A criação das variáveis que irão armazenar estas estruturas é realizada por meio das instruções:

- `vertex sat_vert;`
- `material sat_mat;`
- `mesh sat_mesh;`

nas quais `sat_vert`, `sat_mat` e `sat_mesh` são os nomes das variáveis. Contudo, estas instruções apenas definem as estruturas, mas não aloca a memória necessária para armazenar os dados. O motivo disso é que, em certas situações, a informação sobre a quantidade de membros necessários nas estruturas está disponível apenas durante a execução do programa. Portanto, tão logo tenham sido definidas, será necessário alocar a memória para cada uma delas. Foram criadas três funções para alocar e três para liberar a memória após o uso. Caso sejam utilizadas as funções para leitura de arquivo contendo a geometria (veja-se os Capítulos 6 e 7), não será necessário alocar a memória. Porém a liberação deve ser efetuada ao término da execução. Alguns exemplos de uso das funções de leitura podem ser vistos no Capítulo 12.

- **`vertex alloc_vertex(int size);`**

  A função `alloc_vertex` aloca a quantidade necessária de memória para armazenar a estrutura do tipo `vertex` que armazena a tabela de vértices. O parâmetro `size` deverá ser igual ao número de vértices a serem armazenados. Veja-se também a função `dealloc_vertex`.

  Parâmetros de entrada:

  `size`

  Variável do tipo `int` que fornece o número máximo de vértices a serem armazenados na estrutura. Qualquer tentativa de se armazenar um número maior de vértices além daquele alocado poderá resultar em falha do programa.

  Parâmetros de saída:

  `alloc_vertex`

  Variável do tipo `vertex` contendo um ponteiro para a memória alocada pela função.

- **`void dealloc_vertex(vertex vert);`**

  A função `dealloc_vertex` libera para uso a memória previamente alocada para armazenar uma estrutura do tipo `vertex` que armazena a tabela de vértices. Veja-se também a função `alloc_vertex`.

  Parâmetros de entrada:

  `vert`

  Variável do tipo `vertex` utilizada para armazenar a estrutura de vértices, que tenha sido alocada pela função `alloc_vertex`.

  Parâmetros de saída:

  Não há

- **`material alloc_material(int size);`**

A função `alloc_material` aloca a quantidade necessária de memória para armazenar a estrutura do tipo `material` que armazena a tabela de propriedades da superfície. O parâmetro `size` deverá ser igual ao número de diferentes materiais presentes no satélite. Veja-se também a função `dealloc_material`.

Parâmetros de entrada:

`size`

Variável do tipo `int` que fornece o número máximo de materiais a serem armazenados na estrutura. Qualquer tentativa de se armazenar um número além daquele alocado poderá resultar em falha do programa.

Parâmetros de saída:

`alloc_material`

Variável do tipo `material` contendo um ponteiro para a memória alocada pela função.

- **`void dealloc_material(material mat);`**

A função `dealloc_material` libera para uso a memória previamente alocada para armazenar uma estrutura do tipo `material` que armazena as propriedades das superfícies. Veja-se também a função `alloc_material`.

Parâmetros de entrada:

`mat`

Variável do tipo `material` utilizada para armazenar a estrutura de materiais, que tenha sido alocada pela função `alloc_material`.

Parâmetros de saída:

Não há

- **`mesh alloc_mesh(int size);`**

A função `alloc_mesh` aloca a quantidade necessária de memória para armazenar a estrutura do tipo `mesh` que armazena a tabela de triângulos (polígonos). O parâmetro `size` deverá ser igual ao número de triângulos presentes no satélite. Veja-se também a função `dealloc_mesh`.

Parâmetros de entrada:

`size`

Variável do tipo `int` que fornece o número máximo de triângulos a serem armazenados na estrutura. Qualquer tentativa de se armazenar um número superior àquele alocado poderá resultar em falha do programa.

Parâmetros de saída:

`alloc_mesh`

Variável do tipo `mesh` contendo um ponteiro para a memória alocada pela função.

- **`void dealloc_mesh(mesh s_mesh);`**

A função `dealloc_mesh` libera para uso a memória previamente alocada para armazenar a tabela de triângulos. Veja-se também a função `alloc_mesh`.

Parâmetros de entrada:

`s_mesh`

Variável do tipo `mesh` utilizada para armazenar a tabela de triângulos, alocada pela função `alloc_mesh`.

Parâmetros de saída:

Não há

Para criar e alocar as estruturas necessárias para as tabelas, pode-se utilizar um código análogo ao mostrado na Figura 5.1.

```
vertex sat_vert;
material sat_mat;
mesh sat_mesh;

sat_vert = alloc_vert(12);    // alocação de espaço para 12 vértices
sat_mat = alloc_material(10); // alocação de espaço para 10 materiais
sat_mesh = alloc_mesh(20);    // alocação de espaço para 20 triângulos
...                           // utilização das estruturas
dealloc_vert(sat_vert);       // liberação dos vértices
dealloc_material(sat_mat);    // liberação dos materiais
dealloc_mesh(sat_mesh);       // liberação da malha de polígonos
```

Fig 5.1 – Exemplo de alocação das estruturas de armazenagem de geometria.

6 FORMATO DO ARQUIVO DE DESCRIÇÃO DA GEOMETRIA

Muito embora as funções descritas anteriormente, em conjunto com as estruturas já disponíveis para armazenar a geometria, permitam modelar um satélite a partir de um código de programa, há que se considerar que esta tarefa pode ser complexa principalmente se o satélite for composto por um número grande de polígonos. Um processo automatizado de geração de geometria poderia ser então servir para facilitar esta codificação. De fato, existem diversos programas disponíveis para geração de malha poligonal de objetos, tais como o AutoCad, 3DStudio, Rhinoceros, SolidWorks e Catia, só para ficar com apenas alguns deles. Uma rápida consulta ao sítio "3dlinks.com" permite avaliar a quantidade imensa de programas gráficos de modelagem. Merecem especial atenção, contudo, os programas que podem ser usados livremente, ou seja, de licença pública, como o Blender, Moray, K3D e diversos outros. Infelizmente, quase todos eles (e também os programas comerciais), armazenam a geometria em formatos próprios, por vezes em binário, o que torna o processo de conversão difícil, para não dizer impossível. Uma rara exceção é o Blender, que, além de editor, efetua conversão entre formatos diferentes, alguns deles em ASCII. Por motivos de compatibilidade com versões anteriores do programa de perturbação ambiental, adotou-se aqui o formato do Nastran, que é um programa desenvolvido inicialmente pela NASA, destinado a realizar análise de esforços e vibração por elementos finitos. O Nastran utiliza um arquivo em ASCII contendo a descrição da geometria num formato simples e de fácil compreensão. O arquivo descreve a geometria por meio de uma tabela de vértices e uma tabela de polígonos. Estas tabelas são identificadas por um comando de 8 bytes que inicia cada registro (linha) do arquivo. Descrever-se-á aqui apenas aqueles comandos que serão efetivamente utilizados no cálculo das forças e torques ambientais. Embora a geometria descrita pelo Nastran em geral contenha também elementos internos ao satélite, é óbvio que estes elementos devem ser removidos na geração do arquivo final, principalmente se a geometria for muito complexa. Felizmente, a maior parte dos satélites pode ser descrita eficientemente com um número reduzido de polígonos, que mal chega a 40 triângulos. Deve-se levar em conta, é claro, que pequenas estruturas como antenas, suportes, adaptadores, etc, não necessitam ser incorporados na geometria porque demandam grande processamento e não causam alteração significativa nos resultados. O formato adotado aqui utiliza o conceito de registros com comandos de identificação e constitui, portanto, uma extensão daquele usado pelo Nastran. Ao contrário deste, contudo, as informações contidas nos novos comandos são separadas por espaços em brancos, ao invés de posições fixas. Dois novos comandos foram necessários para a descrição da geometria: `MATERIAL` e `BODYAP`. Todos os comandos são descritos a seguir.

- **`GRID`**

  A tabela de vértices é reconhecida pelo comando `GRID`, nas colunas de 1 a 4 (5 a 8 devem ser espaços brancos). Deverá haver tantos comandos `GRID` quanto for o número de vértices da geometria. As informações contidas neste comando são (veja-se a Figura 6.1):

  - Colunas 1 a 8: comando (`GRID`), ajustado à esquerda
  - Colunas 9 a 16: identificador do vértice. Número inteiro, não necessariamente seqüencial, unívoco do vértice, isto é, dois ou mais vértices não podem ter o mesmo identificador.
  - Colunas 17 a 24: número de identificação de comando. Este comando possui identificador 0.

- Colunas 25 a 32, 33 a 40, 41 a 48: coordenadas do vértice, em quaisquer unidades. A unidade deverá ser ajustada para métrica pelo usuário após o processo de leitura do arquivo por meio de uma variação de escala (veja-se o Capítulo5).
- Colunas 49 a 56: não utilizadas. Mantidas apenas para ser compatível com Nastran.

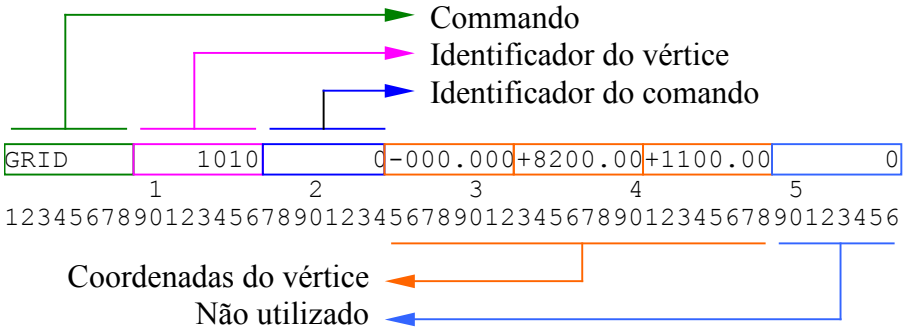

Fig 6.1 – Comando **GRID** do arquivo de descrição da geometria

- **CTRIA3**

A tabela de polígonos é identificada pelos comandos CTRIA3 e CQUAD4, nas colunas de 1 a 6 (7 e 8 devem ser espaços brancos). Deverá haver tantos comandos quanto for o número de polígonos presentes na geometria. CTRIA3 descreve um triângulo enquanto que CQUAD4 define um quadrilátero plano. As informações contidas no comando CTRIA3 são (veja-se a Figura 6.2):

- Colunas 1 a 8: comando (CTRIA3), ajustado à esquerda
- Colunas 9 a 16: identificador do polígono. Deve ser um número inteiro, não necessariamente seqüencial, unívoco de qualquer polígono, ou seja, dois ou mais polígonos não devem compartilhar o mesmo identificador, sejam eles triângulos ou quadriláteros.
- Colunas 17 a 24: número de identificação do comando. Este comando possui identificador igual a 1.
- Colunas 25 a 32, 33 a 40, 41 a 48: referência para os identificadores dos vértices (vistos nas colunas 9 a 16 do comando GRID). A ordem dos vértices deve seguir a regra da mão direita para a definição da normal ao triângulo.
- Colunas 49 a 56: não utilizadas.

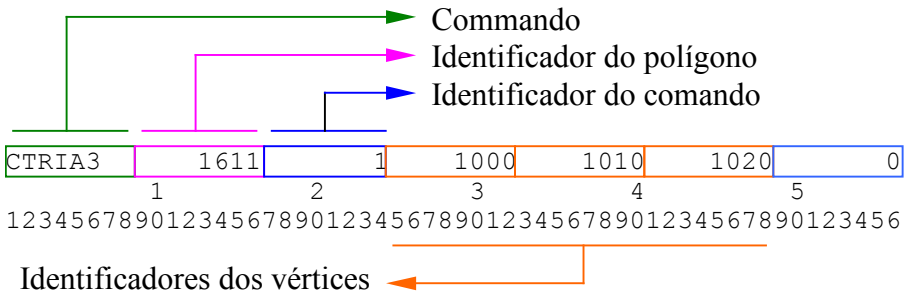

Fig 6.2 – Comando **CTRIA3** do arquivo de descrição da geometria.

- **CQUAD4**

O comando CQUAD4 define um quadrilátero plano. O comando é inserido nas colunas de 1 a 6 (7 e 8 devem ser espaços brancos). Deverá haver tantos comandos CQUAD4 quanto for o número de polígonos presentes na geometria. A estrutura que irá armazenar as informações dos polígonos (veja-se a estrutura mesh no Capítulo 5) permite apenas a criação de triângulos. Portanto, o comando CQUAD4 será dividido durante a leitura em dois triângulos. A ordem de fornecimento dos vértices do quadrilátero deve ser coerente com esta divisão. As informações contidas neste comando são (veja-se a Figura 6.3):

- Colunas 1 a 8: comando (CQUAD4), ajustado à esquerda
- Colunas 9 a 16: identificador do polígono. Deve ser um número inteiro, não necessariamente seqüencial, unívoco do polígono, e também diferente de qualquer identificador utilizado no comando CTRIA3 ou seja, dois ou mais polígonos não devem compartilhar o mesmo identificador.
- Colunas 17 a 24: número de identificação de comando. Este comando possui identificador igual a 1.
- Colunas 25 a 32, 33 a 40, 41 a 48 e 49 a 56: referência aos identificadores dos vértices (encontrados nas colunas 9 a 16 do comando GRID). A ordem dos identificadores deve seguir a regra da mão direita para a definição da normal ao polígono. O quadrilátero deverá ser válido, isto é, a seqüência dos vértices deverá seguir o caminho das arestas.

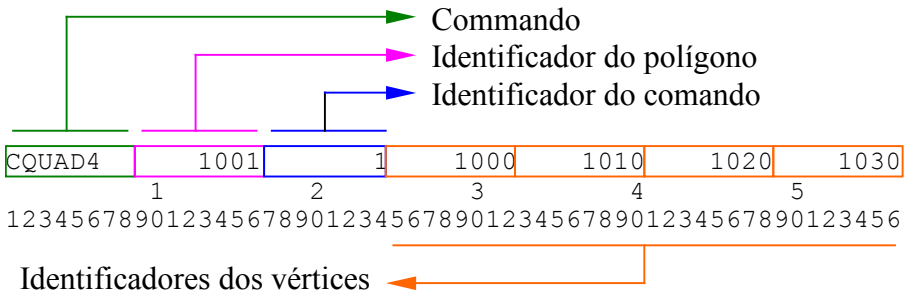

Fig 6.3 – Comando **CQUAD4** do arquivo de descrição da geometria.

- **ENDDATA**

O comando ENDDATA define o fim da descrição da geometria. Quaisquer outros registros ou comandos serão ignorados durante a leitura. É importante assinalar que este comando deverá ser seguido por espaços em branco pelo menos até a coluna 8, caso contrário poderá ocorrer erro durante a leitura.

- **MATERIAL**

O comando MATERIAL (Figura 6.4) permite atribuir as características das superfícies presentes no satélite. Deverá haver tantos comandos deste tipo quanto for o número de diferentes materiais com características distintas e conhecidas presentes na superfície do satélite. As informações contidas neste comando devem ser separadas por um ou mais espaços em branco e são:

- Colunas 1 a 8: comando (MATERIAL).
- Identificador do material, a partir da coluna 10. Deve ser um número inteiro, não necessariamente seqüencial, unívoco do material.

- Número de identificação de comando. `MATERIAL` possui identificador igual a 3.
- Coeficiente de troca de quantidade de movimento na direção normal (σ_n), compreendido entre 0 e 1.
- Coeficiente de troca de quantidade de movimento na direção tangencial (σ_t), compreendido entre 0 e 1.
- Coeficiente de reflexão especular (e)– parcela da radiação incidente que é refletida de forma especular – compreendido entre 0 e 1.
- Coeficiente de reflexão difusa (d) – parcela da radiação incidente que é espalhada difusamente – compreendido entre 0 e 1.
- Emitância ou emissividade da superfície (ε), de acordo com a lei de Stefan-Boltzmann – compreendido entre 0 e 1.
- Temperatura absoluta do elemento em Kelvin.

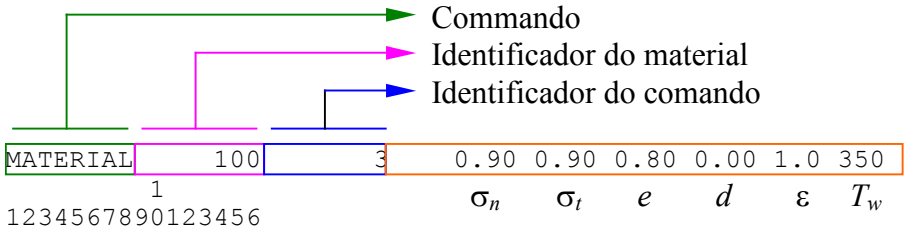

Fig 6.4 – Comando **`MATERIAL`** do arquivo de descrição da geometria.

A soma dos coeficientes de reflexão não deverá ultrapassar a unidade, por motivos de equilíbrio de energia. Na verdade a parcela transmitida, adicionada da parcela absorvida, da refletida na forma especular e na forma difusa, igualam a totalidade da radiação incidente.

