Universidade de Brasília

Faculdade de Ciências e Tecnologias em Engenharia

Determinação de atitude para o CubeSat SPLASH utilizando magnetômetro e sensor solar

Antonio Lucas Suzuk Aguiar

TRABALHO DE CONCLUSÃO DE CURSO

ENGENHARIA AEROESPACIAL

Brasília

2025

Universidade de Brasília
Faculdade de Ciências e Tecnologias em Engenharia

Determinação de atitude para o CubeSat SPLASH utilizando magnetômetro e sensor solar

Antonio Lucas Suzuk Aguiar

Trabalho de Conclusão de Curso submetido como requisito parcial para obtenção do título de Bacharel em Engenharia Aeroespacial.

Orientador: Prof. Dr. William Reis Silva

Brasília
2025

FICHA CATALOGRÁFICA

Aguiar, Antonio Lucas Suzuk.

Determinação de atitude para o CubeSat SPLASH utilizando magnetômetro e sensor solar / Antonio Lucas Suzuk Aguiar; orientador William Reis Silva. -- Brasília, 2025.

119 p.

Trabalho de Conclusão de Curso (Engenharia Aeroespacial) -- Universidade de Brasília, 2025.

1. Atitude. 2. CubeSat. 3. Kalman. 4. Estimação. I. Silva, William Reis, orient. II. Título.

Universidade de Brasília
Faculdade de Ciências e Tecnologias em Engenharia

Determinação de atitude para o CubeSat SPLASH utilizando magnetômetro e sensor solar

Antonio Lucas Suzuk Aguiar

Trabalho de Conclusão de Curso submetido como requisito parcial para obtenção do título de Bacharel em Engenharia Aeroespacial.

Trabalho aprovado. Brasília, 02 de Dezembro de 2025:

Prof. Dr. William Reis Silva,
UnB/FCTE
Orientador

Prof. Dr. Thiago Felippe Kurudez Cordeiro,
UnB/FCTE
Examinador Interno

Prof. Dr. Renato Alves Borges,
UnB/FT
Examinador Interno

Agradecimentos

Aos meus pais e à minha irmã, que me apoiaram na minha escolha de cursar Engenharia Aeroespacial em Brasília. E que, apesar de todo o tempo que levei, sempre me apoiaram integralmente e não mediram esforços para me proporcionar a oportunidade de concluir o curso.

Aos vários amigos que fiz durante toda essa jornada, que me apoiaram e me ajudaram sempre que possível. Em especial meus grandes amigos Mateus e Wolfgang, que sempre me encorajaram nos momentos de incerteza e dúvida e sempre estiveram presentes e dispostos a ajudar com quaisquer que fossem as dificuldades que aparecessem no meu caminho.

Ao professor William Reis Silva, por ter aceitado ser meu orientador e ter me apoiado durante todo o curso e tê-lo feito com tanta dedicação e amizade. Ao professor Manuel Nascimento Dias Barcelos Júnior por sempre ter estado disposto a conversar comigo e trocar ideias quando me encontrava incerto sobre que caminhos ou decisões tomar. Ao professor Artur Elias de Morais Bertoldi, por toda a ajuda que me ofereceu durante toda a graduação, principalmente como Coordenador do curso de Engenharia Aeroespacial. Ao professor Sébastien Roland Marie Joseph Rondineau pela amizade e disposição de ajudar com qualquer que fosse a situação.

“Você tem poder sobre sua mente — não sobre os eventos externos.
Perceba isso e você encontrará força.”
(Marcus Aurelius)

Resumo

O controle e estimação de atitude de um satélite são aspectos importantes para o sucesso de uma missão espacial, pois ajudam a manter uma orientação e posicionamento precisos do satélite no espaço. Este trabalho tem por objetivo fazer uma comparação entre alguns dos tipos de Filtros de Kalman utilizados para esse fim, além de fazer uma comparação com filtros mais simples como o filtro Alfa e o Quaternion Estimator (QUEST) : o Filtro de Kalman Extendido (*Extended Kalman Filter* – EKF) , o Filtro de Kalman Adaptativo (*Adaptive Kalman Filter* – AKF) e o Filtro de Kalman Ensemble (*Ensemble Kalman Filter*). Alguns outros filtros também serão estudados afim de fazer uma comparação menos rigorosa com alguns dos filtros estudados. Satélites operam em um dos ambientes mais dinâmicos e imprevisíveis que é o espaço. A dinâmica não-linear e as incertezas inerentes ao movimento de um satélite fazem com que seja necessário técnicas sofisticadas de filtragem com a finalidade de viabilizar uma estimação e controle de atitude eficientes, que são propriedades determinantes para o sucesso de uma missão. Utilizando uma abordagem baseada em simulações, este trabalho busca avaliar a performance, precisão e eficiência computacional de cada uma destas variações do Filtro de Kalman sob as mesmas condições dinâmicas e cenários orbitais. Para este fim será utilizado como objeto de estudo o satélite conceitual *Self-DePloyable FLexible AeroSHell for de-Orbiting and Space Re-entry* (SPLASH) assim como o cenário para o qual o mesmo está sendo idealizado. E para a simulação de fato será utilizado o software de computação numérica MATLAB . Os dados utilizados para a simulação virão do magnetômetro, giroscópio e sensor solar, que terão suas características simuladas em código, com a introdução de ruídos e erros, com o propósito de gerar dados mais próximos a realidade.

Palavras-chave: Atitude. CubeSat. Kalman. Estimação.

Abstract

The estimation and control of a satellite's attitude are one of the most important aspects for a successful space mission on account that they're responsible for keeping the satellite's positioning and orientation while in orbit. The purpose of this work is to conduct a comparative study between a few variants of the Kalman Filter that can be used for the intents outlined previously as well as comparing them to a few simpler estimators like the alpha filter and the Quaternion Estimator (QUEST). The main variants to be analyzed in this work are: the Extended Kalman Filter (EKF), the Adaptive Kalman Filter and the Ensemble Kalman Filter. A few other filters will also be included to make a less rigorous comparison with some of the analysed filters. Satellites operate in one of the most dynamic and unpredictable environments there are. The nonlinear dynamics and the inherent uncertainties of a satellite's motion require sophisticated methods of filtering with the intent of providing the means for an efficient and accurate attitude estimation and control, which are essential for an accomplished space mission. The wrong satellite orientation may result in batteries not charging efficiently as well as interrupted communication, which can in turn lead to a total mission failure. Using a simulation-based approach, this work aims to evaluate both the performance, accuracy and computational efficiency of each Kalman Filter variant under the same dynamic conditions and orbital scenarios. For this end the Self-DePloyable FLexible AeroSHell for de-Orbiting and Space Re-entry (SPLASH) conceptual satellite is going to be used as object of study along with the orbital scenario it was designed for. For the simulation itself MATLAB will be used to perform all the numerical computation necessary for the accurate simulation of both the satellite and the space environment. For the estimation three sensors will be modeled, with noise and errors, to provide the necessary data for the calculations.

Keywords: Attitude. CubeSat. Kalman. Estimation.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

ADCS	*Attitude Determination and Control System*
ADEPT	*Adaptable, Deployable Entry and PlacementTechnology*
AKF	*Adaptive Kalman Filter*
AOCS	*Attitude and Orbit Control Subsystem*
CAPE	*CubeSat Application for Planetary Entry Mission*
CIRA	*Centro Italiano Ricerche Aerospaziali*
CONFAP	Conselho Nacional das Fundações Estaduais de Amparo à Pesquisa
DSS	*Digital Sun Sensor* - Sensor Solar Digital
EKF	*Extended Kalman Filter*
EnKF	*Ensemble Kalman Filter*
EnSRF	*Ensemble Square Root Filter*
IRENE	*Italian Re-Entry Nacelle for Microgravity Experiments*
ISS	*International Space Station*
MAECI	Ministério dos Negócios Estrangeiros da Itália
MATLAB	*MATrix LABoratory*
NASA	*National Aeronautics and Space Administration*
NRLMSISE-00	*Naval Research Laboratory Mass Spectrometer and Incoherent Scatter Radar Extended - 2000*
QUEST	*Quaternion Estimator*
RIG	*Rate Integrating Gyroscope* - Giroscópio Integrador de Taxa
SPLASH	*Self-DePloyable FLexible AeroSHell for de-Orbiting and Space Re-entry*
TAM	*Three-Axis Magnetometer* - Magnetômetro Triaxial
TRMM	*Tropical Rainfall Measuring Mission*
UARS	*Upper Atmosphere Research Satellite*

Sumário

1 Introdução

A estimação de atitude, que normalmente faz parte dos subsistemas de *Attitude Determination and Control System* (ADCS) e/ou de *Attitude and Orbit Control Subsystem* (AOCS) , constitui um componente essencial de qualquer missão espacial. Essa atividade desempenha um papel fundamental no êxito ou na eventual falha de tais empreendimentos. A correta determinação da atitude de um satélite, por exemplo, possibilita o controle apropriado para que o mesmo consiga direcionar câmeras ou antenas para alvos específicos e desejados. Esse processo de estimação é realizado utilizando-se diversos sensores que capturam medições variáveis, as quais são analisadas e empregadas em estimadores, como o Filtro de Kalman Estendido, Filtro de Partículas, entre outros métodos adotados na área.

A estimativa de atitude se torna especialmente necessária para veículos espaciais projetados para realizar reentradas na atmosfera, desde cápsulas tripuladas como Soyuz e Crew Dragon até satélites de demonstração como o SPLASH, considerando que esses dispositivos devem manter uma orientação específica durante o processo de entrada. Essa orientação é fundamental para garantir uma reentrada bem-sucedida e segura. Um exemplo notável de satélite que apresenta essa característica é o S.P.L.A.S.H. que, em sua concepção, possui um escudo projetado em um formato semelhante ao de um guarda-chuva, o qual é acionado e se abre quando o satélite executa a manobra de reentrada na atmosfera. Essa estratégia é crucial para a proteção do satélite, proporcionando uma forma eficiente de dissipar a energia gerada durante a passagem na camada atmosférica, minimizando danos e garantindo a integridade dos sistemas internos.

O Filtro de Kalman Estendido destaca-se como um dos métodos mais utilizados para a estimação, especialmente no âmbito da determinação da atitude de satélites e outros veículos que operam no espaço. Essa relevância se deve à sua capacidade de tratar incertezas que podem ocorrer tanto nos modelos empregados quanto nas medições realizadas. Além disso, sua habilidade em lidar com problemas dinâmicos não-lineares, que é frequentemente observado na tarefa de estimação da atitude de um satélite, torna-o uma escolha preferencial na área.

1.1 Objetivo Geral

O objetivo primordial deste trabalho de conclusão de curso é efetuar uma comparação da eficácia de diversos tipos de técnicas de filtragem que são empregadas na estimação de atitude de satélites. Para tanto, foi escolhido como objeto de estudo o satélite SPLASH, um satélite conceitual desenvolvido por meio de uma colaboração entre o Ministério dos Negócios Estrangeiros da Itália (MAECI) e o Conselho Nacional das Fundações Estaduais

de Amparo à Pesquisa (CONFAP), coordenado pelo *Centro Italiano Ricerche Aerospaziali* (CIRA) . Essa missão tem como objetivo o desenvolvimento de um escudo de proteção para satélites para reentrada atmosférica. Este satélite se revela um objeto de estudo pertinente, pois demanda uma estimação de atitude robusta para garantir o êxito de sua missão.

1.2 Objetivo Específico

Afim de alcançar o objetivo geral deste trabalho, este foi dividido em diversos objetivos mais específicos:

- Fazer uma revisão de toda a fundamentação teórica da cinemática e da dinâmica de satélites além dos sensores utilizados para estimar a atitude de satélites
- Fazer a implementação dos métodos de estimação que serão objetos de estudos no programa MATLAB
- Adaptar o código de estimação de atitude do satélite *Tropical Rainfall Measuring Mission* (TRMM) disponibilizado no livro *Fundamentals of Spacecraft Attitude Determination and Control* (Markley F. L.; Crassidis, 2014) com os novos filtros e parâmetros orbitais do satélite S.P.L.A.S.H.
- Fazer uma comparação entre as diferentes estimações de atitude obtidas

1.3 Justificativa

Com o contínuo avanço tecnológico dos processadores, que incorporam transistores em dimensões cada vez menores e, por conseguinte, aumentam significativamente seu poder de processamento, surge a viabilidade de conduzir um estudo aprofundado sobre a utilização de filtros de Kalman com melhor desempenho em presença de outliers. Esses filtros podem aproveitar todo esse potencial de processamento disponível para realizar uma estimativa de atitude com maior acurácia. Essa melhoria não apenas possibilita uma estimativa mais acurada, mas também torna factível a execução de um controle de atitude de maneira mais efetiva.

Esses filtros possuem a capacidade de gerenciar uma quantidade significativa de incertezas e não linearidades, o que contribui para a redução de custos ao diminuir a dependência de sensores que precisam ser extremamente acurados. Essa redução de custos beneficia não apenas órgãos governamentais, mas também empresas privadas que apresentam interesse em desenvolver satélites com um custo inferior.

A justificativa para a execução deste trabalho se origina da incerteza provocada pela ampla quantidade de variações do filtro de Kalman. Essas variações foram desenvolvidas ao longo dos anos, desde a sua primeira menção em um artigo que tratou da estimação de

atitude, no ano de 1970, por James Farrell(Farrell, 1970). Nesse cenário, a principal pergunta que se apresenta é: qual variação, dentre as diversas selecionadas para o estudo, oferece a estimativa mais acurada, mantendo, ao mesmo tempo, um custo computacional reduzido?

1.4 Organização do Trabalho

O Capítulo 2 oferece uma contextualização detalhada sobre um tipo específico de satélite que requer uma estimação de atitude que seja ao mesmo tempo robusta e eficiente: os satélites de reentrada. Esses satélites exercem uma função essencial em diversas missões, e sua habilidade de manter uma orientação precisa durante o processo de reentrada na atmosfera é absolutamente fundamental para garantir o sucesso das operações, além de assegurar a proteção dos equipamentos presentes a bordo.

O Capítulo 3 deste trabalho tem como objetivo apresentar a fundamentação teórica básica que se faz necessária para o adequado desenvolvimento e a efetiva realização do trabalho proposto.

No Capítulo 4, são apresentados as principais variações do filtro de Kalman a serem avaliados neste trabalho. Além disso, há a inclusão de alguns filtros mais básicos, os quais servirão como referência para comparação e análise.

O Capítulo 5 apresenta os resultados e uma discussão comparativa sobre os filtros analisados.

No capítulo 6 são apresentados as ideias conclusivas sobre este trabalho.

Ao final do documento, encontra-se o código adaptado para a elaboração deste trabalho, onde são implementados o Filtro de Kalman Estendido, o Filtro de Kalman Adaptativo Basedo em Inovação, o Filtro de Kalman Ensemble e o Filtro de Kalman Raiz Quadrada de Ensemble.

2 Satélites de reentrada atmosférica

Satélites concebidos para reentrada atmosférica representam um avanço significativo no âmbito das missões espaciais por diversos motivos. Um dos aspectos mais notáveis é a possibilidade de recuperar integralmente esses satélites ao término de seu ciclo operacional, em vez de serem simplesmente descartados na atmosfera. Essa prática contrasta com a maioria dos satélites que, durante sua operação, utilizam o arrasto atmosférico como um meio de se desfazer de maneira controlada ao final de suas missões, como por exemplo o satélite *Upper Atmosphere Research Satellite* (UARS) , desenvolvido pela *National Aeronautics and Space Administration* (NASA) , lançado em 1991, e que custou cerca de 750 milhões de dólares para ser construído (Goddard Space Flight Center, 2025). Desenvolver satélites que possam fazer reentrada atmosférica não apenas contribui para a preservação de recursos valiosos, mas também abre a possibilidade de estudo e reutilização das tecnologias embarcadas. Além disso, proporciona um maior controle sobre o local de queda de satélites que foram desativados.

Uma outra motivação significativa para o desenvolvimento de sistemas de reentrada para satélites é a possibilidade de enviar esses veículos espaciais para planetas que possuem uma atmosfera que, devido às elevadas temperaturas ou características corrosivas, podem afetar negativamente a integridade física do satélite, diminuindo substancialmente o seu tempo de vida.

Nas seções que se seguem, alguns satélites que foram desenvolvidos com o propósito de testar sistemas de reentrada serão apresentados. Esta análise culminará com a descrição do satélite que é objeto de estudo deste trabalho, denominado SPLASH.