- **`BODYAP`**

O comando `BODYAP` permite definir os polígonos a serem utilizados no cálculo. Nota-se que a tabela de polígonos definida pelos comandos `CTRIA3` e `CQUAD4` deverá conter, no mínimo, todos os polígonos a serem utilizados no cálculo. Porém os polígonos efetivamente considerados serão aqueles definidos pelo comando `BODYAP` – os excedentes serão descartados. A duplicidade de comandos que definem o polígono se deve ao fato de ser necessário associar um material a cada polígono. A associação é feita pelo comando `BODYAP`, e, para diferenciar o polígono não associado daquele com material, este último será denominado de face. A cada face será associado um único polígono. O comando `BODYAP` atribui as seguintes características (Figura 6.5):

- Colunas 1 a 8: comando (`BODYAP`), ajustado à esquerda.
- Identificador da face, a partir da coluna 9. Deve ser um número inteiro, não necessariamente seqüencial, unívoco da face.
- Número de identificação de comando. `BODYAP` é definido como igual a 2.
- Número de identificação do polígono associado à face. O número do polígono é definido pelas colunas 9 a 16 dos comandos `CTRIA3` e `CQUAD4`.
- Apontador para o sistema de coordenadas. Define se a face pertence ao corpo principal do satélite ou a qual dos apêndices articulados. Veja-se a variável `syst_pt[]` da estrutura `mesh`.
- Número do identificador do material que será associado a esta face.

- Definição do lado da normal associado ao cálculo das forças. Os valores permitidos são 0, 1 e 2. Veja-se a função `edm_set_surface_side` para maiores esclarecimentos.

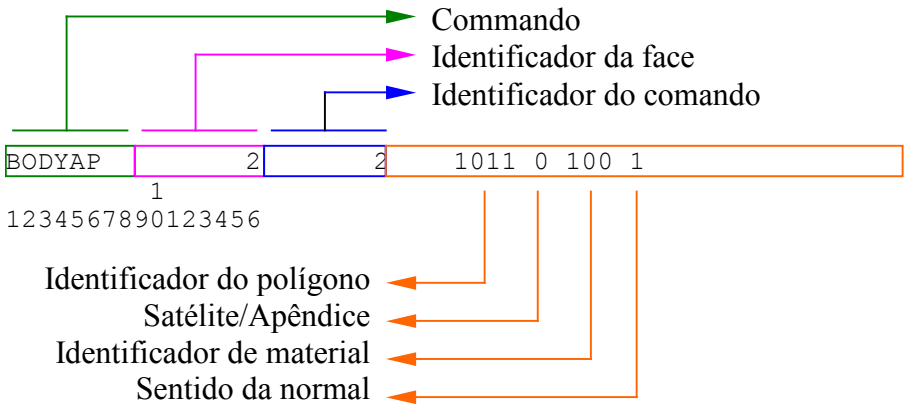

Fig 6.5 – Comando **BODYAP** do arquivo de descrição da geometria.

Quaisquer outros comandos além daqueles especificados acima serão ignorados durante a leitura do arquivo. Podem ser usados, contudo, como comentários ou para nomear o satélite cuja geometria é descrita no arquivo. No exemplo do arquivo de geometria mostrado na Figura 6.6, utiliza-se o comando `SATID` para identificar o satélite – no caso, o satélite CBERS.

```
SATID       CBERS
GRID        1000       0-000.000+1900.00+1100.00       0
GRID        1010       0-000.000+8200.00+1100.00       0
GRID        1020       0+000.000+8200.00-1100.00       0
GRID        1030       0+000.000+1900.00-1100.00       0
GRID        1040       0-1000.00+900.000-1100.00       0
GRID        1050       0-1000.00+900.000+1100.00       0
GRID        1060       0-1000.00-900.000+1100.00       0
GRID        1070       0-1000.00-900.000-1100.00       0
GRID        1080       0+1000.00+900.000-1100.00       0
GRID        1090       0+1000.00+900.000+1100.00       0
GRID        1100       0+1000.00-900.000+1100.00       0
GRID        1110       0+1000.00-900.000-1100.00       0
CQUAD4      1001       1    1000    1010    1020    1030
CQUAD4      1011       1    1080    1090    1100    1110
CQUAD4      1021       1    1070    1110    1100    1060
CQUAD4      1031       1    1050    1090    1080    1040
CQUAD4      1041       1    1080    1110    1070    1040
CQUAD4      1051       1    1070    1060    1050    1040
CQUAD4      1061       1    1060    1100    1090    1050
BODYAP         1       2    1001 1 200 0
BODYAP         2       2    1011 0 100 1
BODYAP         3       2    1021 0 100 1
BODYAP         4       2    1031 0 100 1
BODYAP         5       2    1041 0 100 1
BODYAP         6       2    1051 0 100 1
BODYAP         7       2    1061 0 100 1
MATERIAL     100       3    0.90 0.90 0.80 0.00 1.0 350.
MATERIAL     200       3    0.50 0.50 0.50 0.70 0.8 380.
ENDDATA
```

Figura 6.6 – Arquivo com a descrição da geometria do satélite CBERS.

7 FUNÇÕES PARA LEITURA DO ARQUIVO COM A DESCRIÇÃO DA GEOMETRIA

Foram implementadas 3 funções para efetuar a leitura do arquivo que descreve a geometria do satélite: leitura da tabela de vértices, tabela dos materiais e tabela de faces (ou polígonos). A leitura dos vértices e dos materiais deverá preceder a leitura das faces. Adicionalmente produziu-se uma função para imprimir a geometria em arquivos, de forma a se poder confirmar a leitura correta. Três outras funções foram construídas para permitir a visualização tridimensional da geometria. A primeira delas (`edm_mesh2pov`) destina-se a uma geometria descrita pela estrutura `mesh` ou pelo arquivo de descrição geométrica (veja-se os Capítulos 5 e 6). As demais (`edm_triangle2pov` e `edm_quad2pov`) servem para visualizar geometrias descritas pelas funções de triângulo (`edm_aerodynamic_triangle` e `edm_solar_prsr_triangle`) e quadriláteros (`edm_aerodynamic_quad` e `edm_solar_prsr_quad`). Estas funções escrevem a geometria num arquivo, que posteriormente será lido por um programa em POV para efetuar a síntese da imagem. Descreve-se a seguir cada uma destas funções.

- **`vertex edm_read_vertex (FILE* nfil);`**

  Esta função efetua a leitura dos vértices que definem a geometria do satélite, contida no arquivo apontado por `nfil`, e armazena a tabela numa estrutura do tipo `vertex`.

  Parâmetros de entrada:

  `nfil`

  Apontador do tipo `FILE`, contendo o arquivo previamente aberto por um comando `fopen`.

  Parâmetros de saída:

  `edm_read_vertex`

  Estrutura do tipo `vertex` contendo a tabela de vértices presente no arquivo.

- **`material edm_read_material (FILE* nfil);`**

  Esta função efetua a leitura da tabela de materiais presentes no satélite, contida no arquivo apontado por `nfil`. A tabela será armazenada numa estrutura do tipo `material`.

  Parâmetros de entrada:

  `nfil`

  Apontador do tipo `FILE`, contendo o arquivo previamente aberto por um comando `fopen`.

  Parâmetros de saída:

  `edm_read_material`

  Estrutura do tipo `material` contendo a tabela dos materiais presentes no arquivo.

- **`mesh edm_read_mesh (FILE* nfil, vertex sur_vert, material sur_mat);`**

Esta função efetua a leitura da tabela de faces definida pelo comando `BODYAP` contida no arquivo apontado por `nfil`. A tabela será armazenada numa estrutura do tipo `mesh`.

Parâmetros de entrada:

`nfil`

Apontador do tipo `FILE`, contendo o arquivo previamente aberto por um comando `fopen`.

`sur_vert`

Estrutura do tipo `vertex` contendo a tabela de vértices do satélite.

`sur_mat`

Estrutura do tipo `material` contendo a tabela de materiais.

Parâmetros de saída:

`edm_read_mesh`

Estrutura do tipo `mesh` contendo a tabela de faces (polígonos e materiais associados).

- **`void edm_print_mesh (char *name, vertex sur_grid, mesh sur_mesh, material sur_mat);`**

Esta função imprime as tabelas de vértice, materiais e polígonos em um arquivo especificado pelo nome `name`.

Parâmetros de entrada:

`name`

Constante de caracteres contendo o nome do arquivo (ex: `"cbers_geo.dat"`).

`sur_grid`

Estrutura do tipo `vertex` contendo a tabela de vértices do satélite.

`sur_mesh`

Estrutura do tipo `mesh` contendo a tabela de polígonos.

`sur_mat`

Estrutura do tipo `material` contendo a tabela de materiais.

Parâmetros de saída:

Não há.

- **`void edm_mesh2pov (FILE *nfile, vertex sur_grid, mesh sur_mesh, material sur_mat);`**

  Esta função imprime as tabelas de vértice, materiais e polígonos num arquivo dado pelo apontador de arquivos `nfile`. Este arquivo será utilizado pelo programa POVRay (Persistence Of Vision Raytracer) para gerar imagens da geometria. Veja-se explicação adiante neste capítulo. Compete ao programador efetuar a abertura do arquivo antes da chamada a esta função, bem como proceder ao fechamento do arquivo após a conclusão da descrição completa da geometria. Recomenda-se nomear o arquivo como `mesh2pov.dat` pois o programa em POV foi configurado para ler este arquivo.

  Parâmetros de entrada:

  `nfile`

  Apontador para uma estrutura de arquivo tipo `FILE`.

  `sur_grid`

  Estrutura do tipo `vertex` contendo a tabela de vértices do satélite.

  `sur_mesh`

  Estrutura do tipo `mesh` contendo a tabela de polígonos.

  `sur_mat`

  Estrutura do tipo `material` contendo a tabela de materiais.

  Parâmetros de saída:

  Não há.

- **`void edm_triangle2pov (FILE *nfile, vector3 vertex_1, vector3 vertex_2, vector3 vertex_3, int sys_pt);`**

  Esta função imprime num arquivo dado pelo apontador de arquivos `nfile` as coordenadas de um determinado triângulo que compõe a descrição da geometria de um dado satélite. Esta função deve ser utilizada em conjunto com as funções `edm_aerodynamic_triangle` e `edm_solar_prsr_triangle`, com a intenção de visualizar a geometria e verificar se a descrição está correta. O arquivo será utilizado pelo programa POVRay (Persistence Of Vision Raytracer) para gerar imagens em perspectiva. Veja-se explicação adiante neste capítulo. Compete ao programador efetuar a abertura do arquivo antes da chamada a esta função, bem como proceder ao seu fechamento. Recomenda-se nomear o arquivo como `mesh2pov.dat` pois o programa em POV foi configurado para ler este arquivo.

  Parâmetros de entrada:

  `nfile`

  Apontador para uma estrutura de arquivo tipo `FILE`.

  `vertex_1`

Vetor do tipo `vector3` com a posição do vértice 1 em coordenadas do sistema fixado ao satélite, em metros.

`vertex_2`

Vetor do tipo `vector3` com a posição do vértice 2 em coordenadas do sistema fixado ao satélite, em metros.

`vertex_3`

Vetor do tipo `vector3` com a posição do vértice 3 em coordenadas do sistema fixado ao satélite, em metros.

`sys_pt`

Indicador do sistema de coordenadas ao qual as coordenadas se referem. O Capítulo 5 fornece mais detalhes deste indicador.

Parâmetros de saída:

Não há.

- **`void edm_quad2pov (FILE *nfile, vector3 vertex_1, vector3 vertex_2, vector3 vertex_3, vector3 vertex_4, int sys_pt);`**

Esta função imprime num arquivo dado pelo apontador de arquivos `nfile` as coordenadas de um determinado quadrilátero que compõe a descrição da geometria de um dado satélite. Esta função deve ser utilizada em conjunto com as funções `edm_aerodynamic_quad` e `edm_solar_prsr_quad`, com a intenção de visualizar a geometria e verificar se a descrição está correta. O arquivo será utilizado pelo programa POVRay (Persistence Of Vision Raytracer) para gerar imagens em perspectiva. Para isso veja-se explicação adiante neste capítulo. Compete ao programador efetuar a abertura do arquivo antes da chamada a esta função, bem como proceder ao seu fechamento. Recomenda-se nomear o arquivo como `mesh2pov.dat` pois o programa em POV foi configurado para ler este arquivo.

Parâmetros de entrada:

`nfile`

Apontador para uma estrutura de arquivo tipo `FILE`.

`vertex_1`

Vetor do tipo `vector3` com a posição do vértice 1 em coordenadas do sistema fixado ao satélite, em metros.

`vertex_2`

Vetor do tipo `vector3` com a posição do vértice 2 em coordenadas do sistema fixado ao satélite, em metros.

`vertex_3`

Vetor do tipo `vector3` com a posição do vértice 3 em coordenadas do sistema fixado ao satélite, em metros.

`vertex_4`

Vetor do tipo `vector3` com a posição do vértice 4 em coordenadas do sistema fixado ao satélite, em metros.

`sys_pt`

Indicador do sistema de coordenadas ao qual as coordenadas se referem. O Capítulo 5 fornece mais detalhes deste indicador.

Parâmetros de saída:

Não há.

A Figura 7.1 apresenta a listagem parcial de um programa usado para efetuar a leitura do arquivo da geometria. A Figura 7.2 mostra o arquivo produzido pela função `edm_print_mesh` referente ao arquivo de descrição da geometria mostrado na figura anterior. Nota-se que a geometria impressa não contém quadriláteros (comando `CQUAD4`), pois todos foram transformados em triângulos.

Na Figura 7.2, `MatID` é o identificador do material, `MatP` é o apontador de material (definido internamente), `VxID` é o identificador de vértices, `Vp` é o apontador de vértices, `MsID` é o identificador do polígono (triângulo), `Tp` é o apontador de triângulos, `SysC` é o apontador do sistema de coordenadas (corpo do satélite ou apêndice articulado) e `FcSd` é o identificador de sentido da normal. A diferença entre um identificador e um apontador é que o identificador é um número inteiro unívoco e qualquer, enquanto que o apontador é um número sequencial a partir de zero.

O POV é um programa utilizado para gerar imagens descritas por meio de uma linguagem "script" que se aproxima da linguagem C. O método de síntese é o "raytracing", cujo algoritmo consegue produzir imagens foto-realistas, porém impede que a visualização seja interativa, ou seja, em tempo-real (POVRAY, 2013). O programa "script" escrito em POV para gerar as imagens da geometria do satélite é apresentado no Apêndice A . Nesta seção serão apresentados alguns resultados gerados a partir da geometria descrita na Figura 6.6, e com a aplicação do POV ao arquivo `mesh2pov.dat`. O programa "script" permite selecionar 10 diferentes características do satélite, que modificam as cores de forma a evidenciar uma melhor visualização do parâmetro analisado. São elas:

- face selecionada para o cálculo
- sistema de referência do corpo principal e apêndices
- índice de seletor de materiais
- coeficiente de troca de quantidade de movimento na direção normal
- coeficiente de troca de quantidade de movimento na direção tangencial
- temperatura da superfície, do azul (mais frio) para o vermelho (mais quente)
- coeficiente de reflexão especular
- coeficiente de reflexão difusa
- emissividade superficial
- reflexão difusa e especular

```
void main(void)
{
        vector6 vet_drag, vet_srp;
        vertex sat_vert;
        mesh sat_mesh;
        material sat_mat;
        FILE *nfile;

        nfile = fopen("cbers_geo.dat", "r");

        if (nfile == NULL)
        {
                printf(" Erro na abertura do arquivo de leitura");
                return;
        }

        sat_vert = edm_read_vertex (nfile); // ler tabela de vértices
        sat_mat = edm_read_material (nfile); // ler tabela de materiais
        sat_mesh = edm_read_mesh (nfile, sat_vert, sat_mat); // polígonos

        fclose(nfile);

        sat_vert = 0.001*sat_vert; // transformar milímetros em metros

        // imprimir todas as tabelas
        edm_print_mesh ("sat_trg.dat", sat_vert, sat_mesh, sat_mat);

        // gerar arquivo para o POV
        edm_mesh2pov (sat_vert, sat_mesh, sat_mat);

        // configurar parâmetros de cálculo
        edm_set_drag_properties(vec3_def(1., 0., 0.), 1., 4.);
        edm_set_solar_rad_properties(vec3_def(0., 149.e9, 0.));

        vet_drag = edm_aerodynamic_mesh (sat_mesh, sat_vert, sat_mat);
        vet_srp = edm_solar_prsr_mesh (sat_mesh, sat_vert, sat_mat);

        return;
}
```

Figura 7.1 – Programa para ler e imprimir a geometria de um satélite, e para calcular a força e torque aerodinâmico e de pressão de radiação no satélite CBERS.

```
************************************************* Material table ******************************************
 Number of materials:     2
*********************************************************************************************************
 MatP   MatID Sig_tang Sig_norm Specular  Diffuse Emissiv. SurfTemp
   0      100    0.900    0.900    0.800    0.000    1.000  350.000
   1      200    0.500    0.500    0.500    0.700    0.800  380.000

************************************************* Vertex table ********************************************
 Number of vertexes:    12
*********************************************************************************************************
  Vp     VxID          X          Y          Z
   0     1000     -0.000      1.900      1.100
   1     1010     -0.000      8.200      1.100
   2     1020      0.000      8.200     -1.100
   3     1030      0.000      1.900     -1.100
   4     1040     -1.000      0.900     -1.100
   5     1050     -1.000      0.900      1.100
   6     1060     -1.000     -0.900      1.100
   7     1070     -1.000     -0.900     -1.100
   8     1080      1.000      0.900     -1.100
   9     1090      1.000      0.900      1.100
  10     1100      1.000     -0.900      1.100
  11     1110      1.000     -0.900     -1.100

************************************************* Triangle table ******************************************
 Number of triangles:    14
*********************************************************************************************************
  Tp     MsID SysC MatP FcSd      Vp1      Vp2      Vp3
   0     1001    1    1    0        0        1        2
   1     1001    1    1    0        0        2        3
   2     1011    0    0    1        8        9       10
   3     1011    0    0    1        8       10       11
   4     1021    0    0    1        7       11       10
   5     1021    0    0    1        7       10        6
   6     1031    0    0    1        5        9        8
   7     1031    0    0    1        5        8        4
   8     1041    0    0    1        8       11        7
   9     1041    0    0    1        8        7        4
  10     1051    0    0    1        7        6        5
  11     1051    0    0    1        7        5        4
  12     1061    0    0    1        6       10        9
  13     1061    0    0    1        6        9        5

************************************************* Triangle mesh *******************************************
 Number of triangles:    14
*********************************************************************************************************
   N     MsID SysC MatP FcSd      V1x      V1y      V1z      V2x      V2y      V2z      V3x      V3y      V3z
   0     1001    1    1    0     -0.0      1.9      1.1     -0.0      8.2      1.1      0.0      8.2     -1.1
   1     1001    1    1    0     -0.0      1.9      1.1      0.0      8.2     -1.1      0.0      1.9     -1.1
   2     1011    0    0    1      1.0      0.9     -1.1      1.0      0.9      1.1      1.0     -0.9      1.1
   3     1011    0    0    1      1.0      0.9     -1.1      1.0     -0.9      1.1      1.0     -0.9     -1.1
   4     1021    0    0    1     -1.0     -0.9     -1.1      1.0     -0.9     -1.1      1.0     -0.9      1.1
   5     1021    0    0    1     -1.0     -0.9     -1.1      1.0     -0.9      1.1     -1.0     -0.9      1.1
   6     1031    0    0    1     -1.0      0.9      1.1      1.0      0.9      1.1      1.0      0.9     -1.1
   7     1031    0    0    1     -1.0      0.9      1.1      1.0      0.9     -1.1     -1.0      0.9     -1.1
   8     1041    0    0    1      1.0      0.9     -1.1      1.0     -0.9     -1.1     -1.0     -0.9     -1.1
   9     1041    0    0    1      1.0      0.9     -1.1     -1.0     -0.9     -1.1     -1.0      0.9     -1.1
  10     1051    0    0    1     -1.0     -0.9     -1.1     -1.0     -0.9      1.1     -1.0      0.9      1.1
  11     1051    0    0    1     -1.0     -0.9     -1.1     -1.0      0.9      1.1     -1.0      0.9     -1.1
  12     1061    0    0    1     -1.0     -0.9      1.1      1.0     -0.9      1.1      1.0      0.9      1.1
  13     1061    0    0    1     -1.0     -0.9      1.1      1.0      0.9      1.1     -1.0      0.9      1.1
```

Figura 7.2 – Exemplo de impressão da geometria descrita pelo arquivo mostrado na Figura 6.6.