2.1 CAPE-MIRCA

O CAPE-MIRCA foi um satélite parte do conceito *CubeSat Application for Planetary Entry Missions* (CAPE) de satélites que visava o desenvolvimento de CubeSats com módulos de propulsão além de tecnologias que permitissem a reentrada atmosférica do satélite.

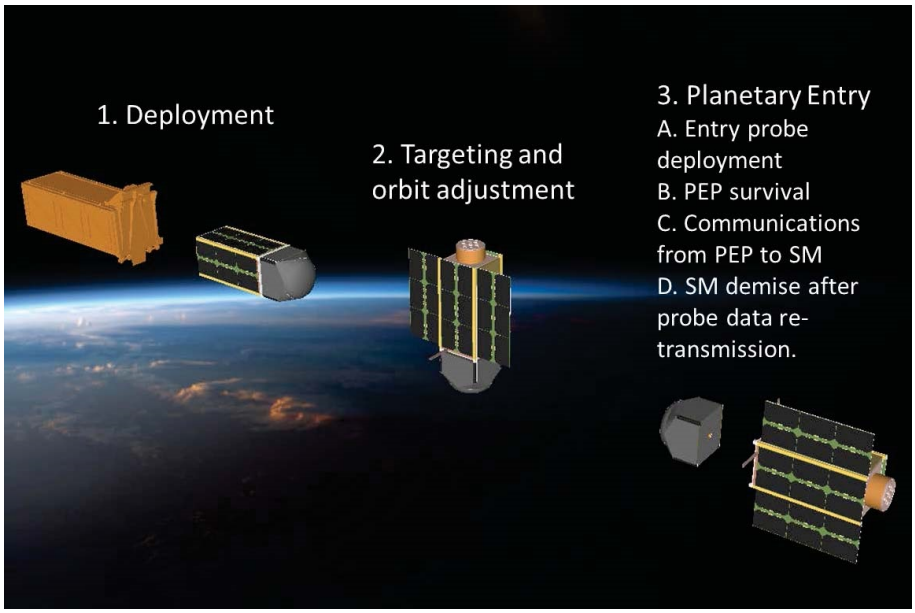


Figura 2.1 – CAPE-MIRCA. Fonte: NASA

O sistema de reentrada consistia em um módulo que se separaria do satélite levando consigo sensores que fariam medições das características atmosféricas terrestres durante sua reentrada. Esse módulo era revestido por um material altamente resistente às variações de temperatura, que sofreria um desgaste controlado à medida que passasse pela atmosfera, garantindo que as medições fossem realizadas de forma eficaz e segura. O satélite foi testado em 2015 ao ser levado por um balão estratosférico da NASA até os limites da atmosfera (Esper, 2025).

2.2 ADEPT

O projeto *Adaptable, Deployable Entry and Placement Technology* (ADEPT) tem como objetivo o desenvolvimento de um sistema de reentrada que se apresenta na forma de um aeroescudo semi-rígido. Este dispositivo possui um coeficiente balístico reduzido, o que facilita a entrada atmosférica e a desaceleração de satélites durante missões planetárias. Uma das características mais notáveis desse sistema é a sua flexibilidade, permitindo que ele seja retrátil. O aeroescudo é confeccionado a partir de um tecido de fibra de carbono (Cassell *et al.*, 2018).

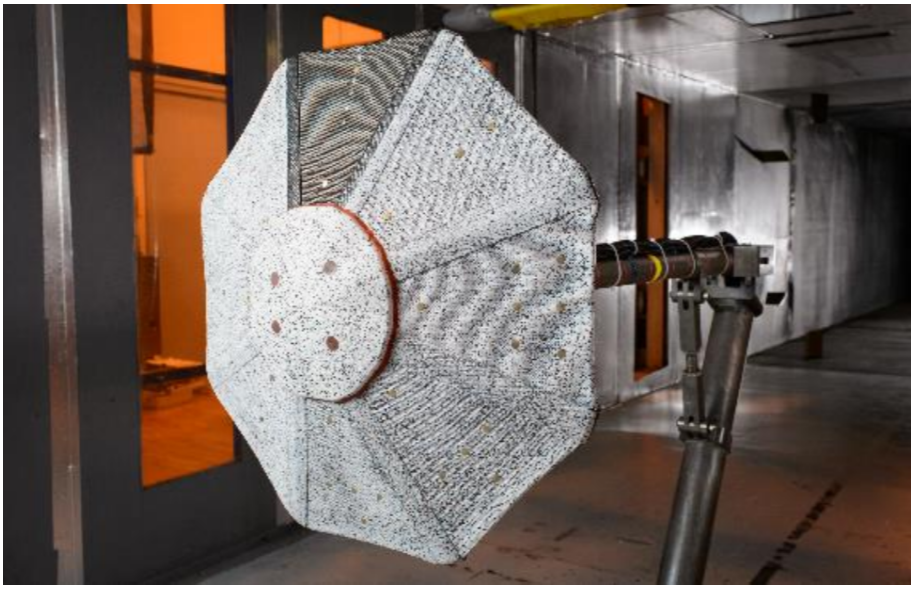


Figura 2.2 – Protótipo do aeroescudo. Fonte:(Cassell *et al.*, 2018)

2.3 IRENE

O *Italian Re-Entry Nacelle for Microgravity Experiments* (IRENE) é uma cápsula de reentrada com escudo inflável em formato de guarda-chuva com o objetivo de permitir, entre outras possibilidades, o envio de paylods da *International Space Station* (ISS) de volta para a terra além de ser um possível veículo de retorno para missões científicas realizadas em órbita terrestre baixa. Esse projeto foi desenvolvido em colaboração entre o *ALI Consortium*, o CIRA e a renomada Universidade de Nápoles Federico II. O teste da cápsula foi realizado com êxito no ano de 2022, por meio do lançamento de um foguete sub-orbital (Mungiguerra, 2023)(Bassano *et al.*, 2011).

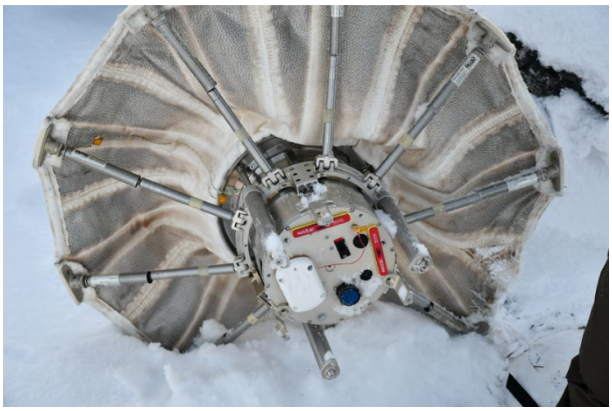


Figura 2.3 – Cápsula de teste. Fonte: (Mungiguerra, 2023)

2.4 SPLASH

O SPLASH é um conceito proposta na forma de um sistema de proteção térmica no formato de guarda-chuva. Ao contrário dos outros sistemas propostos até o presente momento, o SPLASH não faz uso de uma estrutura inflável. O seu conceito se assemelha ao funcionamento real de uma guarda-chuva, onde um tecido com propriedades de resistência térmica é aberto por meio de braços mecânicos. Esses braços mecânicos podem se mover de forma independente, mudando o formato do escudo para a realização de uma reentrada mais precisa. O projeto foi concebido como uma parceria entre órgãos governamentais brasileiros e italianos (Dimino *et al.*, 2023).

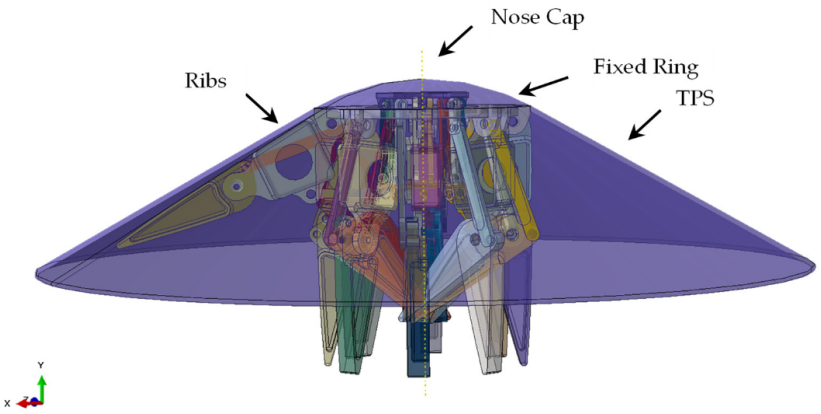


Figura 2.4 – Estrutura proposta do SPLASH. Fonte:(Dimino *et al.*, 2023)

3 Fundamentação Teórica

3.1 Cinemática

A cinemática tem como objetivo explicar as características que são inerentes ao movimento de um corpo, sem levar em conta as causas que provocam esse movimento (Markley F. L.; Crassidis, 2014). Em outras palavras, a cinemática analisa o movimento sem considerar os efeitos de forças, torques e demais interações que possam estar atuando sobre o corpo em questão, descrevendo a correlação entre a mudança na orientação de um corpo e sua velocidade angular (Xie, 2022). Essa abordagem oferece uma compreensão mais aprofundada das trajetórias, velocidades e acelerações dos objetos, sendo essencial para a análise do comportamento dinâmico dos sistemas que se encontram em movimento. A cinemática de atitude pode ser representada de diversas maneiras, com algumas dessas maneiras sendo detalhadas a seguir, mas este trabalho fará uso, principalmente, da representação por meio de quatérnions (Shuster, 1993).