A Figura 7.3 ilustra uma das possíveis visualizações da geometria do satélite CBERS. Esta imagem foi gerada com a opção 9 (reflexão difusa e especular). Nesta visualização, os coeficientes de reflexão são associados aos respectivos coeficientes utilizados no POV para sintetizar a imagem. O coeficiente especular torna a superfície espelhada, enquanto que o coeficiente difuso dá cor à face (cuja cor foi adotada como amarelo).

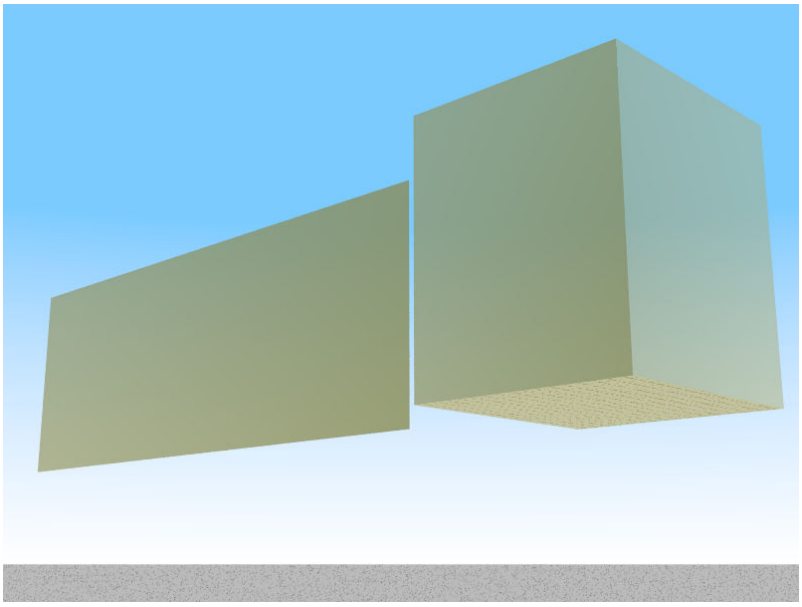

Fig 7.3 – Imagem sintetizada em POV com os coeficientes de reflexão adotados para o CBERS.

O uso das funções para geração de arquivos de visualização da geometria, descrita por triângulos e quadriláteros, é mostrado no código da Figura 7.4, que compõe um satélite formado por uma base em paralelepípedo em conjunto com uma ogiva piramidal. A Figura 7.5 mostra o resultado da síntese da imagem pelo aplicativo POVRay, cujo "script" é apresentado no Apêndice A.

```
FILE *nfile;
nfile = fopen("mesh2pov.dat", "w");

edm_set_drag_surface(0.8, 0.8, 0.1);
edm_set_solar_rad_surface (0.2, 0.8, 1.0, 300.);

edm_set_solar_rad_pressure (vec3_def(0., 0., 1.), 1.);
edm_set_drag_properties(vec3_def(1.0, 0., 0.), 1., 4.);

edm_triangle2pov (nfile, vec3_def(2., 0., 0.), vec3_def(1., .5, .5),
       vec3_def(1., -.5, .5), 0);
edm_triangle2pov (nfile, vec3_def(2., 0., 0.), vec3_def(1., -.5, .5),
       vec3_def(1., -.5, -.5), 0);
edm_triangle2pov (nfile, vec3_def(2., 0., 0.), vec3_def(1., -.5, -.5),
       vec3_def(1., .5, -.5), 0);
edm_triangle2pov (nfile, vec3_def(2., 0., 0.), vec3_def(1., .5, -.5),
       vec3_def(1., .5, .5), 0);

edm_quad2pov (nfile, vec3_def(1., .5, .5), vec3_def(-1., .5, .5),
       vec3_def(-1., -.5, .5), vec3_def(1., -.5, .5),0);
edm_quad2pov (nfile, vec3_def(1., -.5, .5), vec3_def(-1., -.5, .5),
       vec3_def(-1., -.5, -.5), vec3_def(1., -.5, -.5),0);
edm_quad2pov (nfile, vec3_def(1., -.5, -.5), vec3_def(-1., -.5, -.5),
       vec3_def(-1., .5, -.5), vec3_def(1., .5, -.5),0);
edm_quad2pov (nfile, vec3_def(1., .5, -.5), vec3_def(-1., .5, -.5),
       vec3_def(-1., .5, .5), vec3_def(1., .5, .5),0);

edm_quad2pov (nfile, vec3_def(-1., .5, .5), vec3_def(-1., .5, -.5),
       vec3_def(-1., -.5, -.5), vec3_def(-1., -.5, .5),0);

fclose(nfile);
```

Figura 7.4 – Programa para gerar o arquivo de dados para a visualização da geometria.

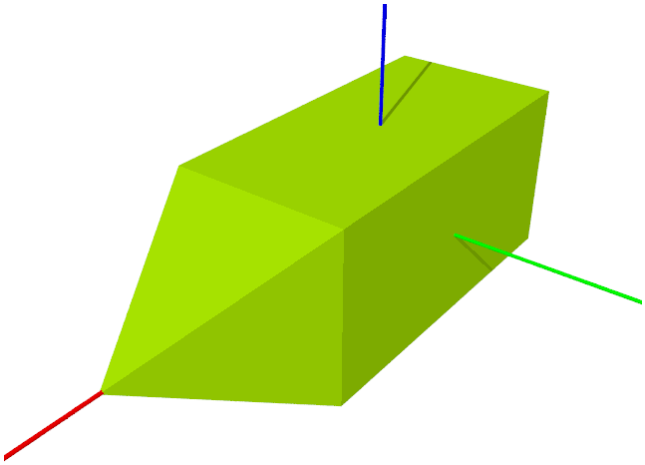

Fig 7.5 – Imagem sintetizada em POV com geometria descrita com triângulos e quadriláteros.

8 FUNÇÕES PARA CÁLCULO DO TORQUE DE GRADIENTE DE GRAVIDADE

O torque de gradiente de gravidade decorre do fato de que certas partes do satélite encontram-se em alturas diferentes, e, portanto, sujeitos a diferentes acelerações uma vez que a força gravitacional decresce com o quadrado da distância ao centro da Terra. Embora estas diferenças na aceleração sejam diminutas, elas conseguem gerar um torque que tende a fazer com que o eixo do menor momento de inércia do satélite fique alinhado com a direção vertical local. Este comportamento é por vezes aproveitado para fazer com que uma face do satélite fique apontada para a Terra, nos satélites com atitude estabilizada por gradiente de gravidade.

Foram implementadas 3 funções para o cálculo do gradiente de gravidade: `edm_set_inertia` e `edm_get_inertia` para definir e recuperar a matriz de inércia do satélite, e `edm_gravity_gradient` para calcular o torque, descritas a seguir. A Seção 12.4 mostra um exemplo de uso das funções apresentadas neste capítulo.

- **`void edm_set_inertia (matrix3 inertia);`**

  A função `edm_set_inertia` atribui a matriz de inércia a ser utilizada no cômputo do torque de gradiente de gravidade.

  Parâmetros de entrada:

  `inertia`

  Matriz do tipo `matrix3` com as componentes da matriz de inércia corrente do satélite, em relação ao centro de massa, em kg m^2.

  Parâmetros de saída:

  Não há.

- **`matrix3 edm_get_inertia ();`**

  A função `edm_get_inertia` recupera o valor corrente da matriz de inércia a ser utilizada no cômputo do torque de gradiente de gravidade.

  Parâmetros de entrada:

  Não há.

  Parâmetros de saída:

  `edm_get_inertia`

  Matriz do tipo `matrix3` com as componentes da matriz de inércia corrente do satélite, em relação ao centro de massa, em kg m^2, conforme definida pela função `edm_set_inertia`.

- **`vector3 edm_gravity_gradient (vector3 sat_pos, matrix3 rmx_att);`**

Esta função calcula o torque de gradiente de gravidade num satélite cuja matriz de inércia foi fornecida pela função `edm_set_inertia`.

Parâmetros de entrada:

`sat_pos`

Vetor do tipo `vector3` contendo a posição do satélite em coordenadas retangulares no sistema de referência geocêntrico inercial, em metros.

`rmx_att`

Matriz do tipo `matrix3` contendo a matriz de atitude do satélite, tal que o produto desta matriz por um vetor qualquer fornecido no sistema geocêntrico inercial transforma este vetor para o sistema de coordenadas do satélite.

Parâmetros de saída:

`edm_gravity_gradient`

Vetor do tipo `vector3` com as componentes do torque de gradiente de gravidade, no sistema de coordenadas fixado ao satélite, em Nm.

9 FUNÇÕES PARA CÁLCULO DO TORQUE MAGNÉTICO RESIDUAL

O torque magnético constitui uma das maiores perturbações na atitude de satélites. Origina-se da presença de materiais magnetizados ou de altas permeabilidades magnéticas a bordo. São também fontes de momentos magnéticos eventuais correntes elétricas que formam laços, particularmente em painéis solares para geração de energia. A interação deste momento magnético com o campo geomagnético irá provocar um pequeno torque no satélite, dado pelo produto do momento pelo campo. Foram implementadas aqui funções para definir (atribuir) valores ao momento magnético do satélite, recuperar o último valor atribuído e calcular o torque. A Seção 12.5 ilustra o uso das funções destinadas ao cálculo do torque magnético. Estas funções são descritas a seguir.

- **`void edm_set_magnetic_moment (vector3 mag_moment);`**

  Esta função atribui o valor corrente a ser usado no cálculo do torque magnético.

  Parâmetros de entrada:

  `mag_moment`

  Vetor do tipo `vector3` contendo o momento magnético atual em A m^2, no sistema de referência fixado ao corpo do satélite.

  Parâmetros de saída:

  Não há

- **`vector3 edm_get_magnetic_moment ();`**

  Esta função devolve o valor corrente do momento magnético residual do satélite empregado no cálculo do torque magnético.

  Parâmetros de entrada:

  Não há

  Parâmetros de saída:

  `edm_get_magnetic_moment`

  Vetor do tipo `vector3` contendo o momento magnético atual em A m^2, no sistema de referência fixado ao corpo do satélite.

- **`vector3 edm_magnetic_torque (vector3 b_sat_coord);`**

  A função `edm_magnetic_torque` calcula o torque devido ao momento magnético residual, em Nm, no sistema de referência fixado ao satélite.

  Parâmetros de entrada:

  `b_sat_coord`

Vetor do tipo `vector3` contendo o campo magnético terrestre, em Tesla, no sistema de coordenadas fixado ao satélite.

Parâmetros de saída:

`edm_magnetic_torque`

Vetor do tipo `vector3` contendo o torque magnético que age no satélite.

10 FUNÇÕES PARA CÁLCULO DA ACELERAÇÃO LUNI-SOLAR

A aceleração sofrida por um satélite terrestre devido à gravidade da Lua e do Sol afeta os elementos keplerianos da órbita. O principal efeito é a precessão dos nodos, que se sobrepõe ao mesmo efeito causado pelo achatamento dos pólos da Terra – este último é predominante. A atração da Lua e do Sol afeta também a inclinação orbital, produzindo um efeito de longo período. O modelo introduzido aqui para cálculo da aceleração é baseado na lei da gravitação universal de Newton, mas considerando o problema dos 3 corpos aplicado a um sistema inercial centrado na Terra. A aceleração devido ao terceiro corpo será dada por:

$$\mathbf{a}_{sat} = \mu_G \left(\frac{\mathbf{r}_G - \mathbf{r}_{sat}}{|\mathbf{r}_G - \mathbf{r}_{sat}|^3} - \frac{\mathbf{r}_G}{|\mathbf{r}_G|^3} \right), \tag{1.8}$$

onde μ_G é a constante gravitacional da Lua ou Sol (μ_M = 4.902794 10^{12} m^3/s^2, μ_S = 1.32712438 10^{20} m^3/s^2), $\mathbf{r}_G$ é o vetor posição da Lua ou Sol no sistema geocêntrico inercial, e $\mathbf{r}_{sat}$ é o vetor posição do satélite neste mesmo sistema, ambos fornecidos em unidades métricas. As posições do Sol e da Lua podem ser obtidas por meio das funções `vector3 sun_coord (int output_sel, int djm, double ts)`, e `vector6 moon_coord (int djm, double ts)`, integrantes da biblioteca `orbit.cpp` (CARRARA, 2007). A Seção 12.6 apresenta um exemplo de uso destas funções, descritas a seguir.

- **`vector3 edm_moon_accel (vector3 sat_iner, vector3 moon_iner);`**

  A função `edm_moon_accel` calcula a aceleração sofrida pelo satélite devido à gravidade da Lua, no sistema de referência geocêntrico inercial.

  Parâmetros de entrada:

  `sat_iner`

  Vetor do tipo `vector3` contendo a posição do satélite em coordenadas geocêntricas inerciais em metros.

  `moon_iner`

  Vetor do tipo `vector3` contendo a posição da Lua em coordenadas geocêntricas inerciais, em metros. Esta posição é obtida pela função `moon_cord`, do pacote `orbit.cpp`.

  Parâmetros de saída:

  `edm_moon_accel`

  Vetor do tipo `vector3` contendo a aceleração devido à gravidade da Lua.

- **`vector3 edm_sun_accel (vector3 sat_iner, vector3 sun_iner);`**

  A função `edm_sun_accel` calcula a aceleração sofrida pelo satélite devido à gravidade do Sol, no sistema de referência geocêntrico inercial.

  Parâmetros de entrada:

`sat_iner`

Vetor do tipo `vector3` contendo a posição do satélite em coordenadas geocêntricas inerciais em metros.

`sun_iner`

Vetor do tipo `vector3` contendo a posição do Sol em coordenadas geocêntricas inerciais, em metros. Esta posição é obtida pela função `sun_cord`, do pacote `orbit.cpp`.

Parâmetros de saída:

`edm_sun_accel`

Vetor do tipo `vector3` contendo a aceleração devido à gravidade do Sol.

11 FUNÇÕES PARA CÁLCULO DA ACELERAÇÃO DO POTENCIAL GRAVITACIONAL TERRESTRE

O potencial terrestre é modelado em série de harmônicos esféricos, ou polinômios de Legendre. O algoritmo para cálculo da série foi implementado por Kuga em Fortran, e convertido para C por este autor. Duas funções foram construídas: a primeira efetua a leitura dos coeficientes dos harmônicos esféricos a partir de um arquivo contendo o modelo do geopotencial, e a segunda calcula as acelerações numa dada localização sobre o geóide, no sistema de coordenadas geocêntrico terrestre. O presente modelo do geopotencial é o EGM96 (EGM, 2012) com resolução de 1 grau. A ordem do polinômio é portanto de 360, mas o algoritmo suporta qualquer resolução, como, por exemplo, o modelo EGM2008, cujos coeficientes atingem a ordem de 2159 (2,5 minutos de arco). O arquivo de coeficientes dos polinômios deve estar presente na pasta de execução do programa, e deve ser fornecido como argumento à função de leitura `leg_initialize` descrita a seguir. A utilização destas funções é demonstrada na Seção 12.7.

- **`void leg_initialize(char *name, int nmax);`**

  A função `leg_initialize` efetua a leitura dos coeficientes dos harmônicos esféricos para cálculo da aceleração devida ao geopotencial terrestre. A memória necessária para armazenar os coeficientes é alocada durante a leitura. O modelo atual dos coeficientes é o EGM96, cujo arquivo é `egm96.dat`, com coeficientes até a ordem 360. Contudo, pode-se usar o modelo com ordem menor do que a ordem máxima, bastando selecionar a ordem desejada no parâmetro de entrada `nmax`. O valor desta variável não é verificado pela função e, portanto, poderá ocorrer um erro caso ela seja maior do que a ordem máxima contida no arquivo.

  <u>Parâmetros de entrada:</u>

  `name`

  Vetor de caracteres com o nome do arquivo com os coeficientes dos harmônicos esféricos. Na presente implementação deverá ser igual a `"egm96.dat"`.

  `nmax`

  Variável do tipo `int` contendo a ordem máxima dos coeficientes a serem lidos. Deverá ser igual ou menor do que 360 caso o arquivo de coeficientes seja o `egm96.dat`.

  <u>Parâmetros de saída:</u>

  `leg_initialize`

  Não há variável de retorno.

- **`vector3 leg_forcol_ac (int nm, vector3 x);`**

  A função `leg_forcol_ac` calcula a aceleração devida ao geopotencial terrestre no sistema de coordenadas geocêntrico terrestre, numa dada localização sobre o geóide.

Parâmetros de entrada:

`nm`

Variável do tipo `int` passada por valor contendo a ordem máxima a ser usada no cálculo da aceleração. Deverá ser igual ou menor do que `nmax` (veja-se `leg_initialize`).

`x`

Variável do tipo `vector3` passada por referência contendo a posição para o cálculo da aceleração, no sistema de coordenadas geocêntrico terrestre, em metros.

Parâmetros de saída:

`leg_forcol_ac`

Variável do tipo `vector3` contendo a aceleração devida ao geopotencial, no sistema de coordenadas geocêntrico terrestre, em m/s^2.

12 EXEMPLOS DE UTILIZAÇÃO DAS FUNÇÕES

Apresenta-se nesta seção diversos exemplos de utilização das funções descritas nas seções anteriores. São apresentados exemplos de todos os modelos implementados no pacote, ou seja, forças e torques aerodinâmicos, de pressão de radiação, torque de gradiente de gravidade, torque magnético, aceleração Luni-Solar e aceleração do geopotencial. Inicialmente descreve-se um exemplo de um código para girar o painel do satélite CBERS.

12.1 Rotação de geometria

Foi mencionado, no Capítulo 5, que a malha de descrição da geometria conta com um vetor de índices de componente do triângulo, `syst_pt[]`, cuja finalidade é identificar os triângulos da geometria que constituem partes móveis do satélite, como painéis solares rotativos, braços robóticos, antenas direcionais, etc. No exemplo que se segue, será mostrado como efetuar a rotação do painel do satélite CBERS, cuja geometria já foi apresentada no Capítulo 7. Para efetuar a rotação, deve-se aplicar uma transformação nas coordenadas dos vértices, que ficam armazenados na estrutura do tipo `vertex`. A transformação, contudo, pode sobrescrever os valores originais, que assim seriam perdidos, caso fosse necessário efetuar novas transformações. Logo, é necessário criar uma duplicata das coordenadas dos vértices, que irá armazenar as coordenadas transformadas, enquanto que a estrutura original mantém as coordenadas inalteradas. Na listagem da Figura 12.1, as coordenadas originais são armazenadas na estrutura `sat_vert` enquanto que a estrutura `rot_vert` conterá as coordenadas modificadas. Após a leitura da geometria nas estruturas `sat_vert` (coordenadas dos vértices), `sat_mat` (materiais) e `sat_mesh` (tabela de triângulos), usa-se a função `alloc_vertex` para alocar a memória necessária para os vértices modificados (`rot_vert`). A seguir efetua-se uma mudança de unidades nos vértices originais, que passam de milímetros, fornecidos no arquivo, para metros, como requerido para os cálculos das forças e torques. O número de vértices é então transferido para a nova variável (`rot_vert.numb_vtx = sat_vert.numb_vtx;`), e um laço permite transferir as coordenadas e a identificação dos vértices para a estrutura `rot_vert`.

A rotação pode ser então efetuada, bastando para isso que se obtenha a matriz de transformação, e que ela seja aplicada somente nas coordenadas dos vértices que componham o apêndice, selecionado por meio da componente da estrutura `syst_pt`. No código mostrado na Figura 12.2 o painel do CBERS é girado de 60 graus sobre o eixo y. Nota-se que a transformação a ser efetuada deve ser invertida (matriz transposta), pois objetos giram em sentido contrário à de uma rotação num sistema de coordenadas. O efeito da rotação é visualizado na Figura 12.3, feita a partir do arquivo gerado para o POV, apresentado no Apêndice A. Para definir as matrizes de rotação são utilizadas funções da biblioteca `attaux.cpp` que faz parte do pacote `attpro` (CARRARA, 2007).

Contudo, nem sempre é possível efetuar a rotação a partir de um ângulo conhecido. O apontamento de um painel, por exemplo, é feito por meio de uma eletrônica de controle que opera em malha fechada, de forma a maximizar a energia captada pelas células. Quando na sombra da Terra, a eletrônica comanda o painel para seguir uma referência em velocidade angular, para que, ao retornar na região iluminada, o erro de apontamento seja o menor possível. Deve-se considerar, além disso, que raramente um painel solar aponta diretamente para o Sol, pois o eixo de rotação na maioria dos satélites coincide com a normal ao plano orbital, e, este, em geral, não é perpendicular à direção do Sol. A condição de que o erro de apontamento seja mínimo leva à constatação de que a projeção da direção do Sol, no plano

perpendicular ao eixo de rotação do painel, coincida com a normal deste. Se **s** for o versor que fornece a direção do Sol no sistema de referência do satélite, e **a** for o versor que indica o eixo de rotação do painel no mesmo sistema, então a direção da normal ao painel **p** será encontrada por:

$$\mathbf{p} = \mathbf{s} - (\mathbf{s} \cdot \mathbf{a})\, \mathbf{a}\,. \tag{1.9}$$

```
vertex sat_vert, rot_vert;
mesh sat_mesh;
material sat_mat;

FILE *nfile;
nfile = fopen("cbers_geo.dat", "r");

if (nfile == NULL)
{
      printf(" Erro na abertura do arquivo de leitura");
}
else
{
      sat_vert = edm_read_vertex (nfile);
      sat_mat = edm_read_material (nfile);
      sat_mesh = edm_read_mesh (nfile, sat_vert, sat_mat);
      fclose(nfile);
      rot_vert = alloc_vertex(sat_vert.numb_vtx);
}
sat_vert = 0.001*sat_vert;

rot_vert.numb_vtx = sat_vert.numb_vtx;
for (i = 0; i < sat_vert.numb_vtx; i++)
{
      rot_vert.vtx_id[i] = sat_vert.vtx_id[i];
      rot_vert.vtx_tab[i] = sat_vert.vtx_tab[i];
}

fclose (nfile);
```

Figura 12.1 – Código usado para gerar uma cópia das coordenadas dos vértices.

```
// definir matriz de rotação
mat_rot = transpose(rotmay(60.*RADIANS));

// efetuar a rotação do painel
for (i = 0; i < sat_mesh.numb_trg; i++)
{
      // localizar o elemento do painel (com sistema = 1)
      if (sat_mesh.syst_pt[i] == 1)
      {
            rot_vert.vtx_tab[sat_mesh.trg_tab[i]._1] =
                  mat_rot*sat_vert.vtx_tab[sat_mesh.trg_tab[i]._1];
            rot_vert.vtx_tab[sat_mesh.trg_tab[i]._2] =
                  mat_rot*sat_vert.vtx_tab[sat_mesh.trg_tab[i]._2];
            rot_vert.vtx_tab[sat_mesh.trg_tab[i]._3] =
                  mat_rot*sat_vert.vtx_tab[sat_mesh.trg_tab[i]._3];
      }
}
// imprimir resultados
edm_mesh2pov (rot_vert, sat_mesh, sat_mat);
```

Figura 12.2 – Código para efetuar uma rotação no apêndice 1 do satélite.

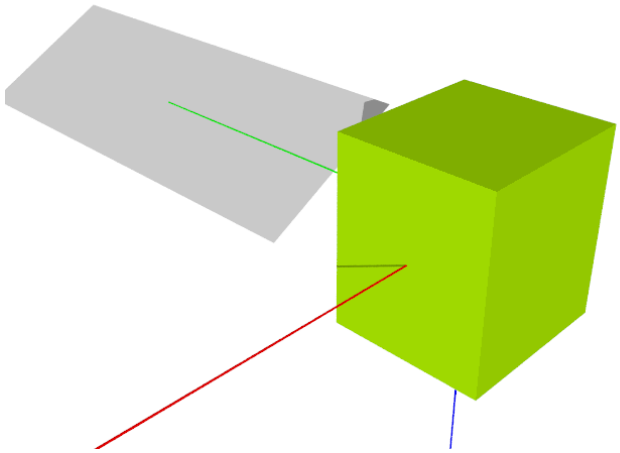

Figura 12.3 – Rotação efetuada no painel do satélite CBERS.

A normal ao painel **n** tem a mesma direção de **p**, porém possui módulo unitário. Logo **n** = **p** / |**p**|. Pode-se agora definir um terceiro versor perpendicular tanto a **n** quanto a **a**, e que esteja no plano do painel:

$$\mathbf{r} = \mathbf{n} \times \mathbf{a} \,. \tag{1.10}$$

Estes três versores, **n, a** e **r**, constituem a base que indica a posição do painel relativo ao sistema de coordenadas fixado ao satélite. Se **i**, **j** e **k** constituírem a base de versores deste último sistema, então a matriz de rotação do painel será dada por:

$$\mathbf{C}_{ps} = \begin{pmatrix} \mathbf{n}\cdot\mathbf{i} & \mathbf{n}\cdot\mathbf{j} & \mathbf{n}\cdot\mathbf{k} \\ \mathbf{a}\cdot\mathbf{i} & \mathbf{a}\cdot\mathbf{j} & \mathbf{a}\cdot\mathbf{k} \\ \mathbf{r}\cdot\mathbf{i} & \mathbf{r}\cdot\mathbf{j} & \mathbf{r}\cdot\mathbf{k} \end{pmatrix}, \tag{1.11}$$

que, multiplicada por um vetor conhecido no sistema do satélite, fornecerá as componentes deste vetor no sistema do painel. Uma vez que as coordenadas dos vértices do painel são conhecidas no sistema fixado ao próprio painel, então se deve utilizar a transposta da matriz acima para efetuar a transformação, ou seja

$$\mathbf{v}_s = \mathbf{C}_{sp}\ \mathbf{v}_p = \mathbf{C}_{ps}^T\ \mathbf{v}_p \tag{1.12}$$

onde $\mathbf{v}_p$ é um vetor com coordenadas de um vértice no sistema do painel, que corresponde à posição inicial do vértice no sistema do satélite, e $\mathbf{v}_p$ representa as componentes deste vetor no sistema fixado ao satélite após ter sido girado. Contudo, eventualmente o eixo de rotação do painel não passa pela origem do sistema de coordenadas, o que poderia fazer com que o painel girasse de forma excêntrica. Para que a rotação seja efetuada corretamente, introduz-se um vetor de translação **d**, que fornece as coordenadas de um ponto qualquer do eixo de rotação do painel no sistema de coordenadas fixado ao satélite, em unidades métricas. A transformação final fica então:

$$\mathbf{v}_s = \mathbf{C}_{ps}^T\ (\mathbf{v}_p - \mathbf{d}) + \mathbf{d} \tag{1.13}$$

Para o CBERS o versor **a** tem a mesma direção do eixo y, isto é, $\mathbf{a} = (0\ 1\ 0)^T$ e **d** é nulo ($\mathbf{d} = (0\ 0\ 0)^T$). O código C da Figura 12.4 pode usado para se efetuar a rotação do painel deste satélite.

```
// ************************************** apontar painéis

eixr   = vec3_def(0., 1., 0.);
desl   = vec3_def(0., 0., 0.);

sun_v  = sunp - (sunp*eixr)*eixr;       // direção da normal
normal = unity_vector(sun_v);           // normal do painel
vecr   = normal^eixr;                   // versor no plano do painel
mat_rot._1 = normal;             // matriz de rotação do painel
mat_rot._2 = eixr;         // matriz de rotação do painel
mat_rot._3 = vecr;         // matriz de rotação do painel

for (i = 0; i < sat_mesh.numb_trg; i++)
{
       // localizar o elemento do painel (tem sistema = 1)
       if (sat_mesh.syst_pt[i] == 1)
       {
              rot_vert.vtx_tab[sat_mesh.trg_tab[i]._1] =
                     mat_rot*(sat_vert.vtx_tab[sat_mesh.trg_tab[i]._1]
                      - desl) + desl;
              rot_vert.vtx_tab[sat_mesh.trg_tab[i]._2] =
                     mat_rot*(sat_vert.vtx_tab[sat_mesh.trg_tab[i]._2]
                      - desl) + desl;
              rot_vert.vtx_tab[sat_mesh.trg_tab[i]._3] =
                     mat_rot*(sat_vert.vtx_tab[sat_mesh.trg_tab[i]._3]
                      - desl) + desl;
       }
}
```

Figura 12.4 – Código para apontar o painel do satélite CBERS para o Sol.

12.2 Arrasto aerodinâmico

A Figura 12.5 mostra a listagem de um programa que usa a função `edm_aerodynamic_plane` para calcular o coeficiente de arrasto numa placa plana sob diferentes ângulos de incidência do fluxo para diversos valores da razão de velocidades s, mostrado na Figura 12.6. Nota-se que o arrasto não é nulo mesmo em ângulos de incidência acima de 90°, e que o coeficiente de arrasto tende ao valor 2 quando s for grande, compatível com resultados teóricos e experimentais. O coeficiente de arrasto C_D é definido como o produto escalar da força aerodinâmica $\mathbf{F}_{aer}$ pela direção de incidência das moléculas, dividido pela pressão aerodinâmica e pela área de referência A_r, ou seja:

$$C_D = -\frac{2}{\rho_i\ \mathbf{v}^2\ A_r}\frac{\mathbf{F}_{aer}\cdot\mathbf{v}}{|\mathbf{v}|}, \tag{1.14}$$

onde ρ_i é densidade local da atmosfera e **v** é a velocidade do satélite relativo à atmosfera. O sinal negativo no coeficiente de arrasto indica que a força tem sentido contrário à velocidade.

```
#include "stdafx.h"
void main(void)
{
        vector3 veloc, snorm;
        double cd, alfa, sratio;

        FILE *nfile;
        nfile = fopen("drag.dat", "w");

        snorm = vec3_def(0., 1., 0.);
        edm_set_drag_surface(1., 1., 1.);

        for (alfa = 0; alfa < 180.; alfa++)
        {
                printf ("%8.2f  ", alfa);
                fprintf (nfile, "%8.2f  ", alfa);

                for (sratio = 2; sratio < 9; sratio += 2)
                {
                        veloc = vec3_def(0., cos(alfa*RADIANS), sin(alfa*RADIANS));
                        edm_set_drag_properties(veloc, 1., sratio);
                        cd = veloc*edm_aerody (snorm, 1.);
                        printf ("%12.6f ", -cd);
                        fprintf (nfile, "%12.6f ", -cd);
                }
                printf ("\n");
                fprintf (nfile, "\n");
        }
        fclose(nfile);
}
```

Figura 12.5 – Exemplo de utilização da função `edm_aerodynamic_plane`.

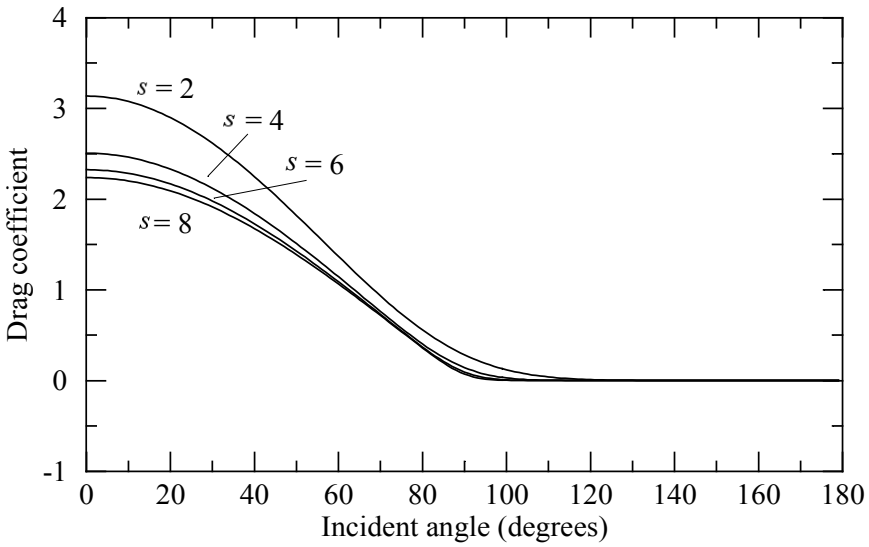

Figura 12.6 – Coeficiente de arrasto numa placa plana sob diferentes ângulos de incidência do fluxo.

A Figura 12.7 apresenta o coeficiente de arrasto num cilindro em função do ângulo de ataque (direção da velocidade) em relação ao plano perpendicular ao eixo de simetria. Os resultados foram comparados com a solução analítica do arrasto num cilindro, mostrando perfeita coincidência, exceto por pequenas diferenças causadas por erros de integração numérica. O cilindro foi descrito pelos seus vértices, usando-se para isso a função

`edm_aerodynamic_quad`, cujo código fonte pode ser visto no código mostrado na Figura 12.8.