3.1.1 Parametrizações

3.1.1.1 Ângulos de Euler

Ângulos de Euler descrevem atitude através de três rotações consecutivas em eixos distintos (Markley F. L.; Crassidis, 2014). A rotação transforma o referencial do corpo para coincidir com o referencial inercial desejado.

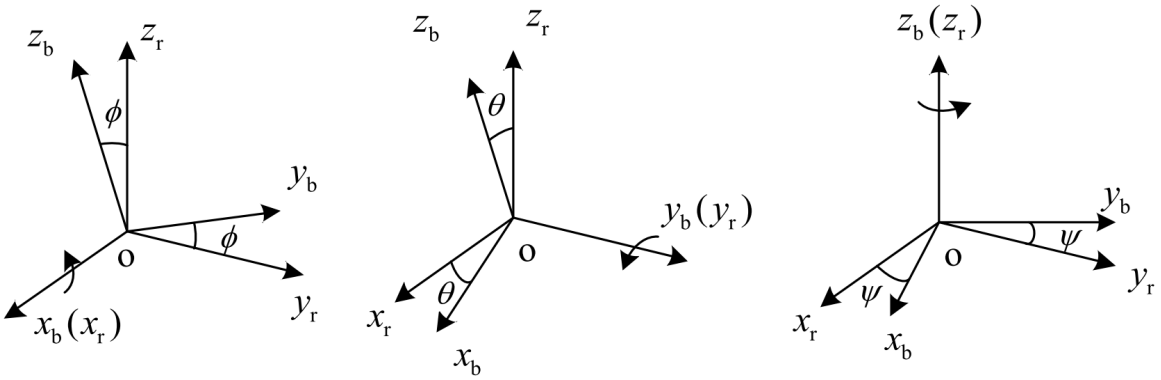


Figura 3.1 – Rotações realizadas em torno dos três eixos. Fonte:(Xie, 2022)

Matrizes são utilizadas para representar diferentes tipos de rotação em torno de eixos específicos e distintos. As matrizes básicas são mostradas a seguir.

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{3.1}$$

$$R_y(\varphi) = \begin{bmatrix} \cos(\varphi) & 0 & \sin(\varphi) \\ 0 & 1 & 0 \\ -\sin(\varphi) & 0 & \cos(\varphi) \end{bmatrix} \tag{3.2}$$

$$R_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

A matriz resultante, que descreve as três rotações é chamada de matriz de cossenos diretores, também conhecida como matriz de atitude. No total, existem 12 possíveis sequências de rotação, geradas a partir da combinação das três matrizes fundamentais (Xie, 2022). Os eixos x, y e z são geralmente denominados de 1, 2 e 3, respectivamente, quando referenciados no contexto de uma sequência de rotações de ângulos Euler. A sequência mostrada na figura 3.1, por exemplo, realiza uma rotação no eixo x, depois uma rotação no eixo y e uma última rotação no eixo z. Essa sequência específica é conhecida como 1-2-3 e possui a seguinte matriz de atitude — os termos cos e sen são substituídos por c e s, respectivamente, com o intuito de tornar o texto mais conciso:

$$R_{123} = \begin{bmatrix} c(\varphi)c(\psi) & -c(\varphi)s(\psi) & s(\varphi) \\ c(\theta)s(\psi) + s(\theta)s(\varphi)c(\psi) & c(\theta)c(\psi) - s(\theta)s(\varphi)s(\psi) & -s(\theta)c(\varphi) \\ s(\theta)s(\psi) - c(\theta)s(\varphi)c(\psi) & s(\theta)c(\psi) + c(\theta)s(\varphi)s(\psi) & c(\theta)c(\varphi) \end{bmatrix} \tag{3.4}$$

Neste trabalho, utiliza-se a sequência de rotação 3-2-1 (yaw-pitch-roll), também conhecida como sequência aeronáutica ou ZYX, que é a convenção mais comum em aplicações de estimação de atitude de satélites (Markley F. L.; Crassidis, 2014). A conversão entre quatérnions e ângulos de Euler depende da sequência escolhida, portanto esta deve ser mantida consistente em todo o documento.

Um dos problemas da representação por ângulos de Euler é o fenômeno conhecido como *gimbal lock*, que se caracteriza pela perda de um grau de liberdade quando dois eixos coincide após uma rotação de 90° em um eixo específico (Markley F. L.; Crassidis, 2014). É importante notar que o gimbal lock é uma limitação da parametrização por ângulos de Euler, não da matriz de cossenos diretores (DCM) em si, que é uma representação completa e não-singular. Esse fenômeno resulta na situação em que a rotação em um determinado eixo acaba gerando, de maneira involuntária, uma rotação em outro eixo. Isso ocorre devido ao alinhamento existente entre os eixos envolvidos. Esse tipo de situação é frequentemente

observado, por exemplo, em sistemas físicos que utilizam três suspensões cardãs, sendo uma destinada a cada um dos eixos envolvidos. Esse dilema pode ser abordado e solucionado por meio de diversas estratégias, e uma das abordagens que pode ser adotada é a utilização de diferentes sistemas de representação. Um desses sistemas é o eixo-ângulo de Euler, o qual será explorado e apresentado na seção a seguir.

3.1.1.2 Eixo-ângulo de Euler

A representação por eixo-ângulo de Euler é considerada uma abordagem mais geral e compacta (Xie, 2022) para descrever as rotações de um objeto tridimensional. Essa representação utiliza uma única rotação em torno de um eixo arbitrário para representar qualquer orientação, evitando as singularidades dos ângulos de Euler.

$$\mathbf{r} = \theta\hat{e} = \theta \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} \tag{3.5}$$

com $\mathbf{r}$ o vetor de rotação, $\hat{e}$ o eixo unitário e θ o ângulo de rotação.

Dessa maneira, é possível realizar a conversão das rotações que estão formatadas em ângulos de Euler para o formato eixo-ângulo de Euler, utilizando as definições a seguir:

$$\theta = \arccos\left(\frac{\mathrm{tr}(\mathbf{R}) - 1}{2}\right) \tag{3.6}$$

$$\mathbf{e} = \frac{1}{2\sin(\theta)} \begin{pmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{pmatrix} \tag{3.7}$$

onde $\mathbf{R}$ é a matriz de atitude e R_{ij} é o elemento na linha i, coluna j.

A partir deste momento, é possível estabelecer a matriz de atitude por meio da utilização da fórmula de rotação de Rodrigues (Xie, 2022; Markley F. L.; Crassidis, 2014). Essa fórmula é amplamente utilizada em campos que envolvem a dinâmica de sistemas rotacionais, pois permite a conversão de um vetor de rotação em uma matriz de rotação.

$$\mathbf{R} = \mathbf{I} + \sin(\theta)\mathbf{K} + (1 - \cos(\theta))\mathbf{K}^2 \tag{3.8}$$

onde $\mathbf{I}$ é a matriz identidade e $\mathbf{K}$ é a matriz anti-simétrica de $\hat{e}$:

$$\mathbf{K} = \begin{bmatrix} 0 & -e_3 & e_2 \\ e_3 & 0 & -e_1 \\ -e_2 & e_1 & 0 \end{bmatrix} \tag{3.9}$$

Dessa maneira, a expressão ampliada da matriz de atitude em eixo-ângulo de Euler, a qual efetua a transformação de vetores do referencial do corpo para o referencial inercial, é apresentada da seguinte forma:

$$\mathbf{R} = \begin{bmatrix} c\theta + e_1^2(1-c\theta) & e_1e_2(1-c\theta) - e_3s\theta & e_1e_3(1-c\theta) + e_2s\theta \\ e_2e_1(1-c\theta) + e_3s\theta & c\theta + e_2^2(1-c\theta) & e_2e_3(1-c\theta) - e_1s\theta \\ e_3e_1(1-c\theta) - e_2s\theta & e_3e_2(1-c\theta) + e_1s\theta & c\theta + e_3^2(1-c\theta) \end{bmatrix} \tag{3.10}$$

Embora essa representação consiga solucionar a problemática de *gimbal lock* e apresente uma abordagem mais intuitiva em comparação com os ângulos de Euler para fins de análise, é importante notar que, do ponto de vista computacional, ela não é a opção mais eficiente disponível (Markley F. L.; Crassidis, 2014). Dessa forma, este trabalho utiliza quatérnions e a matemática que os envolve para realizar esses cálculos.