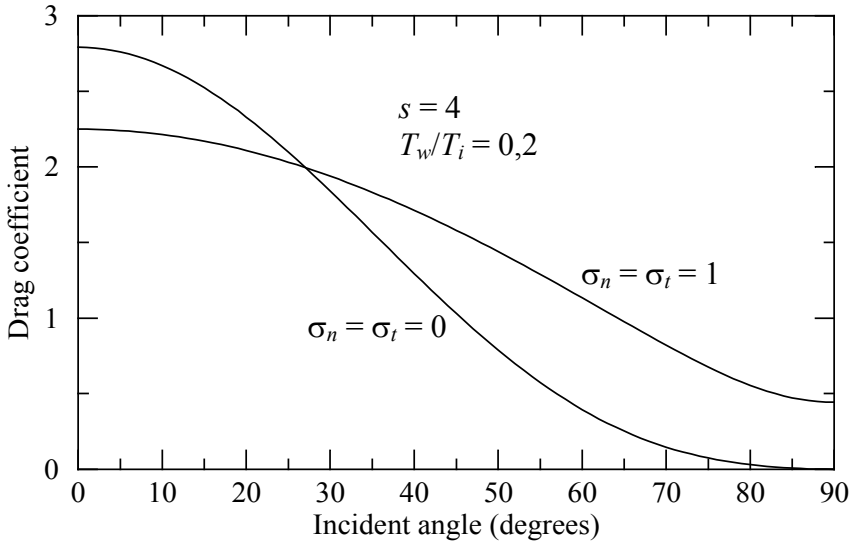

Figura 12.7 – Integração numérica do coeficiente de arrasto num cilindro em função do ângulo de incidência do fluxo.

```
#include "stdafx.h"
void main(void)
{
	vector3 veloc, snorm;
	vector3 v1, v2, v3, v4, f_t;
	double cd, alfa, sratio, sg, teta, dteta, tetad;
	FILE *nfile;

	nfile = fopen("drag.dat", "w");
	sratio = 4;

	for (alfa = 0; alfa < 91.; alfa++)
	{
		printf ("%8.2f  ", alfa);
		fprintf (nfile, "%8.2f  ", alfa);
		veloc = vec3_def(cos(alfa*RADIANS), 0., sin(alfa*RADIANS));

		for (sg = 0; sg < 2; sg++)
		{
			edm_set_drag_surface(sg, sg, 0.4472);
			teta = 0;
			dteta = 2;
			tetad = teta + dteta;
			v1    = vec3_def(cos(teta*RADIANS), sin(teta*RADIANS), 0.5);
			v2    =      vec3_def(cos(teta*RADIANS), sin(teta*RADIANS), -0.5);
			f_t   = vec6_def(0., 0., 0., 0., 0., 0.);

			while (tetad < 361)
			{
				edm_set_drag_properties(veloc, 1., sratio);
				v3    = vec3_def(cos(tetad*RADIANS), sin(tetad*RADIANS), -0.5);
				v4    = vec3_def(cos(tetad*RADIANS), sin(tetad*RADIANS), 0.5);
				f_t   = f_t + edm_aerodynamic_quad (v1, v2, v3, v4);
				v1 = v4;
				v2 = v3;
				tetad += dteta;
			}
			cd = veloc*f_t._1/2;		// 2 é a área projetada
			printf ("%12.6f ", -cd);
			fprintf (nfile, "%12.6f ", -cd);
		}
		printf ("\n");
		fprintf (nfile, "\n");
	}
	fclose(nfile);
	return;
}
```

Figura 12.8 – Exemplo de utilização da função `edm_aerodynamic_quad`.

As funções que armazenam a geometria do satélite em estruturas foram utilizadas na configuração do satélite CBERS, mostrado na Figura 12.9. Os parâmetros utilizados foram aqueles mostrados na Figura 6.6, e variou-se o ângulo de ataque, ou ângulo de arfagem, entre os limites –45 a 45°. Na Figura 12.9 o eixo x de rolamento é vermelho, arfagem y é verde e guinada z é azul. O coeficiente de torque ao redor do eixo z é mostrado na Figura 12.10. O coeficiente de torque é definido como a relação entre o torque e o produto da pressão aerodinâmica pela área frontal (ou uma área de referência), ou seja:

$$C_i = \frac{2}{\rho v^2 A_r} T_i, \tag{1.15}$$

na qual T_i é o torque no eixo i, ρ é a densidade da atmosfera, v é a velocidade do satélite em relação à atmosfera e Ar é a área de referência, adotada como igual a 6,93 m^2 para o satélite CBERS. Considerou-se nos cálculos uma temperatura local da atmosfera de 1000°K. A Figura 12.10 mostra o coeficiente de torque para dois valores da razão de velocidades: 4 e 8. Nota-se que este coeficiente tanto pode ser positivo quanto negativo. No caso do CBERS, o valor do

coeficiente indica que o painel exerce um torque aerodinâmico positivo ao redor do eixo *z*, como era esperado. A Figura 12.11 apresenta a listagem do programa utilizado para gerar as curvas mostradas na Figura 12.10.

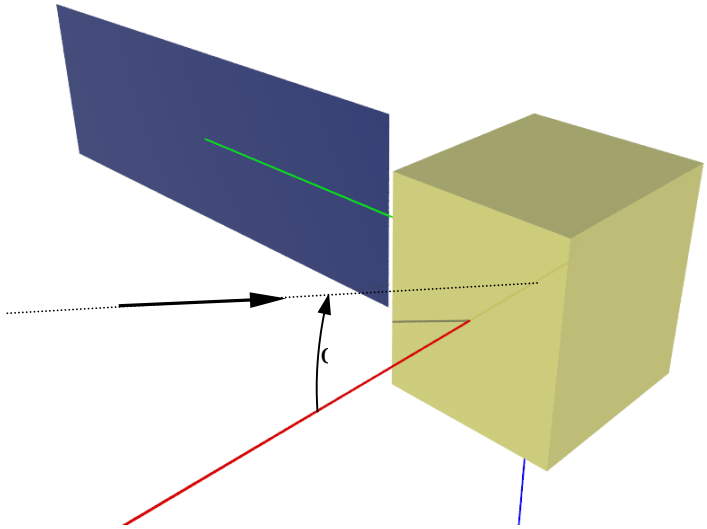

Figura 12.9 – Geometria do satélite CBERS. O ângulo de arfagem α situa-se no plano *x*-*z*.

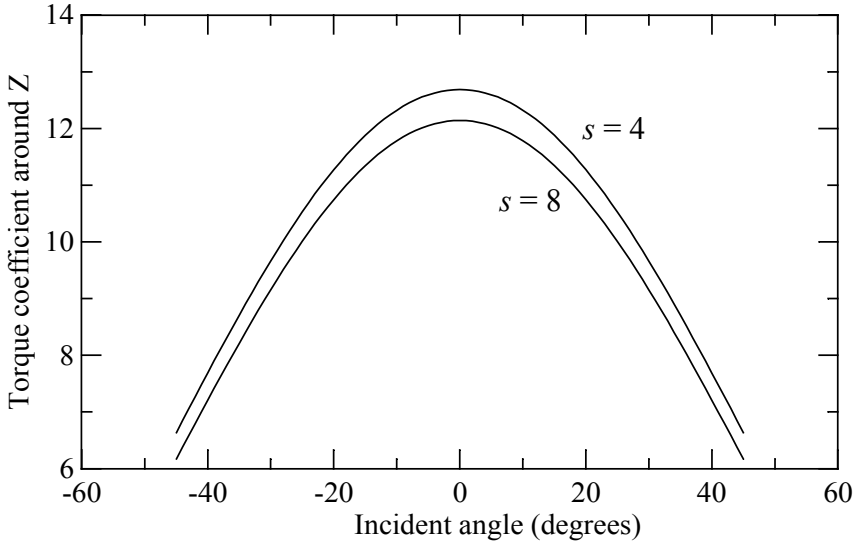

Figura 12.10 – Integração numérica do coeficiente de torque ao redor do eixo *z* para o CBERS, em função do ângulo de ataque α.

```
void main(void)
{
      vector3 vert1, vert2, vert3, veloc, snorm;
      vector6 f_t;
      vertex sat_vert, vert_sat;
      mesh sat_mesh;
      material sat_mat;
      double cd, alfa, sratio, sg, area_frontal;
      double teta, dteta, tetad;

      unsigned char surface_side;

      FILE *nfile;
      nfile = fopen("cbers_geo.dat", "r");

      sat_vert = edm_read_vertex (nfile);
      sat_mat = edm_read_material (nfile);
      sat_mesh = edm_read_mesh (nfile, sat_vert, sat_mat);
      fclose(nfile);

      sat_vert = 0.001*sat_vert;    // conversão de unidades
      edm_print_mesh ("sat_trg.dat", sat_vert, sat_mesh, sat_mat);
      edm_mesh2pov (sat_vert, sat_mesh, sat_mat);
      area_frontal = 2.2*(8.2-1.9) + 2.2*1.8;   // painel + satelite

      nfile = fopen("drag.dat", "w");
      // Ajuste da temperatura (1000K). Outros parâmetros não são utilizados
      edm_set_environ_properties(vec3_def(1., 0., 0.), 1., 1000., 1.);

      for (alfa = -45; alfa < 46.; alfa++)
      {
            printf ("%8.2f  ", alfa);
            fprintf (nfile, "%8.2f  ", alfa);

            veloc = vec3_def(cos(alfa*RADIANS), 0., -sin(alfa*RADIANS));

            for (sratio = 4; sratio < 9; sratio +=4)
            {
                  edm_set_drag_properties(veloc, 1., sratio);
                  f_t = edm_aerodynamic_mesh (sat_mesh, sat_vert, sat_mat);
                  cd = veloc*f_t._1/area_frontal;
                  f_t._2 = f_t._2/area_frontal;
                  printf ("%12.6f ", -cd);
                  fprintf (nfile, "%12.6f ", -cd);
                  fprintf (nfile, "%8.3f %8.3f %8.3f ", f_t._2._1,  f_t._2._2, f_t._2._3);
            }
            printf ("\n");
            fprintf (nfile, "\n");
      }
      fclose(nfile);
      return;
}
```

Figura 12.11 – Exemplo de cálculo usando as estruturas que armazenam a geometria.

12.3 Pressão de radiação solar

As funções para cálculo das forças e torques de pressão de radiação solar foram aplicadas num cilindro, sem incluir o efeito das faces superior e inferior. A listagem deste exemplo é mostrada na Figura 12.12 que evidencia o cálculo em três situações: superfície especular (e = 1, δ = 0), superfície difusa (e = 0, δ = 1) e corpo negro (e = 0, δ = 0). Em todas as situações a temperatura da superfície foi de 330°K, porém essa informação é irrelevante, uma vez que os resultados não são afetados quando a temperatura for constante na superfície. Os resultados são mostrados na Figura 12.13, e comparados com valores obtidos da solução analítica da pressão de radiação para um cilindro, mostrando perfeita concordância. As curvas mostram o coeficiente de pressão de radiação, definido por

$$C_R = -\frac{\mathbf{F}_R \cdot \mathbf{s}}{p_s \; A_r}, \tag{1.16}$$

onde $\mathbf{F}_R$ é a força de pressão de radiação solar agindo no satélite, **s** é a direção (vetor unitário) do Sol relativo ao sistema fixado ao satélite, A_r é uma área de referência, adotada como sendo o produto do diâmetro pela altura do cilindro, e p_s é a pressão de radiação solar (relação entre a constante solar – 1353 W/m^2 – e a velocidade da luz). O sinal negativo decorre do fato da força ser contrária à direção do Sol.

```
#include "stdafx.h"
void main(void)
{
	vector3 v1, v2, v3, v4, f_t;
	vector3 un_sun_vec;
	double cd, alfa, teta, dteta, tetad;
	int i;
	FILE *nfile;

	nfile = fopen("solrad.dat", "w");

	for (alfa = 0; alfa < 91.; alfa++)
	{
		printf ("%8.2f  ", alfa);
		fprintf (nfile, "%8.2f  ", alfa);
		un_sun_vec = vec3_def(cos(alfa*RADIANS), 0., sin(alfa*RADIANS));
		edm_set_solar_rad_pressure (un_sun_vec, 1.);

		for (i = 0; i < 3; i++)
		{
			if (i == 0)
				edm_set_solar_rad_surface (1., 0., 1., 330.);  // especular
			if (i == 1)
				edm_set_solar_rad_surface (0., 1., 1., 330.);  // difuso
			if (i == 2)
				edm_set_solar_rad_surface (0., 0., 1., 330.);  // corpo negro

			teta = 0;
			dteta = 2;
			tetad = teta + dteta;
			v1    = vec3_def(cos(teta*RADIANS), sin(teta*RADIANS), 0.5);
			v2    =     vec3_def(cos(teta*RADIANS), sin(teta*RADIANS), -0.5);
			f_t   = vec6_def(0., 0., 0., 0., 0., 0.);

			while (tetad < 361)
			{	// gera um cilindro. Os elementos são quadriláteros
				v3    = vec3_def(cos(tetad*RADIANS), sin(tetad*RADIANS), -0.5);
				v4    = vec3_def(cos(tetad*RADIANS), sin(tetad*RADIANS), 0.5);
				f_t   = f_t + edm_solar_prsr_quad (v1, v2, v3, v4);
				v1 = v4;
				v2 = v3;
				tetad += dteta;
			}

			cd = un_sun_vec*f_t._1/2;		// 2 é a área projetada
			printf ("%12.6f ", -cd);
			fprintf (nfile, "%12.6f ", -cd);
		}
		printf ("\n");
		fprintf (nfile, "\n");
	}
	fclose(nfile);

	return;
}
```

Figura 12.12 – Exemplo de utilização da função `edm_solar_prsr_quad`.

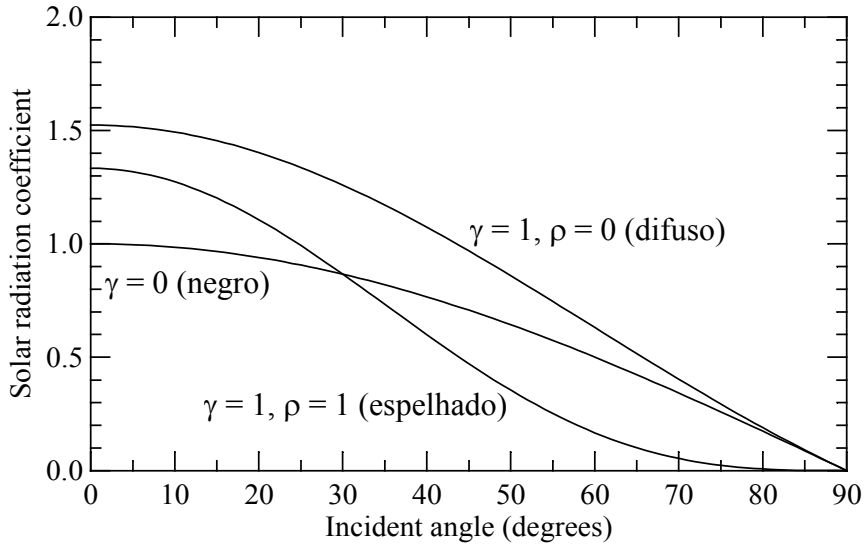

Figura 12.13 – Integração numérica do coeficiente de pressão de radiação num cilindro em função do ângulo de incidência da luz.

No próximo exemplo será investigado o comportamento do centro de pressões do satélite CBERS em função do ângulo de incidência do Sol no satélite. O ângulo de incidência foi mostrado na Figura 12.9 e o centro de pressões corresponde à posição **r** na qual o torque **T** seja gerado diretamente pelo produto vetorial desta posição pela resultante das forças, **F**. Em outras palavras, deseja-se **r** tal que:

$$\mathbf{T} = \mathbf{r} \times \mathbf{F} \,. \tag{1.17}$$

Contudo, nem sempre a força é ortogonal ao torque, como faz supor a equação. Para que o problema tenha solução, portanto, deve-se decompor o torque na componente paralela à força e na componente ortogonal a ela, como ilustra a Figura 12.14. $\hat{\mathbf{u}}$, $\hat{\mathbf{v}}$ e $\hat{\mathbf{w}}$ são versores unitários definidos de tal forma que **F** é paralelo a $\hat{\mathbf{u}}$ e o torque encontra-se no plano formado por $\hat{\mathbf{u}}$ e $\hat{\mathbf{w}}$. Decorre disso que

$$\hat{\mathbf{u}} = \frac{\mathbf{F}}{|\mathbf{F}|} \,, \tag{1.18}$$

$$\hat{\mathbf{v}} = \frac{\mathbf{T} \times \mathbf{F}}{|\mathbf{T} \times \mathbf{F}|} \,, \tag{1.19}$$

e

$$\hat{\mathbf{w}} = \hat{\mathbf{u}} \times \hat{\mathbf{v}} \tag{1.20}$$

O torque pode agora ser decomposto neste sistema de coordenadas, na forma:

$$\mathbf{T} = \mathbf{T} \cdot \hat{\mathbf{u}}\, \hat{\mathbf{u}} + \mathbf{T} \cdot \hat{\mathbf{w}}\, \hat{\mathbf{w}} \,. \tag{1.21}$$

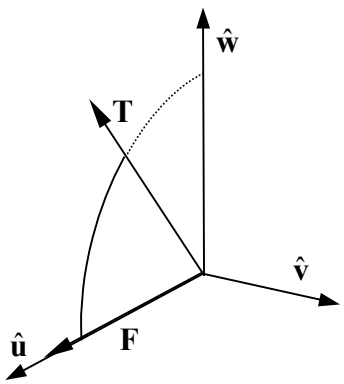

Figura 12.14 – Diagrama da resultante das forças e dos torques no satélite.