3.1.1.3 Quatérnions

Quatérnions foram inicialmente propostos por Hamilton (Hamilton, 1844) como um sistema numérico para extender os números complexos, denominados de números hipercomplexos. Este sistema numérico partiu da necessidade de Hamilton encontrar uma forma de obter o quociente de dois pontos em um espaço tri-dimensional, pontos estes que poderiam ser interpretados como números complexos. Um quatérnion é definido como:

$$q = q_0 + iq_1 + jq_2 + kq_3 \quad \text{onde} \quad q_0, q_1, q_2, q_3 \in \mathbb{R} \tag{3.11}$$

Para alcançar os objetivos propostos neste trabalho, a representação de quatérnions que será utilizada é a mesma estabelecida por Crassidis e Markley (Markley F. L.; Crassidis, 2014), onde a parte escalar é posicionada na parte inferior.

$$\mathbf{q} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_0 \end{bmatrix} \tag{3.12}$$

Também é de grande relevância definir o conjugado de um quatérnion, que para um quatérnion unitário corresponde à sua inversa:

$$\mathbf{q}^* = \begin{bmatrix} -q_1 \\ -q_2 \\ -q_3 \\ q_0 \end{bmatrix} \tag{3.13}$$

Dessa maneira, é viável representar a atitude de um corpo em relação a um referencial inercial por meio de quatérnions unitários. A angulação de uma aeronave, por exemplo, é conhecida como arfagem, rolagem e guinada. Estes movimentos são comumente representados em ângulos de Euler, utilizando o sistema de referência inercial denominado North–East–Down (NED). Neste sistema, os eixos são definidos de forma que x corresponde ao Norte polar, y representa o Leste terrestre e z aponta para baixo, em direção ao centro da Terra. A conversão de ângulos de Euler para quatérnions é dada por (Markley F. L.; Crassidis, 2014):

$$\begin{aligned}
q_1 &= \sin\left(\frac{\vartheta}{2}\right)\cos\left(\frac{\phi}{2}\right)\cos\left(\frac{\psi}{2}\right) - \cos\left(\frac{\vartheta}{2}\right)\sin\left(\frac{\phi}{2}\right)\sin\left(\frac{\psi}{2}\right) \\
q_2 &= \cos\left(\frac{\vartheta}{2}\right)\sin\left(\frac{\phi}{2}\right)\cos\left(\frac{\psi}{2}\right) + \sin\left(\frac{\vartheta}{2}\right)\cos\left(\frac{\phi}{2}\right)\sin\left(\frac{\psi}{2}\right) \\
q_3 &= \cos\left(\frac{\vartheta}{2}\right)\cos\left(\frac{\phi}{2}\right)\sin\left(\frac{\psi}{2}\right) - \sin\left(\frac{\vartheta}{2}\right)\sin\left(\frac{\phi}{2}\right)\cos\left(\frac{\psi}{2}\right) \\
q_0 &= \cos\left(\frac{\vartheta}{2}\right)\cos\left(\frac{\phi}{2}\right)\cos\left(\frac{\psi}{2}\right) + \sin\left(\frac{\vartheta}{2}\right)\sin\left(\frac{\phi}{2}\right)\sin\left(\frac{\psi}{2}\right)
\end{aligned} \tag{3.14}$$

Além disso, é possível realizar a conversão de um eixo-ângulo de Euler para um quatérnion, operação que será extremamente útil em etapas posteriores, especialmente para efetuar rotações em um vetor. Esse processo pode ser facilmente executado da seguinte maneira (Markley F. L.; Crassidis, 2014):

$$\mathbf{q} = \begin{bmatrix} \sin\frac{\theta}{2}\,\mathbf{e}_1 \\ \sin\frac{\theta}{2}\,\mathbf{e}_2 \\ \sin\frac{\theta}{2}\,\mathbf{e}_3 \\ \cos\frac{\theta}{2} \end{bmatrix} \tag{3.15}$$

Tendo em vista esses conceitos, é possível representar rotações por meio da utilização de quatérnions. Para que isso ocorra, é essencial, em primeiro lugar, entender como a multiplicação desses quatérnions é realizada, uma vez que essa operação é indispensável para realizar rotações utilizando esse método específico. A multiplicação de dois quatérnions é expressa da seguinte maneira:

$$\mathbf{q} \otimes \mathbf{p} = \begin{bmatrix} q_0p_1 + q_1p_0 + q_2p_3 - q_3p_2 \\ q_0p_2 - q_1p_3 + q_2p_0 + q_3p_1 \\ q_0p_3 + q_1p_2 - q_2p_1 + q_3p_0 \\ q_0p_0 - q_1p_1 - q_2p_2 - q_3p_3 \end{bmatrix} \tag{3.16}$$

É importante notar que a ordem dos elementos multiplicados é importante. Com essa operação em mãos, é possível, por exemplo, aplicar uma rotação a um vetor em três dimensões utilizando quatérnions. Para efetuar esse procedimento de maneira adequada,

inicialmente, é imprescindível realizar a conversão do vetor em um quatérnion, no qual os componentes do vetor correspondem aos primeiros elementos do quatérnion, enquanto a parte escalar é definida como igual a zero:

$$\mathbf{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ 0 \end{bmatrix} \tag{3.17}$$

A equação 3.15 é utilizada para determinar o quatérnion que representa a rotação a ser aplicada ao vetor considerado. Em seguida, uma sequência de multiplicações entre quatérnions é realizada com o intuito de obter o vetor rotacionado desejado. Inicialmente, o quatérnion de rotação é multiplicado pelo quatérnion correspondente ao vetor. Após essa operação, a multiplicação pelo conjugado do quatérnion de rotação é executada, completando assim o cálculo necessário para a transformação do vetor na nova orientação (Markley F. L.; Crassidis, 2014).

$$\mathbf{v}' = \mathbf{q} \otimes \mathbf{v} \otimes \mathbf{q}^* \tag{3.18}$$

Este método é conhecido como método conjugado e é empregado principalmente na tarefa de converter medições obtidas no referencial do corpo, utilizando sensores, por exemplo, para o referencial inercial. Outro método, identificado como o método direto, será empregado para a estimação da atitude de um satélite ao longo do tempo. Esse método será detalhado na Seção 3.1.2.1.

A matriz de atitude pode ser obtida a partir de um quatérnion utilizando a seguinte equação (Markley F. L.; Crassidis, 2014):

$$\mathbf{R} = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & 1-2(q_1^2+q_3^2) & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & 1-2(q_1^2+q_2^2) \end{bmatrix} \tag{3.19}$$

A conversão entre as três parametrizações que foram apresentadas até o momento é de suma importância, pois, apesar de os quatérnions se mostrarem mais eficientes do ponto de vista computacional no que diz respeito ao cálculo de rotações, os ângulos de Euler se revelam mais intuitivos quando se trata de visualização e análises (Shuster, 1993).

3.1.2 Cinemática de Atitude

As parametrizações anteriores não incluíram velocidade angular — componente essencial da cinemática (Markley F. L.; Crassidis, 2014). A velocidade angular será introduzida usando quatérnions, a representação principal deste trabalho.