A componente $\mathbf{T}\cdot\hat{\mathbf{u}}$ do torque possui a direção da força, e produz um efeito do tipo "cata-vento" e, portanto, não pode ser gerado a partir de um produto vetorial com a força. (teria que ser necessariamente ortogonal a ela). O centro de pressões passa então a ser definido como:

$$\mathbf{T}\cdot\hat{\mathbf{w}}\;\hat{\mathbf{w}} = -\mathbf{r}\times\mathbf{F}\,. \tag{1.22}$$

Mas, uma vez que o primeiro membro é paralelo a $\hat{\mathbf{w}}$, então o segundo membro deve produzir um vetor nesta direção. Logo, $\mathbf{r}$ tem que estar no plano formado por $\hat{\mathbf{u}}$ e $\hat{\mathbf{v}}$. Seja então $\mathbf{r} = r_u\;\hat{\mathbf{u}} + r_v\;\hat{\mathbf{v}}$. Porém, a componente de $\mathbf{r}$ na direção de $\hat{\mathbf{u}}$ irá gerar um torque nulo ao ser substituída na expressão acima. Decorre disso que a outra componente é constante, e necessariamente é negativa. Pode-se então escrever que

$$|\mathbf{r}\times\mathbf{F}| = -r_v\;|\mathbf{F}| \tag{1.23}$$

de onde se tira que

$$r_v = -\frac{|\mathbf{T}\cdot\hat{\mathbf{w}}|}{|\mathbf{F}|} = -\frac{\mathbf{T}\cdot\hat{\mathbf{w}}}{|\mathbf{F}|}\,, \tag{1.24}$$

que resulta, após substituir em $\mathbf{r} = r_u\;\hat{\mathbf{u}} + r_v\;\hat{\mathbf{v}}$ e trocar o sinal do produto vetorial, em

$$\mathbf{r} = r_u\,\frac{\mathbf{F}}{|\mathbf{F}|} + \frac{\mathbf{T}\cdot\hat{\mathbf{w}}}{|\mathbf{F}|}\,\frac{\mathbf{F}\times\mathbf{T}}{|\mathbf{F}\times\mathbf{T}|}\,. \tag{1.25}$$

Nota-se que a projeção do torque na direção $\hat{\mathbf{w}}$ pode igualmente ser obtida por:

$$\mathbf{T}\cdot\hat{\mathbf{w}} = \sqrt{\mathbf{T}^2 - (\mathbf{T}\cdot\hat{\mathbf{u}})^2}\,, \tag{1.26}$$

que resulta, após a substituição do versor $\hat{\mathbf{u}}$, em

$$\mathbf{T}\cdot\hat{\mathbf{w}} = \sqrt{\mathbf{T}^2 - \frac{(\mathbf{T}\cdot\mathbf{F})^2}{\mathbf{F}^2}} = \frac{1}{|\mathbf{F}|}\sqrt{\mathbf{T}^2\mathbf{F}^2 - (\mathbf{F}\cdot\mathbf{T})^2} \tag{1.27}$$

Efetuando a substituição deste último resultado na expressão de **r**, tem-se que

$$\mathbf{r} = r_u \frac{\mathbf{F}}{|\mathbf{F}|} + \frac{\sqrt{\mathbf{F}^2\mathbf{T}^2 - (\mathbf{F}\cdot\mathbf{T})^2}}{\mathbf{F}^2} \frac{\mathbf{F}\times\mathbf{T}}{|\mathbf{F}\times\mathbf{T}|} \tag{1.28}$$

Nota-se que a primeira parcela do segundo membro é indeterminada. Pode-se, contudo, estimar o valor de r_u com base em cálculos das forças e torques sob diferentes ângulos de incidência das moléculas ou da luz solar. Se $\mathbf{r}_i$ for a posição do centro de pressão para um dado ângulo de incidência i, uma forma de se obter a melhor estimativa do centro de pressões $\mathbf{r}_{cp}$ do satélite seria por meio de

$$I_d = \min \sum_i (\mathbf{r}_{cp} - \mathbf{r}_i)^2 , \tag{1.29}$$

que irá permitir o cálculo de $\mathbf{r}_{cp}$ e de cada $r_{i,u}$ por meio de algoritmos de otimização.

O centro de pressões foi calculado para o satélite CBERS supondo-se que r_u seja nulo. Os resultados são mostrados na Figura 12.15, nas cores vermelho, verde e azul, correspondentes às componentes de **r** nos eixos x, y e z (os eixos podem ser vistos na Figura 12.9). Percebe-se que o centro de pressões está localizado sobre o eixo y e próximo ao centro do painel, que dista 5,05 m do centro do satélite. De fato, o distanciamento do centro de pressões provoca um torque elevado no satélite, que deve ser compensado pelo sistema de controle de atitude. Deve-se mencionar, ainda, que a coordenada x do centro de pressões, em vermelho, não é visualizada na figura por estar coincidindo com a coordenada z, em azul. A variação na posição do centro de pressões na direção y, em função do ângulo de incidência da luz solar, é visualizada na Figura 12.16. Percebe-se que a variação máxima é de 5% para uma excursão do ângulo de incidência de 90°, de –45° a 45°.

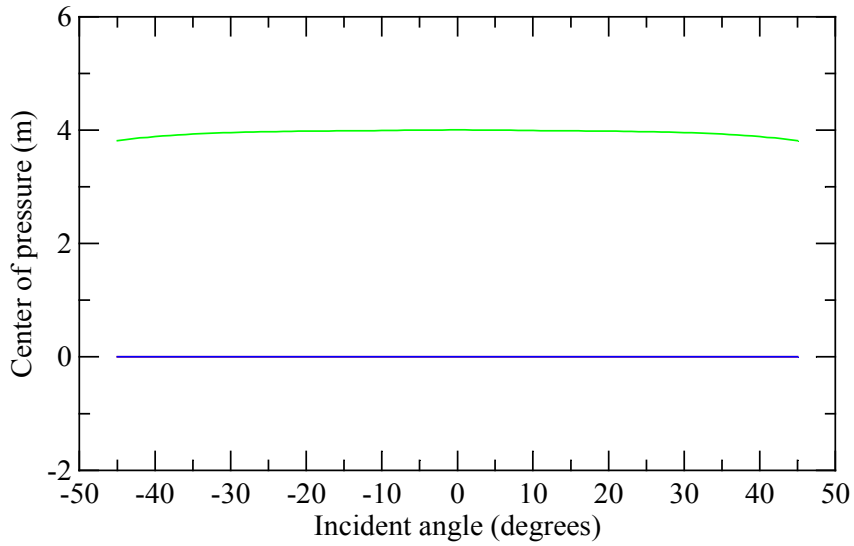

Figura 12.15 – Posição do centro de pressões no satélite CBERS em função do ângulo de incidência da luz solar.

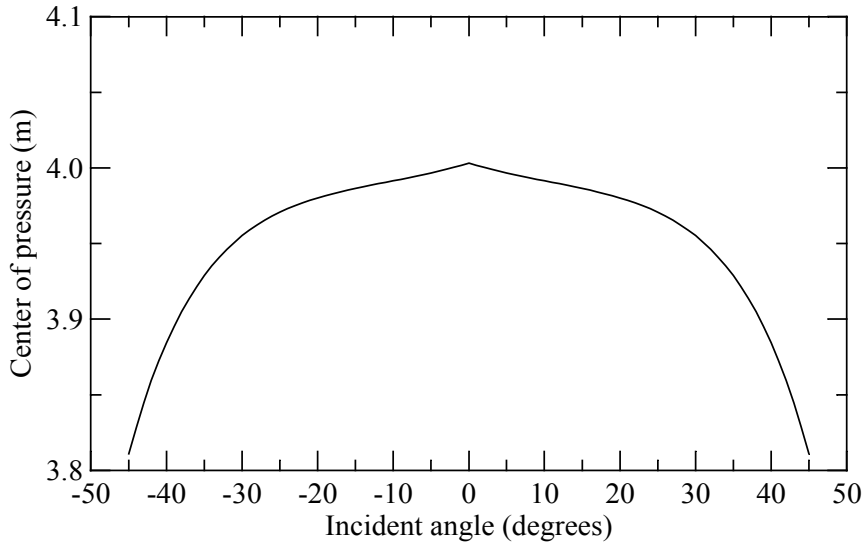

Figura 12.16 – Posição do centro de pressões ao longo do eixo *y* do satélite CBERS em função do ângulo de incidência da luz solar.

A listagem mostrada na Figura 12.17 apresenta a malha utilizada no cálculo do centro de pressão de radiação do satélite CBERS. A malha é semelhante àquela mostrada na Figura 12.11, com pequenas alterações.

```
for (alfa = -45; alfa < 46.; alfa++)
{
      printf ("%8.2f  ", alfa);
      fprintf (nfile, "%8.2f  ", alfa);

      un_sun_vec = vec3_def(cos(alfa*RADIANS), 0., -sin(alfa*RADIANS));
      edm_set_solar_rad_pressure (un_sun_vec, 1.);

      f_t = edm_solar_prsr_mesh (sat_mesh, sat_vert, sat_mat);

      u_vers = f_t._1^f_t._2;
      u_vers = u_vers/modulus(u_vers);
      r      = sqrt((f_t._1*f_t._1)*(f_t._2*f_t._2)-pow(f_t._1*f_t._2,2))/
            (f_t._1*f_t._1)*u_vers;

      printf ("%12.6f %12.6f %12.6f %12.6f ", r._1, r._2, r._3, modulus(r));
      fprintf (nfile, "%12.6f %12.6f %12.6f %12.6f ",
            r._1, r._2, r._3, modulus(r));
      printf ("\n");
      fprintf (nfile, "\n");
}
```

Figura 12.17 – Malha interna para o cálculo do centro de pressões do satélite CBERS.

12.4 Torque de gradiente de gravidade

O torque de gradiente de gravidade é analisado na Figura 12.18. Adotou-se um satélite com geometria cilíndrica tal que os momentos principais de inércia são $I_x = 10$ kg m^2, e $I_y = I_z = 100$ kg m^2. A matriz de atitude foi adotada como sendo identidade, e assim o eixo de simetria *x* do satélite coincide com o eixo *x* do sistema geocêntrico inercial. Calculou-se então o torque de gradiente de gravidade para várias posições do satélite no plano equatorial, ao longo de uma órbita circular a 500 km de altitude. A Figura 12.18 mostra o torque do eixo *z* apenas, já que as componentes nos demais eixos são nulas. A magnitude máxima mostrada no gráfico é de 0,1654 10^{-3} Nm, coincidente com o resultado teórico calculado pela expressão

$$T_{\max} = \frac{3\mu}{2R^3}(I_y - I_x), \tag{1.30}$$

onde μ é a constante geogravitacional e R é a distância geocêntrica. O código fonte para o cálculo do torque de gradiente de gravidade é apresentado na listagem da Figura 12.19.

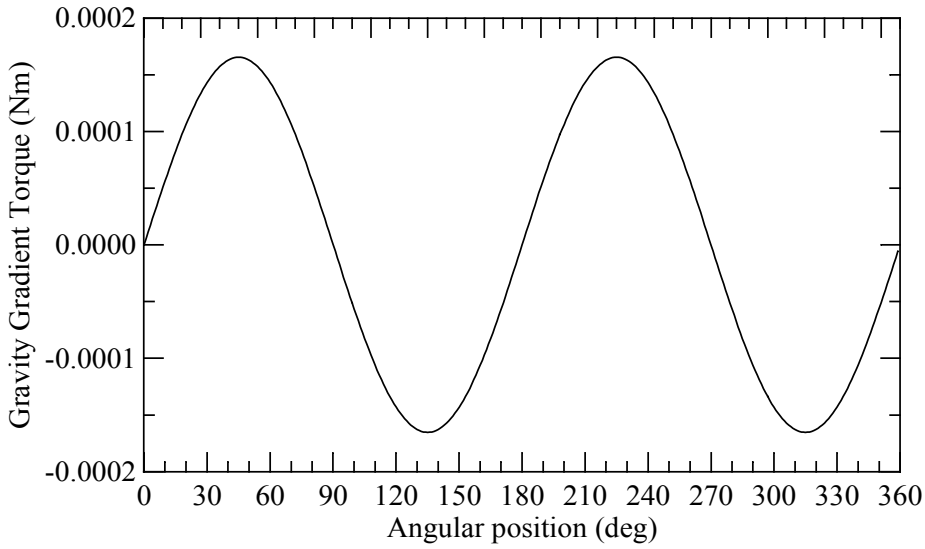

Figura 12.18 – Torque de gradiente de gravidade no eixo *z* de um satélite com distribuição de massa cilíndrica.

```
double angl, erre;
matrix3 inert, rmx_att;
vector3 tgg, sat_pos;

FILE *nfile;
nfile = fopen("gradgrav.dat", "w");

edm_set_inertia(mat3_def(10, 0, 0, 0, 100, 0, 0, 0, 100));

inert = edm_get_inertia();
rmx_att = identity (1.);

for (angl = 0; angl < 360.; angl++)
{
      printf ("%8.2f  ", angl);
      fprintf (nfile, "%8.2f  ", angl);

      erre = EARTH_RADIUS + 500000.;
      sat_pos = erre*vec3_def(cos(angl*RADIANS), sin(angl*RADIANS), 0.);
      tgg = edm_gravity_gradient(sat_pos, rmx_att);
      printf ("%12.9f %12.9f %12.9f ", tgg._1, tgg._2, tgg._3);
      fprintf (nfile, "%12.9f %12.9f %12.9f ", tgg._1, tgg._2, tgg._3);
      printf ("\n");
      fprintf (nfile, "\n");
}
fclose(nfile);
```

Figura 12.19 – Listagem do programa utilizado no cálculo do torque de gradiente de gravidade.

12.5 Torque magnético

O próximo exemplo a ser apresentado é o cálculo do torque magnético ou, por vezes, denominado de torque magnético residual, por se tratar de um torque causado por um

momento magnético não compensado durante o processo de desmagnetização do satélite. Considerou-se neste exemplo um momento magnético residual de 1 Am^2, na direção do eixo *z* do satélite, e um campo magnético terrestre com amplitude de 400 mG, localizado no plano *y*-*z* e girando ao redor do eixo *x* a partir do eixo *y*. Nesta situação o torque, mostrado na Figura 12.20, possui componente exclusivamente no eixo *x*. O código fonte deste exemplo é mostrado na listagem da Figura 12.21.

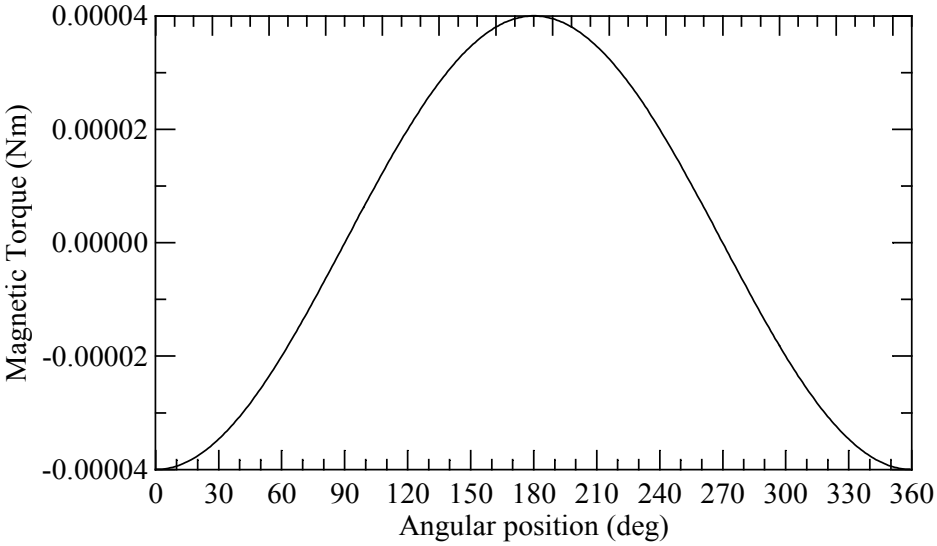

Figura 12.20 – Torque magnético num satélite de momento magnético de 1 Am^2 no eixo *z*.

```
double angl, bsat;
vector3 tm, mag_mom;

FILE *nfile;
nfile = fopen("magnetic.dat", "w");

edm_set_magnetic_moment(vec3_def(0., 0., 1.));

mag_mom = edm_get_magnetic_moment();
bsat    = 0.4e-4;        // 400 mGauss

for (angl = 0; angl < 360.; angl++)
{
      printf ("%8.2f  ", angl);
      fprintf (nfile, "%8.2f  ", angl);

      tm = edm_magnetic_torque
            (bsat*vec3_def(0., cos(angl*RADIANS), sin(angl*RADIANS)));
      printf ("%12.9f %12.9f %12.9f ", tm._1, tm._2, tm._3);
      fprintf (nfile, "%12.9f %12.9f %12.9f ", tm._1, tm._2, tm._3);
      printf ("\n");
      fprintf (nfile, "\n");
}
fclose(nfile);
```

Figura 12.21 – Código fonte do programa utilizado no cálculo do torque magnético.

12.6 Aceleração Luni-Solar

As acelerações causadas pelo Sol e Lua são mostradas nas Figuras 12.22 e 12.23. A Figura 12.24 ilustra o código fonte do programa que gerou as acelerações. Adotou-se uma órbita hélio-síncrona como exemplo, propagada analiticamente. As acelerações do Sol e Lua nesta órbita atingem magnitude de 10^{-6} m/s^2, com ligeira predominância da aceleração lunar.