O giroscópio mede a velocidade angular nos três eixos, representada pelo vetor $\boldsymbol{\omega}$:

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{3.20}$$

A velocidade angular é um conceito de grande relevância, uma vez que possibilita a avaliação da variação da atitude. Em outras palavras, essa medida nos fornece informações sobre a rapidez com que a orientação de um corpo se altera. Para calcular a variação do quatérnion, transformamos o vetor de velocidade angular em um quatérnion, conforme ilustrado em 3.17, adicionando um componente escalar igual a zero ao final. Posteriormente, a derivada do quatérnion em relação ao tempo é calculada, conforme descrito em (Markley F. L.; Crassidis, 2014):

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \otimes \boldsymbol{\omega} \tag{3.21}$$

Para otimizar essa operação sob a perspectiva computacional, é bastante viável realizar a conversão do processo em uma multiplicação entre uma matriz e um vetor. Com esse objetivo, o vetor $\boldsymbol{\omega}$ é transformado em uma matriz 4x4, cujo formato é anti-simétrico, e a essa matriz designamos o nome de $\boldsymbol{\Omega}$:

$$\boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix} \tag{3.22}$$

Isso possibilita a obtenção da variação do quatérnion por meio da seguinte operação (Markley F. L.; Crassidis, 2014):

$$\dot{\mathbf{q}} = \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega})\mathbf{q} \tag{3.23}$$

A transformação da representação da velocidade angular de vetor para matriz possibilita a execução de operações computacionais com maior eficiência. Isso se deve ao fato de que os computadores são projetados para realizar multiplicações de matrizes e vetores de maneira otimizada. É importante ressaltar que essa abordagem assegura que o resultado obtido permaneça equivalente ao que seria alcançado por meio da álgebra de quatérnions.

Existem diferentes convenções na literatura para a matriz $\boldsymbol{\Omega}$, com variações nos sinais de alguns elementos dependendo da convenção de multiplicação de quatérnions adotada. Este trabalho segue a convenção de Crassidis & Markley (Markley F. L.; Crassidis, 2014).

3.1.2.1 Propagação Discreta de Quatérnions

Para integração numérica, a forma discreta exata assumindo $\boldsymbol{\omega}$ constante é:

$$\mathbf{q}_{k+1} = \boldsymbol{\Phi}(\boldsymbol{\omega}_k, \Delta t)\mathbf{q}_k \tag{3.24}$$

onde $\boldsymbol{\Phi}$ é a matriz de transição de estado do quatérnion, definida como:

$$\boldsymbol{\Phi}(\boldsymbol{\omega}, \Delta t) = \begin{bmatrix} \cos\frac{\theta}{2} & n_z \sin\frac{\theta}{2} & -n_y \sin\frac{\theta}{2} & n_x \sin\frac{\theta}{2} \\ -n_z \sin\frac{\theta}{2} & \cos\frac{\theta}{2} & n_x \sin\frac{\theta}{2} & n_y \sin\frac{\theta}{2} \\ n_y \sin\frac{\theta}{2} & -n_x \sin\frac{\theta}{2} & \cos\frac{\theta}{2} & n_z \sin\frac{\theta}{2} \\ -n_x \sin\frac{\theta}{2} & -n_y \sin\frac{\theta}{2} & -n_z \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{3.25}$$

com $\theta = \|\boldsymbol{\omega}\|\Delta t$ e $\mathbf{n} = \boldsymbol{\omega}/\|\boldsymbol{\omega}\|$ sendo o eixo unitário de rotação (Markley F. L.; Crassidis, 2014).

3.2 Dinâmica

A dinâmica de atitude estuda movimentos rotacionais e os torques que os causam (Markley F. L.; Crassidis, 2014; Xie, 2022). As equações que governam este campo de estudo estabelecem relações entre a velocidade angular de um corpo, o tensor de momento de inércia e os torques externos que atuam sobre ele. Considerando que não se fará uso de atuadores no contexto deste trabalho, a modelagem da dinâmica de atitude será tratada de maneira mais simplificada e direta. Essa equação é conhecida como equação do movimento de Euler (Markley F. L.; Crassidis, 2014):

$$\mathbf{J}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\mathbf{J}\boldsymbol{\omega}) = \boldsymbol{\tau} \tag{3.26}$$

onde $\mathbf{J}$ é o tensor momento de inércia:

$$\mathbf{J} = \begin{bmatrix} J_{xx} & J_{xy} & J_{xz} \\ J_{yx} & J_{yy} & J_{yz} \\ J_{zx} & J_{zy} & J_{zz} \end{bmatrix} \tag{3.27}$$

$\boldsymbol{\omega}$ é a velocidade angular, $\dot{\boldsymbol{\omega}}$ a aceleração angular e $\boldsymbol{\tau}$ o torque externo.

3.3 Sensores de Atitude

Para se conseguir calcular a atitude de um satélite, é imprescindível o uso de sensores, pois eles fornecem as medições necessárias para essa estimativa. Sem esses dispositivos,

seria inviável determinar a orientação adequada do satélite enquanto ele opera no espaço, dado que os sensores possibilitam a coleta de dados cruciais para essa finalidade.

Diversos tipos de sensores podem ser encontrados no mercado (Wertz, 1978). No entanto, para os objetivos deste estudo, nossa atenção estará voltada exclusivamente para três deles: o DSS (Digital Sun Sensor, Sensor Solar Digital) , o TAM (Three-Axis Magnetometer, Magnetômetro Triaxial) e o RIG (Rate Integrating Gyroscope, Giroscópio Integrador de Taxa) . Sensores dessa natureza foram selecionados especificamente devido à sua frequente utilização como a base principal na determinação da atitude, especialmente em situações onde os sensores primários deixam de funcionar adequadamente (Markley F. L.; Crassidis, 2014).

3.3.1 Modelo Vetorial Genérico

Todos os sensores vetoriais (magnetômetro, sensores solares) seguem um modelo geral que relaciona a medição no referencial do corpo com o vetor de referência no referencial inercial (Markley F. L.; Crassidis, 2014; Wertz, 1978):

$$\mathbf{z} = \mathbf{A}(\mathbf{q})\mathbf{r}_i + \mathbf{v} \tag{3.28}$$

onde $\mathbf{z} \in \mathbb{R}^3$ é o vetor medido no referencial do corpo, $\mathbf{A}(\mathbf{q}) \in SO(3)$ é a matriz de rotação do quatérnion (Eq. 3.19), $\mathbf{r}_i \in \mathbb{R}^3$ é o vetor de referência inercial e $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ é ruído gaussiano de média zero.

Alguma suposições precisam ser feitas para que modelo seja verdade:

1. O sensor está rigidamente montado no corpo do satélite
2. O referencial do sensor e do corpo são alinhados entre si
3. O ruído de medição é aditivo e gaussiano
4. A referência inercial $\mathbf{r}_i$ é conhecida com precisão

Para utilização em filtros de Kalman, a medição é linearizada em torno da estimativa atual, resultando na equação de inovação (Markley F. L.; Crassidis, 2014):

$$\boldsymbol{\nu} = \mathbf{z} - \mathbf{A}(\hat{\mathbf{q}})\mathbf{r}_i \approx -\mathbf{A}(\hat{\mathbf{q}})[\mathbf{r}_i\times]\,\delta\boldsymbol{\theta} + \mathbf{v} \tag{3.29}$$

onde $[\mathbf{r}_i\times]$ é a matriz anti-simétrica de $\mathbf{r}_i$ e $\delta\boldsymbol{\theta}$ é o vetor de erro de atitude.

3.3.2 DSS

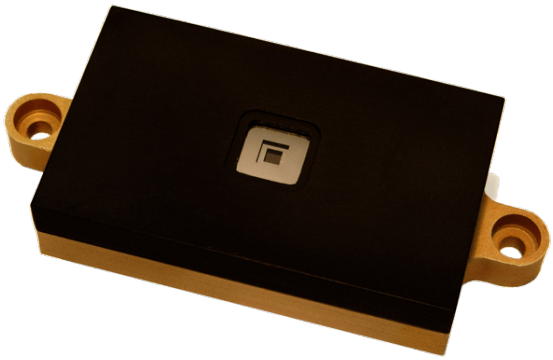


Figura 3.2 – *Digital Sun Sensor* (CubeSatShop, 2023)

Um Sensor Solar Digital é composto por uma matriz de fotocélulas que gera um sinal quando expostas à luz. Este sensor é geralmente colocado atrás de uma abertura estreita, permitindo que a luz passe por ela. A abertura cria uma sombra variável sobre as fotocélulas, iluminando seletivamente certas células dependentemente do ângulo pelo qual a luz solar incide. Com base nessa distribuição seletiva de luz e sombra, o dispositivo eletrônico é capaz de determinar a posição da fonte luminosa em relação à face onde o sensor está montado (Wertz, 1978).