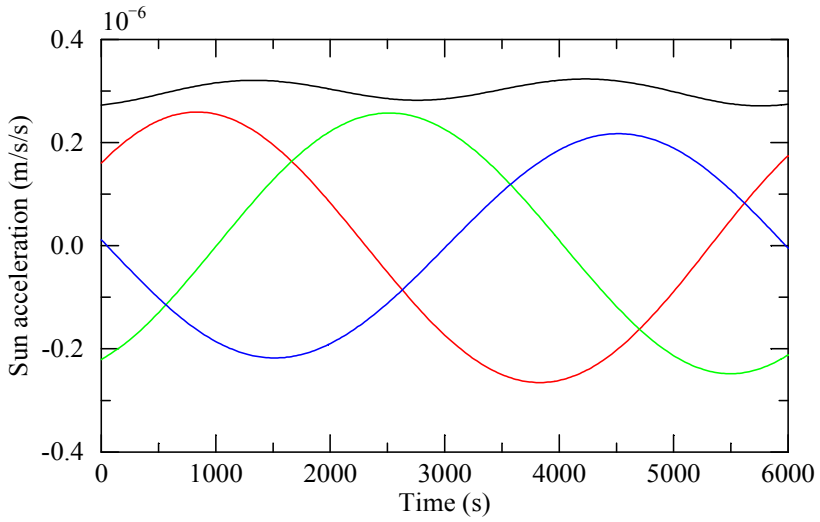

Figura 12.22 – Aceleração do Sol numa órbita polar, no sistema geocêntrico inercial, nos eixos *x* (vermelho), *y* (verde) e *z* (azul), e seu módulo (preto).

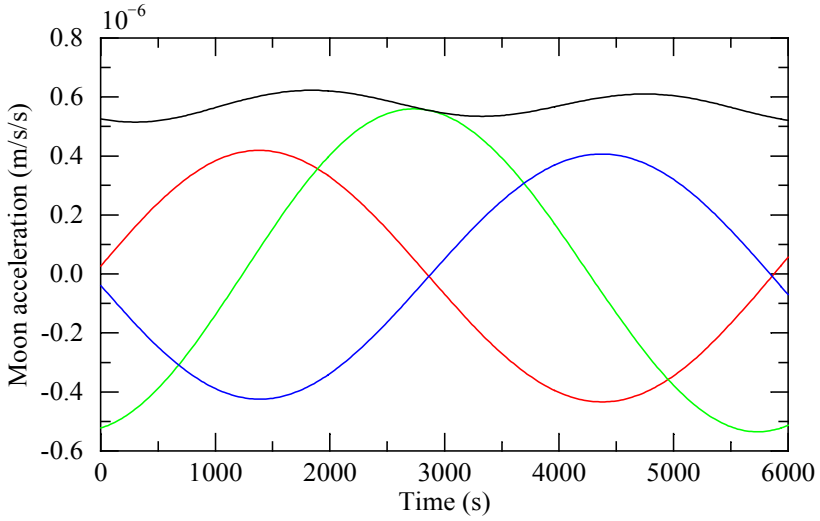

Figura 12.23 – Aceleração da Lua numa órbita polar, no sistema geocêntrico inercial, nos eixos *x* (vermelho), *y* (verde) e *z* (azul), e seu módulo (preto).

```
vector6 kepel, delke, statv, moon_iner;
vector3 sun_iner, sun_ac, moon_ac;
int diju;
double dfra, t;

FILE *nfile;
nfile = fopen("sun_moon.dat", "w");

diju = djm(20, 4, 2012);
dfra = 9.*3600. + 20.*60.;

// elementos keplerianos
kepel = vec6_def(EARTH_RADIUS + 700000., 0.02, 1.71, 2., 0., 0.);
delke = delkep(kepel);        // variação dos elementos keplerianos

for (t = 0; t <= 6000; t += 20)
{
      statv = kepel_statvec(kepel + delke*t);
      sun_iner = sun_coord(0, diju, dfra);
      sun_ac = edm_sun_accel(statv._1, sun_iner)*1000000.;
      moon_iner = moon_coord(diju, dfra);
      moon_ac = edm_moon_accel(statv._1, moon_iner._1)*1000000.;
      fprintf (nfile,
            "%8.1f %12.9f %12.9f %12.9f %12.9f %12.9f %12.9f %12.9f %12.9f ",
            t, sun_ac._1, sun_ac._2, sun_ac._3, modulus(sun_ac),
            moon_ac._1, moon_ac._2, moon_ac._3, modulus(moon_ac));
      fprintf (nfile, "\n");
}
fclose(nfile);
```

Figura 12.24 – Código fonte do programa utilizado no cálculo das acelerações do Sol e Lua.

12.7 Aceleração do geopotencial

As Figuras 12.25 e 12.26 ilustram o comportamento da aceleração devido ao geopotencial terrestre ao longo de uma órbita hélio-síncrona, cujos elementos keplerianos podem ser vistos na variável `kepel` da listagem da Figura 12.27. Admitiu-se que o sistema geocêntrico terrestre coincida no instante considerado com o sistema geocêntrico inercial (com Terra parada no intervalo de propagação da órbita). A Figura 12.25 mostra a aceleração gravitacional no sistema geocêntrico inercial, enquanto que a Figura 12.26 mostra a mesma aceleração no sistema orbital. Neste último sistema de coordenadas a aceleração está alinhada com o sentido negativo do eixo x (vermelho). As demais componentes são quase nulas (da ordem de 10^{-2} m/s^2).

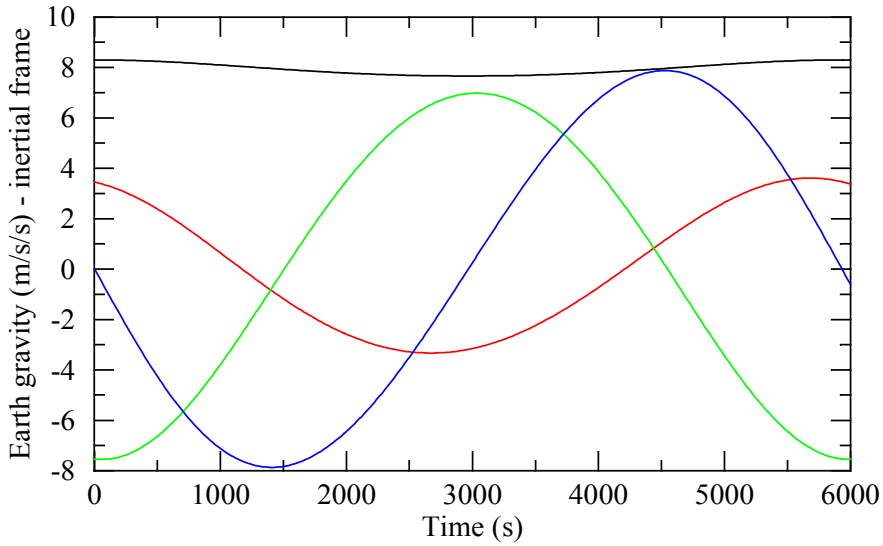

Figura 12.25 – Aceleração devida ao geopotencial terrestre numa órbita polar, no sistema geocêntrico inercial, nos eixos *x* (vermelho), *y* (verde) e *z* (azul), e seu módulo (preto).

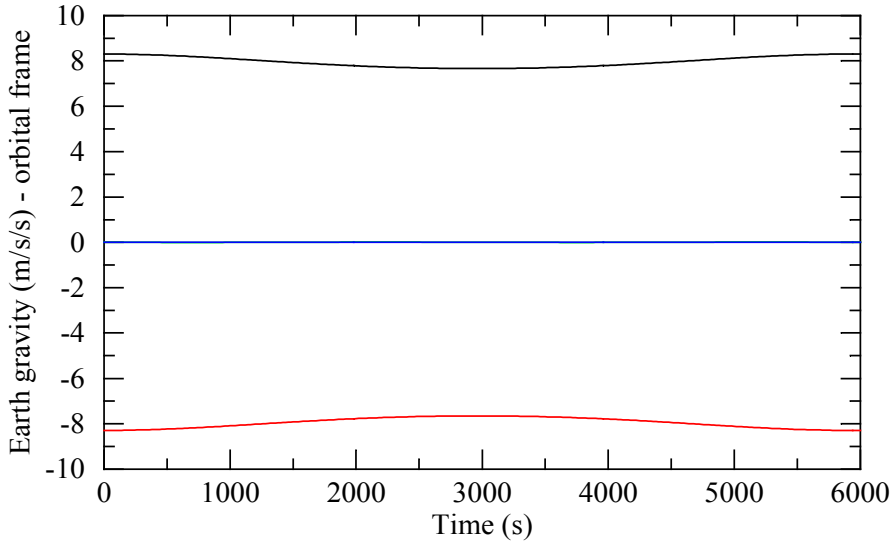

Figura 12.26 – Aceleração devida ao geopotencial terrestre numa órbita polar, no sistema orbital inercial, nos eixos *x* (vermelho), *y* (verde) e *z* (azul – sobre o verde), e seu módulo (preto).

```
vector6 kepel, delke, statv;
vector3 acel;

int diju;
double dfra, t;

FILE *nfile;
nfile = fopen("leg_out.dat", "w");

leg_initialize("egm96.dat", 360);

diju = djm(20, 4, 2012);
dfra = 9.*3600. + 20.*60.;

// elementos keplerianos
kepel = vec6_def(EARTH_RADIUS + 700000., 0.02, 1.71, 2., 0., 0.);
delke = delkep(kepel);        // variação dos elementos keplerianos

for (t = 0; t <= 6000; t += 20)
{
      statv = kepel_statvec(kepel + delke*t);
      acel = leg_forcol_ac (360, statv._1);

      fprintf (nfile,
            "%8.1f %12.9f %12.9f %12.9f ",
            t, acel._1, acel._2, acel._3);
      acel = transpose(orbital_to_inertial_sv (statv))*acel;
      fprintf (nfile,
            "%12.9f %12.9f %12.9f %12.9f ",
            acel._1, acel._2, acel._3, modulus(acel));
      fprintf (nfile, "\n");
}
fclose(nfile);
```

Figura 12.27 – Código fonte do programa utilizado no cálculo das acelerações do Sol e Lua.

BIBLIOGRAFIA

CARRARA, V. **Implementações de modelos atmosféricos para uso em propagadores de órbita e atitude**. S. J. Campos, INPE, maio 1990 (INPE-5094-RPI/231).

CARRARA, V. **Modelagem das forças e torques atuantes em satélites**. São José dos Campos, INPE, 1982. (INPE 2454 TDL/094).

CARRARA, V. **A Program to Compute the Aerodynamic or Solar Radiation Forces and Torques on Satellites**. Ste. Anne du Bellevue, SPAR, March 1988 (SPAR-RML-009-87-11).

CARRARA, V. **Pacote computacional para simulação de atitude de satélites**. Disponível em http://www2.dem.inpe.br/val/projetos/att_pro/manual_simulador.pdf, 2007.

CARRARA, V. Attitude Simulation Software to Support Brazilian Space Missions. 22nd International Symposium on Space Flight Dynamics. **Proceedings**. São José dos Campos, Feb-March 2011.

CARRARA, V. **A (huge) collection of interchangeable empirical models to compute the thermosphere density, temperature and composition**. Disponível em http://www2.dem.inpe.br/val/atmod/default.html, 2012.

CARRARA, V. **Modelos de perturbações em satélites**. São José dos Campos, INPE, 2013. (a ser publicado).

EGM. **Earth Gravitational Model 1996**, Disponível em http://cddis.nasa.gov/926/egm96/egm96.html, 2012.

NASA. **Spacecraft Radiation Torques**, NASA Space Vehicle Design Criteria (Guidance and Control). NASA SP-8027, 1969.

MOREAU, J-P. **Bessel Programs in C/C++**. Disponível em http://jean-pierre.moreau.pagesperso-orange.fr/c_bessel.html, 2012.

POVRAY. **The Persistence of Vision Raytracer**. Disponível em http://www.povray.org/, 2013.

SINGER, S. F. **Torques and attitude sensing in Earth satellites**. Academic Press, 1964

WERTZ, J. R. **Spacecraft attitude determination and control**. D. Reidel Publishing, 1978.

APÊNDICE A

O objetivo deste apêndice é descrever o roteiro necessário para visualizar a geometria e as características superficiais do satélite fornecidas por meio do arquivo que a descreve (veja-se o Capítulo 6). Inicialmente deve-se efetuar a descarga e a instalação do programa POVRay, obtido gratuitamente no sítio www.povray.org. Em seguida deve-se criar um arquivo ASCII contendo o "script" de visualização, mostrado na listagem a seguir, com extensão `pov`. O nome do arquivo é irrelevante, e será adotado aqui o nome `mesh.pov`. Este arquivo deve ser armazenado em algum diretório junto com o arquivo `mesh2pov.dat` gerado pela função `edm_mesh2pov`. Executa-se, a seguir, o programa POV e, a partir do menu File, deve-se selecionar a opção Open file... e direcionar a busca ao diretório que contém o arquivo `mesh.pov`. A linguagem utilizada pelo POV é bastante intuitiva e semelhante à linguagem C. Uma rápida leitura deste "script" irá revelar que algumas variáveis podem ser ajustadas para evidenciar uma característica ou outra. Por exemplo, o comando `camera` permite configurar a câmera, e para isso foram criadas as variáveis `loc_cam`, para posicioná-la, `sky_cam` para informar qual é o eixo associado à direção vertical da imagem, `fov_cam` para aumentar ou reduzir o FOV, e `look_cam` para informar a posição para a qual a câmera aponta.

Para sintetizar a imagem da geometria basta selecionar o botão "run". A variável `vis_sel` contida no script seleciona a característica da superfície a ser visualizada. A Tabela A.1 mostra as diversas visualizações implementadas.

Tabela A.1 – Visualizações possíveis com o script do programa em POV, na variável `vis_sel`.

Seletor	Ação
0	Face selecionada para o cálculo (selecionada = verde, não selecionada = vermelha)
1	Sistema de referência: (verde: corpo principal, gradações de cinza (1 a 8): apêndices)
2	Índice de seletor de materiais em tons de amarelo 0 = amarelo claro, 10 = preto
3	Valor do coeficiente de troca de quantidade de movimento na direção normal (em tons de vermelho)
4	Valor do coeficiente de troca de quantidade de movimento na direção tangencial (em tons de azul)
5	Valor da temperatura da superfície do azul (mais frio), para o vermelho (mais quente) (ver `tpmax` e `tpmin`)
6	Valor do coeficiente de reflexão especular (em tons de ciano)
7	Valor do coeficiente de reflexão difusa (em tons de amarelo)
8	Valor da emissividade superficial (em tons de magenta)
9	Imagem gerada com reflexão difusa e especular (em tons de amarelo)

De particular interesse é a visualização de número 0 (`vis_sel = 0`), que permite verificar se a geometria e a seleção de faces a serem utilizadas no cálculo estão corretas. As faces selecionadas são mostradas em verde, enquanto aquelas não selecionadas são vermelhas. No caso de um satélite, nenhuma face externa deverá ser vermelha. Recomenda-se visualizar o satélite sob várias direções, para garantir que todas as superfícies e faces possam ser identificadas. A Figura A.1 mostra a execução do arquivo mostrado na Figura 6.6, que contém a geometria do satélite CBERS. Nota-se que os eixos cartesianos são automaticamente gerados, tal que x é vermelho, y é verde e z é azul. A Figura A.2 mostra um exemplo no qual a temperatura é associada a uma dada cor, do azul (mais frio), para o vermelho (mais quente).

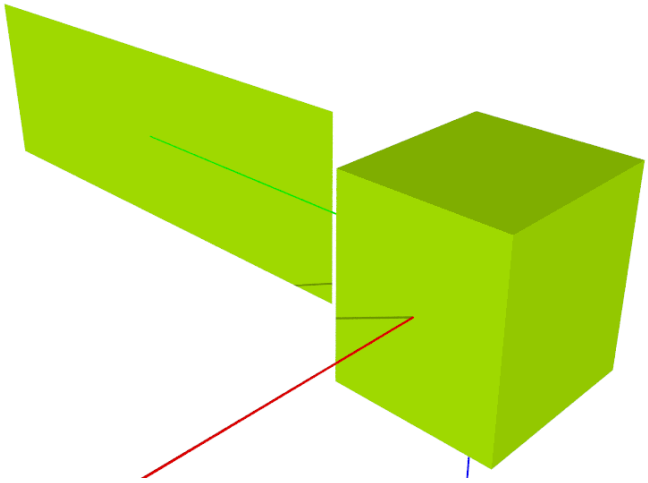

Fig. A.1 – Geometria do satélite CBERS, gerado pelo "script" de POV.

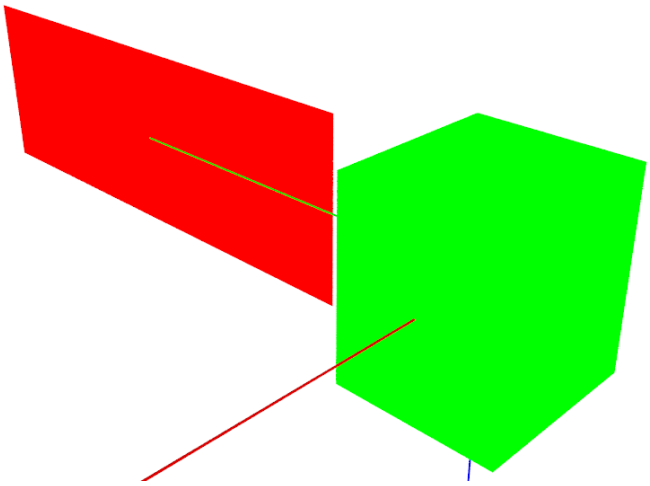

Fig. A.2 – Distribuição de temperaturas nas faces do satélite CBERS.