Para que o sensor possa ser utilizado para estimação, é necessário que o seu funcionamento seja descrito de forma matemática. Desta forma, o modelo de medição de um Sensor Solar Digital é dado da seguinte forma (Markley F. L.; Crassidis, 2014):

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \arctan\left(\frac{x}{f}\right) \\ \arctan\left(\frac{y}{f}\right) \end{bmatrix} + \mathbf{v}_{sun} \tag{3.30}$$

onde α e β são ângulos medidos em relação aos eixos x e y do sensor, (x,y) são coordenadas do centroide no detector, f é a distância focal e $\mathbf{v}_{sun}$ é ruído branco gaussiano.

Com esses ângulos, é possível obter o vetor solar medido:

$$\mathbf{s}_s = \frac{1}{\sqrt{1+\tan^2\alpha+\tan^2\beta}} \begin{bmatrix} \tan\alpha \\ \tan\beta \\ 1 \end{bmatrix} \tag{3.31}$$

Este vetor segue o modelo vetorial genérico da Eq. 3.28, onde $\mathbf{z} = \mathbf{s}_s$ e $\mathbf{r}_i$ é a direção do Sol no referencial inercial.

3.3.3 TAM

Figura 3.3 – Magnetômetro de 3 eixos (SatCatalog, 2024)

Um magnetômetro tridimensional é um instrumento que realiza a medição da força e da orientação do campo magnético ao longo de três eixos que são perpendiculares entre si. No contexto da determinação da atitude de um satélite, esse dispositivo exerce um papel crucial ao identificar a orientação do satélite em relação ao campo magnético presente na Terra (Markley F. L.; Crassidis, 2014). Por meio das informações obtidas a partir dessas medições, é possível calcular a atitude do satélite, utilizando o conhecimento pré-existente sobre o comportamento do campo magnético na região da órbita em que o satélite se encontra posicionado. Essa capacidade de medição não apenas facilita, mas também permite a realização de ajustes precisos na orientação do satélite, otimizando seu alinhamento, especialmente quando combinado com outros sensores complementares.

O modelo matemático de um magnetômetro de três eixos é definido da seguinte forma (Wertz, 1978):

$$\mathbf{B}_m = \mathbf{C}_{sf}(\mathbf{I} + \mathbf{C}_{non-orth})\mathbf{B}_t + \mathbf{b} + \mathbf{v}_{mag} \tag{3.32}$$

onde $\mathbf{B}_m$ é o campo medido, $\mathbf{B}_t$ o campo verdadeiro, $\mathbf{C}_{sf} = \text{diag}(s_x, s_y, s_z)$ a matriz de fatores de escala, $\mathbf{C}_{non-orth}$ a matriz de não-ortogonalidade, $\mathbf{b}$ o bias e $\mathbf{v}_{mag}$ ruído gaussiano.

Entretanto, para utilização dentro de filtros de Kalman é utilizado um modelo simplificado, que despreza os erros de fator de escala e não ortogonalidades ou que compensa estes erros por meio de calibração:

$$\mathbf{B}_m = \mathbf{B}_t + \mathbf{b} + \mathbf{v}_{mag} \tag{3.33}$$

Onde o vetor de bias $\mathbf{b}$ é estimado pelo filtro de Kalman.

3.3.4 RIG

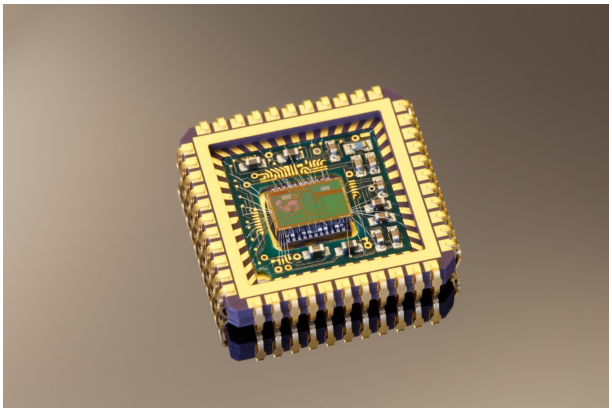


Figura 3.4 – Giroscópio do tipo Sistema Microeletromecânico (Fraunhofer, 2025)

Um giroscópio integrador de taxa é um dispositivo que mede a taxa de rotação e a integra ao longo do tempo para determinar a orientação relativa ou ângulo rotacional. É importante destacar que esse equipamento não fornece a orientação absoluta, mas sim a variação na atitude ao longo de um determinado intervalo de tempo (Markley F. L.; Crassidis, 2014). Esse instrumento é amplamente utilizado para identificar as mudanças na atitude ou na orientação do satélite em relação a um ou mais eixos. Os dados obtidos do giroscópio são frequentemente combinados com informações provenientes de outros sensores, como magnetômetros, com o intuito de fornecer uma estimativa precisa e confiável da atitude total do satélite.

A relação entre a velocidade angular verdadeira do satélite e a velocidade angular medida é dada da seguinte forma (Farrell, 2008):

$$\tilde{\omega} = (\mathbf{I} + \mathbf{S})\omega + \mathbf{b}_g + \mathbf{n}_g \tag{3.34}$$

Onde:

$\tilde{\omega}$ é o vetor de velocidade angular medido.

ω é o vetor de velocidade angular verdadeiro da espaçonave.

I é a matriz identidade.

S é a matriz de erro de fator de escala e acoplamento cruzado.

$\dot{b}_g$ é o vetor de bias do giroscópio. O bias é frequentemente modelado como uma combinação de um termo constante e um termo variável no tempo:

$$\dot{\mathbf{b}}_g = \mathbf{n}_{wg} \tag{3.35}$$

onde n_{wg} é um processo de ruído branco Gaussiano de média zero. Isso permite que o filtro de Kalman estime o bias variável no tempo.

3.4 Cenário de Missão

Este trabalho considera uma missão de reentrada atmosférica controlada de um CubeSat 12U equipado com um escudo térmico articulado do tipo guarda-chuva (*umbrella heat shield*). A missão é dividida em duas fases distintas, com abertura do escudo a 250 km de altitude, simulando a descida orbital desde 400 km até 120 km.

A missão foi dividida em duas fases com o objetivo de aproximá-la a uma situação mais real. A abertura do escudo foi decidida em 250 km pois é o momento logo antes de alguns fenômenos de reentrada começarem a acontecer. A simulação contém apenas os modelos físicos essenciais para a propagação orbital e determinação de atitude: densidade atmosférica (NRLMSISE-00(Picone *et al.*, 2002)), arrasto aerodinâmico (Vallado, 2013), e campo magnético terrestre (WMM-2010).

A Tabela 3.1 apresenta os parâmetros físicos do satélite para ambas as fases da missão:

Tabela 3.1 – Parâmetros físicos do CubeSat 12U

Parâmetro	**Fase 1**	**Fase 2**
Massa	24 kg	24 kg
Configuração	Escudo recolhido	Escudo aberto
Diâmetro do escudo	-	1 m
Área de arrasto	0.15 m^2	0.785 m^2
Coeficiente de arrasto	$C_D = 2.2$	$C_D = 1.9$
Razão A/m	0.00625 m^2/kg	0.03271 m^2/kg
Órbita inicial	400 km circular, equatorial	
Altitude alvo	120 km	

O alto coeficiente de arrasto do cubesat com escudo recolhido decorre da geometria aproximadamente cúbica com apêndices (antenas, painéis solares retraídos). Durante a fase com escudo aberto, o formato contuso em regime de escoamento molecular livre (típico de órbitas baixas) resulta em coeficiente ligeiramente menor. Apesar do menor C_D na fase 2, a área 5× maior (0,785 vs 0,15 m^2) aumenta significativamente a força de arrasto, acelerando o decaimento (Vallado, 2013).

3.5 Dinâmica Orbital com Arrasto Atmosférico

A propagação da órbita é realizada considerando a força gravitacional da Terra (modelo de dois corpos) e o arrasto atmosférico (Vallado, 2013).