Apresenta-se a seguir a listagem do "script" em POV para visualização da geometria.

```
//
//
// Programa para sintetizar a imagem de um satélite descrito por um arquivo com a tabela
// de vértices
//
//
// O arquivo deve ser gerado pela função edm_mesh2pov, que faz parte do módulo
// de perturbações ambientais
//
//

background { color <1.00,1.000,1.000> }
#include "colors.inc"

// posição da câmera no sistema de coordenadas do satélite:
#declare loc_cam = <3, -2, 1>*2.2;

// definição do eixo vertical:
#declare sky_cam = -z;   // obs: se o eixo vertical for y, então o vetor up
// deverá ser definido como -y para que o sistema seja dextrógiro
```

```
// ângulo de abertura da câmera (FOV) em graus
#declare fov_cam = 40;

// posição para a qual o centro da imagem aponta (normalmente a origem do sistema
// de coordenadas)
#declare look_cam = <0, 2.0, .0>;

// posição da fonte de luz (pode ser incluído mais de uma fonte de luz)
#declare pos_light = (loc_cam + <1, -1, 2>)*1000;

// tipo de visualização:
// 0: face selecionada para o cálculo (selecionada = verde, não selecionada = vermelha)
// 1: sistema de referência: (verde: corpo principal, gradações de cinza (1 a 8):
//    apêndices)
// 2: índice de seletor de materiais em tons de amarelo; 0 = amarelo claro, 10 = preto
// 3: valor do coeficiente de troca de quantidade de movimento na direção normal (em tons
//    de vermelho)
// 4: valor do coeficiente de troca de quantidade de movimento na direção tangencial (em
//    tons de azul)
// 5: valor da temperatura da superfície, do azul (mais frio) para o vermelho (mais quente)
//    (ver tpmax e tpmin)
// 6: valor do coeficiente de reflexão especular (em tons de ciano)
// 7: valor do coeficiente de reflexão difusa (em tons de amarelo)
// 8: valor da emissividade superficial (em tons de magenta)
// 9: imagem gerada com reflexão difusa e especular (em tons de amarelo)

#declare vis_sel = 0;
#declare tpmin   = 320;     // menor temperatura em kelvin encontrada no satélite – usada
// para ajustar a cor azul
#declare tpmax   = 380;     // maior temperatura em kelvin encontrada no satélite - usada
// para ajustar a cor vermelha

// definição da câmera:

camera {  //  Camera Camera01
  location  loc_cam
  sky       sky_cam
  up        sky_cam
  angle     fov_cam
  look_at   look_cam
}

// definição das luzes:

light_source {  <0.0, 0.0, 0.0>
  color rgb <1.000, 1.000, 1.000>
  translate  pos_light
}

// materiais:

// textura da face usada para o cálculo (verde):
#declare mte1 = texture { pigment{ color rgb< 0.5, 1.0, 0.0> }
finish { diffuse 0.5 ambient 0.3 } }

// textura da face não usada no cálculo (vermelha)
#declare mte2 = texture { pigment{ color rgb< 1., .0, 0.0> } finish { ambient 1 } }

// descrição da cena:

// separação entre superfície externa e interna
#declare delta = 0.0001;

// abrir arquivo com a geometria do satélite:
#fopen MecFile "mesh2pov.dat" read

#declare icount = 0;

// malha de leitura e geração da imagem:
#while (defined(MecFile))
        #read(MecFile, v1, v2, v3, vsf, s_n, s_t, tmp, esp, dif, emt)
        #declare icount = icount + 1;

        // imagem com a seleção de faces
        #if (vis_sel = 0)
            // flag de seleção da face (0=ambas, 1=normal, 2=contraria a normal)
            #declare fs = vsf.y;
```

```
            // vetor normal ao triângulo, direção da face externa:
            #declare nm = vcross(v2-v1, v3-v1)*delta;

            triangle{  v1, v2, v3 // face da direção da normal
                #if (fs = 1 | fs = 0) texture {mte1}
                #else texture {mte2}
                #end
            }
            triangle{  v1-nm, v2-nm, v3-nm  // face contrária à direção da normal
                #if (fs = 2 | fs = 0) texture {mte1}
                #else texture {mte2}
                #end
            } //
        #end
        // imagem com a seleção do sistema de referência
        #if (vis_sel = 1)
            #declare fs = vsf.x;    // seletor de sistema de referência

            #if (fs = 0)
                triangle{  v1, v2, v3 texture {mte1} }
            #else
                triangle{  v1, v2, v3
                    texture { pigment{ color rgb < 1.0, 1.0, 1.0>*(1-fs/8) }
finish { diffuse 0.6 ambient 0.3 } }
                }
            #end
        #end
        // imagem com a seleção do índice de material
        #if (vis_sel = 2)
            #declare fs = vsf.z;    // índice do material
                triangle{  v1, v2, v3
                    texture { pigment{ color rgb < 1.0, 1.0, 0.0>*(1-fs/10) }
finish { ambient 1 diffuse 0 } }
                }
        #end
        // imagem com o coeficiente sigma_normal
        #if (vis_sel = 3)
                triangle{  v1, v2, v3
                    texture { pigment{ color rgb < 1.0, 0.0, 0.0>*s_n }
finish {  ambient 1 diffuse 0 } }
                }
        #end
        // imagem com o coeficiente sigma_tangencial
        #if (vis_sel = 4)
                triangle{  v1, v2, v3
                    texture { pigment{ color rgb < 0.0, 0.0, 1.0>*s_t }
finish {  ambient 1 diffuse 0 } }
                }
        #end
        // imagem com a temperatura da superfície
        #if (vis_sel = 5)
            #declare cor = CHSV2RGB(<(tpmax-tmp)/(tpmax-tpmin)*240, 1, 1>);
                triangle{  v1, v2, v3
                    texture { pigment{ color cor } finish {  ambient 1 diffuse 0 } }
                }
        #end
        // imagem com o coeficiente de reflexão especular
        #if (vis_sel = 6)
                triangle{  v1, v2, v3
                    texture { pigment{ color rgb < 0.0, 1.0, 1.0>*esp }
finish {  ambient 1 diffuse 0 } }
                }
        #end
        // imagem com o coeficiente de reflexão difusa
        #if (vis_sel = 7)
                triangle{  v1, v2, v3
                    texture { pigment{ color rgb < 1.0, 1.0, 0.0>*dif }
finish {  ambient 1 diffuse 0 } }
                }
        #end
        // imagem com a emissividade superficial
        #if (vis_sel = 8)
                triangle{  v1, v2, v3
                    texture { pigment{ color rgb < 1.0, 0.0, 1.0>*emt }
finish {  ambient 1 diffuse 0 } }
                }
```

```
        #end
        // imagem com reflexão difusa e especular
        #if (vis_sel = 9)
                triangle{  v1, v2, v3
                    texture { pigment{ color rgb < 1.0, 1.0, 0.0> }
finish { reflection {0.05, esp} ambient 0.2 diffuse dif-0.2 } }
                }

        #end

#end

#if (vis_sel = 9)
    //-------------------------------------------------------------------------
    sky_sphere { pigment { gradient <0,0,1>
                        color_map { [0.00 rgb <1.0,1.0,1.0>]
                                    [0.30 rgb <0.2,0.6,1.0>]
                                    [0.70 rgb <0.2,0.6,1.0>]
                                    [1.00 rgb <1.0,1.0,1.0>]
                                  }

                        scale 1
                      } // end of pigment
            } //end of skysphere
    //-------------------------------------------------------------------------
    #macro Axis_( AxisLen, Dark_Texture,Light_Texture)
        union{
            cylinder { <0,-AxisLen,0>,<0,AxisLen,0>,0.05
               texture{checker texture{Dark_Texture }
                               texture{Light_Texture}
                        translate<0.1,0,0.1>}
             }
        cone{<0,AxisLen,0>,0.2,<0,AxisLen+0.7,0>,0
              texture{Dark_Texture}
             }
         } // end of union
    #end // of macro "Axis()"
    //-------------------------------------------------------------------------
    #macro AxisXYZ( AxisLenX, AxisLenY, AxisLenZ, Tex_Dark, Tex_Light)
    //--------------------- drawing of 3 Axes --------------------------------
        union{
            #if (AxisLenX != 0)
                object { Axis_(AxisLenX, Tex_Dark, Tex_Light)
rotate< 0,0,-90> rotate 90*x}// x-Axis
                text   { ttf "arial.ttf",  "x",  0.15,  0  texture{Tex_Dark}
                rotate<20,-45,0> scale 0.75 translate <AxisLenX+0.05,0.4,-0.10>
rotate 90*x no_shadow}
            #end // of #if
            #if (AxisLenY != 0)
                object { Axis_(AxisLenY, Tex_Dark, Tex_Light)   rotate< 0,0,  0>
rotate 90*x}// y-Axis
                text   { ttf "arial.ttf",  "y",  0.15,  0  texture{Tex_Dark}
                rotate<10,0,0> scale 0.75 translate <-0.65,AxisLenY+0.50,-0.10>
rotate<0,-45,0> rotate 90*x no_shadow}
            #end // of #if
            #if (AxisLenZ != 0)
                object { Axis_(AxisLenZ, Tex_Dark, Tex_Light)   rotate<90,0,  0>
rotate 90*x}// z-Axis
                text   { ttf "arial.ttf",  "z",  0.15,  0  texture{Tex_Dark}
                rotate<20,-45,0> scale 0.85 translate <-0.75,0.2,AxisLenZ+0.10>
rotate 90*x no_shadow}
            #end // of #if
        } // end of union
    #end// of macro "AxisXYZ( ... )"
    //-------------------------------------------------------------------------

    #declare Texture_A_Dark  = texture {
                               pigment{ color rgb<1,0.45,0>}
                               finish { phong 1}
                             }
    #declare Texture_A_Light = texture {
                               pigment{ color rgb<1,1,1>}
                               finish { phong 1}
                             }

    // object{ AxisXYZ( 4.50, 3.00, 5.00, Texture_A_Dark, Texture_A_Light) rotate 90*x}
    //------------------------------------------------- end of coordinate axes
```

```
    // ground ------------------------------------------------------------------
    //---------------------------------<<< settings of squared plane dimensions
    #declare RasterScale = 1.0;
    #declare RasterHalfLine  = 0.035;
    #declare RasterHalfLineZ = 0.035;
    //-------------------------------------------------------------------------
    #macro Raster(RScale, HLine)
       pigment{ gradient x scale RScale
                color_map{[0.000   color rgbt<1,1,1,0>*0.6]
                          [0+HLine color rgbt<1,1,1,0>*0.6]
                          [0+HLine color rgbt<1,1,1,1>]
                          [1-HLine color rgbt<1,1,1,1>]
                          [1-HLine color rgbt<1,1,1,0>*0.6]
                          [1.000   color rgbt<1,1,1,0>*0.6]} }
    #end// of Raster(RScale, HLine)-macro
    //-------------------------------------------------------------------------


    plane { <0,1,0>, 0    // plane with layered textures
        texture { pigment{color White*1.1}
                  finish {ambient 0.45 diffuse 0.85}}
        texture { Raster(RasterScale,RasterHalfLine ) rotate<0,0,0> }
        texture { Raster(RasterScale,RasterHalfLineZ) rotate<0,90,0> }
        rotate<0,0,0>  rotate 90*x translate 10*z
      }
    //------------------------------------------------ end of squared plane XZ
#else
    // eixos: x = vermelho, y = verde, z = azul
    cylinder { <0,0,0>,<5,0,0>, 0.01
           texture { pigment { color rgb<1,0,0>}
                     finish  { ambient 0.4 diffuse 0.6 }
                   }
             }
    cylinder { <0,0,0>,<0,5,0>, 0.01
           texture { pigment { color rgb<0,1,0>}
                     finish  { ambient 0.4 diffuse 0.6 }
                   }
             }
    cylinder { <0,0,0>,<0,0,5>, 0.01
           texture { pigment { color rgb<0,0,1>}
                     finish  { ambient 0.4 diffuse 0.6 }
                   }
             }
#end
```

APÊNDICE B

Este apêndice apresenta as funções da biblioteca EDM, bem como as estruturas que armazenam a geometria.

Estruturas para descrição da geometria do satélite.

- `vertex sat_vert;`
- `material sat_mat;`
- `mesh sat_mesh;`

Funções gerais e auxiliares.

- `void edm_set_surface_side (unsigned char surfaceside);`
- `unsigned char edm_get_surface_side();`
- `double BESSI0 (double X);`
- `double BESSI1 (double X);`
- `double BESSI (int N, double X);`
- `double edm_erf(double x);`

Funções para alocação de memória das estruturas.

- `vertex alloc_vertex(int size);`
- `void dealloc_vertex(vertex vert);`
- `material alloc_material(int size);`
- `void dealloc_material(material mat);`
- `mesh alloc_mesh(int size);`
- `void dealloc_mesh(mesh s_mesh);`

Funções para leitura do arquivo com a descrição da geometria

- `vertex edm_read_vertex (FILE* nfil);`
- `material edm_read_material (FILE* nfil);`
- `mesh edm_read_mesh (FILE* nfil, vertex sur_vert, material sur_mat);`
- `void edm_print_mesh (char *name, vertex sur_grid, mesh sur_mesh, material sur_mat);`

Funções para a visualização em POV

- `void edm_mesh2pov (FILE *nfile, vertex sur_grid, mesh sur_mesh, material sur_mat);`
- `void edm_triangle2pov (FILE *nf, vector3 vertex_1, vector3 vertex_2, vector3 vertex_3, int sys_pt);`
- `void edm_quad2pov (FILE *nf, vector3 vertex_1, vector3 vertex_2, vector3 vertex_3, vector3 vertex_4, int sys_pt);`

Funções para atribuição de características aerodinâmicas.

- `void edm_set_drag_surface (double sigma_normal, double sigma_tang, double sq_temp_ratio);`
- `vector3 edm_get_drag_surface();`

- `void edm_set_environ_properties (vector3 sat_velocity, double atmo_density, double atmo_temperature, double mean_molecular_weight);`
- `void edm_set_drag_properties (vector3 un_sat_vel, double aerodynamic_pressure, double speed_ratio);`
- `vector6 edm_get_drag_properties();`

Funções para cálculo aerodinâmico de malhas e estruturas.

- `vector3 edm_aerodynamic_plane (vector3 un_normal, double area);`
- `vector6 edm_aerodynamic_plane_fb (vector3 un_normal, double length, double area);`
- `vector6 edm_aerodynamic_triangle (vector3 vertex_1, vector3 vertex_2, vector3 vertex_3);`
- `vector6 edm_aerodynamic_quad (vector3 vertex_1, vector3 vertex_2, vector3 vertex_3, vector3 vertex_4);`
- `vector6 edm_aerodynamic_mesh (mesh s_mesh, vertex s_vert, material s_mat);`

Funções para cálculo aerodinâmico de geometrias definidas.

- `vector3 edm_aerodynamic_sphere (double radius);`
- `vector6 edm_aerodynamic_sphere_sector (double radius, vector3 ang_meridian, vector3 ang_paralel);`
- `vector6 edm_aerodynamic_cylinder (double radius, vector3 center_bottom, vector3 center_top);`
- `vector6 edm_aerodynamic_cone (double radius_bottom, vector3 center_bottom, double radius_top, vector3 center_top, int n_div);`
- `vector6 edm_aerodynamic_box (vector3 bottom_corner, vector3 top_corner);`

Funções para atribuição de características de pressão de radiação.

- `void edm_set_solar_rad_surface (double specular_coef, double diffuse_coef, double emmitance, double temperature);`
- `vector3 edm_get_solar_rad_surface ();`
- `double edm_get_surface_temperature ();`
- `void edm_set_solar_rad_properties (vector3 sun_vector);`
- `void edm_set_solar_rad_pressure (vector3 un_sun_vec, double rad_prsr);`
- `vector6 edm_get_solar_rad_properties ();`

Funções para cálculo de pressão de malhas e estruturas.

- `vector3 edm_solar_prsr_plane (vector3 un_normal, double area);`
- `vector6 edm_solar_prsr_triangle (vector3 vertex_1, vector3 vertex_2, vector3 vertex_3);`
- `vector6 edm_solar_prsr_quad (vector3 vertex_1, vector3 vertex_2, vector3 vertex_3, vector3 vertex_4);`
- `vector6 edm_solar_prsr_mesh (mesh s_mesh, vertex s_vert, material s_mat);`

Funções para cálculo de pressão de radiação em geometrias definidas.

- `vector3 edm_solar_prsr_sphere (double radius);`

- `vector6 edm_solar_prsr_sphere_sector (double radius, vector3 ang_meridian, vector3 ang_paralel);`
- `vector6 edm_solar_prsr_cylinder (double radius, vector3 center_bottom, vector3 center_top);`
- `vector6 edm_solar_prsr_cone (double radius_bottom, vector3 center_bottom, double radius_top, vector3 center_top);`
- `vector6 edm_solar_prsr_box (vector3 bottom_corner, vector3 top_corner);`

Funções para cálculo do torque de gradiente de gravidade

- `void edm_set_inertia (matrix3 inertia);`
- `matrix3 edm_get_inertia ();`
- `vector3 edm_gravity_gradient (vector3 sat_pos, matrix3 rmx_att);`

Funções para cálculo do torque de magnético residual

- `void edm_set_magnetic_moment (vector3 mag_moment);`
- `vector3 edm_get_magnetic_moment ();`
- `vector3 edm_magnetic_torque (vector3 b_sat_coord);`

Funções para cálculo da aceleração luni-solar

- `vector3 edm_moon_accel (vector3 sat_iner, vector3 moon_iner);`
- `vector3 edm_sun_accel (vector3 sat_iner, vector3 sun_iner);`

Funções para cálculo da aceleração do potencial gravitacional terrestre

- `void leg_initialize(char *name, int nmax);`
- `vector3 leg_forcol_ac (int nm, vector3 x);`