3.5.1 Equações de Movimento

O movimento do satélite é governado por:

$$\ddot{\mathbf{r}} = \mathbf{a}_{\text{grav}} + \mathbf{a}_{\text{drag}} \tag{3.36}$$

A aceleração gravitacional (modelo de dois corpos) é:

$$\mathbf{a}_{\text{grav}} = -\frac{\mu}{r^3}\mathbf{r} \tag{3.37}$$

onde:

- $\mathbf{r}$ é o vetor posição (km, referencial inercial ECI)
- $r = \|\mathbf{r}\|$ é o raio orbital
- $\mu = 398600.64\ \text{km}^3/\text{s}^2$ é o parâmetro gravitacional da Terra

3.5.2 Modelo de Arrasto Atmosférico

A aceleração devido ao arrasto atmosférico é dada por (Vallado, 2013):

$$\mathbf{a}_{\text{drag}} = -\frac{1}{2}\frac{\rho C_D A}{m}\|\mathbf{v}_{\text{rel}}\|\mathbf{v}_{\text{rel}} \tag{3.38}$$

onde:

- ρ é a densidade atmosférica (kg/m^3) do modelo NRLMSISE-00
- $C_D = 1.9$ é o coeficiente de arrasto
- $A = 0.785\ \text{m}^2$ é a área da seção transversal
- $m = 24\ \text{kg}$ é a massa do satélite
- $\mathbf{v}_{\text{rel}}$ é a velocidade relativa à atmosfera rotativa (m/s)

A velocidade relativa leva em conta a rotação da Terra:

$$\mathbf{v}_{\text{rel}} = \mathbf{v}_{\text{sat}} - \boldsymbol{\omega}_{\oplus} \times \mathbf{r} \tag{3.39}$$

onde:

$$\boldsymbol{\omega}_{\oplus} = \begin{bmatrix} 0 \\ 0 \\ 7.2921159 \times 10^{-5} \end{bmatrix} \text{rad/s} \tag{3.40}$$

é o vetor de rotação da Terra (apontando para o pólo Norte).

O modelo atmosférico NRLMSISE-00 (Picone *et al.*, 2002) é utilizado para calcular a densidade atmosférica ρ em função da altitude, latitude, longitude, data, hora, e índices

de atividade solar (F10.7 e Ap). Este modelo empírico fornece a densidade total (neutra) da atmosfera terrestre na faixa de 0–1000 km, utilizado na fórmula acima para o cálculo da força de arrasto atmosférico.

A densidade atmosférica varia exponencialmente com a altitude segundo a equação:

$$\rho(h) = \rho_0 \exp\left(-\frac{h - h_0}{H}\right) \tag{3.41}$$

onde $H \approx 30$–60 km é a altura de escala atmosférica. Na faixa de altitude da missão (400–120 km), a densidade aumenta em aproximadamente 600,000 vezes, resultando em aceleração exponencial do decaimento orbital.

3.5.3 Integração Numérica (Runge-Kutta de Quarta Ordem)

A propagação orbital é realizada utilizando o método de Runge-Kutta de quarta ordem (RK4) (Burden; Faires; Burden, 2015; Vallado, 2013). Este método foi escolhido devido ao seu excelente equilíbrio entre precisão, estabilidade e custo computacional para problemas de dinâmica orbital (Curtis, 2013).

3.5.3.1 Formulação Matemática

Vetor de estado:

$$\mathbf{x} = \begin{bmatrix} \mathbf{r} \\ \dot{\mathbf{r}} \end{bmatrix} \in \mathbb{R}^6 \tag{3.42}$$

Derivada do estado:

$$\dot{\mathbf{x}} = f(\mathbf{x},t) = \begin{bmatrix} \dot{\mathbf{r}} \\ \ddot{\mathbf{r}} \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{r}} \\ -\frac{\mu}{r^3}\mathbf{r} + \mathbf{a}_{\text{drag}}(\mathbf{r},\dot{\mathbf{r}},t) \end{bmatrix} \tag{3.43}$$

Esquema de atualização RK4:

$$\mathbf{k}_1 = f(\mathbf{x}_n, t_n) \tag{3.44}$$

$$\mathbf{k}_2 = f\left(\mathbf{x}_n + \frac{\Delta t}{2}\mathbf{k}_1, t_n + \frac{\Delta t}{2}\right) \tag{3.45}$$

$$\mathbf{k}_3 = f\left(\mathbf{x}_n + \frac{\Delta t}{2}\mathbf{k}_2, t_n + \frac{\Delta t}{2}\right) \tag{3.46}$$

$$\mathbf{k}_4 = f(\mathbf{x}_n + \Delta t\mathbf{k}_3, t_n + \Delta t) \tag{3.47}$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{\Delta t}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4) \tag{3.48}$$

3.5.3.2 Parâmetros de Integração

O passo de tempo escolhido para a propagação orbital foi de 60 s, o que devido a órbita circular inicial de 400 km e o consequente período orbital de cerca de 92.7 minutos, permite obter aproximadamente 93 pontos por órbita, o que garante um erro de posição menor que 1 km ao longo da missão, mantendo o erro acumulado abaixo de 0.01% da magnitude da órbita (Vallado, 2013).

A escolha do passo de tempo é fundamentada em (Vallado, 2013):

- Para órbitas circulares em LEO (400 km), o período orbital é $T \approx 92.7$ min
- Com $\Delta t = 60$ s, obtemos aproximadamente 93 pontos por órbita
- Este passo garante erro de posição < 1 km ao longo da missão
- O erro relativo acumulado permanece abaixo de 0.01% da magnitude da órbita

Já a condição de término da simulação foi definida em 120 km, que marca o limite inferior da termosfera e o início da mesopausa, onde o arrasto atmosférico muito grande devido a densidade atmosférica e a reentrada final é iminente (Vallado, 2013).

3.5.4 Análise Teórica do Decaimento Orbital

Para uma órbita circular com densidade constante, a taxa de decaimento do semieixo maior é aproximadamente (Vallado, 2013):

$$\frac{da}{dt} \approx -\frac{2\pi\rho C_D A}{mT} a^2 \tag{3.49}$$

onde $T = 2\pi\sqrt{a^3/\mu}$ é o período orbital.

Para uma atmosfera exponencial $\rho = \rho_0 e^{-h/H}$ com altura de escala H (Vallado, 2013):

$$\frac{dh}{dt} \propto -\rho(h) \propto -e^{-h/H} \tag{3.50}$$

Isso resulta em **decaimento rápido em altitudes baixas** (realimentação exponencial: altitude menor → densidade maior → decaimento mais rápido → altitude menor...).

Dissipação de energia:

A energia mecânica total é (Vallado, 2013):

$$E = \frac{1}{2}v^2 - \frac{\mu}{r} \tag{3.51}$$

A taxa de perda de energia é (Vallado, 2013):

$$\frac{dE}{dt} = \mathbf{v} \cdot \mathbf{a}_{\text{drag}} < 0 \tag{3.52}$$

A taxa é sempre negativa (energia é dissipada como calor na atmosfera).

O modelo NRLMSISE-00 (Naval Research Laboratory Mass Spectrometer and Incoherent Scatter Radar Extended) é um modelo atmosférico empírico que fornece densidade de massa, temperatura e composição atmosférica em função da altitude, latitude, longitude, dia do ano, hora do dia e índices de atividade solar (Picone *et al.*, 2002).

3.5.5 Parâmetros de Atividade Solar

Os parâmetros fixos utilizados neste trabalho representam condições de atividade solar moderada:

$$F10.7_{\text{avg}} = 150 \text{ sfu} \quad \text{(média de 81 dias)} \tag{3.53}$$

$$F10.7_{\text{daily}} = 150 \text{ sfu} \quad \text{(fluxo diário)} \tag{3.54}$$

$$\text{Ap}_{\text{daily}} = 15 \quad \text{(índice geomagnético diário)} \tag{3.55}$$

onde 1 sfu = 10^{-22} W/(m^2 · Hz) é a unidade de fluxo solar.

Interpretação dos índices:

- F10.7 (fluxo de rádio solar em 10.7 cm):
  - 70–100: Mínimo solar
  - 100–150: Atividade moderada
  - 150–200: Sol ativo
  - $>$ 200: Máximo solar
- Ap (índice geomagnético):
  - 0–7: Calmo
  - 8–15: Instável
  - 16–29: Ativo
  - 30–49: Tempestade menor
  - $\geq$ 50: Tempestade maior

3.5.6 Variação da Densidade com Altitude

Utilizando este modelo atmosférico é possível obter a densidade atmosférica ao longo da missão planejada, de uma altitude de 400 km até 120 km.

Altura (km)	**Densidade (kg/m^3)**
400	5×10^{-12}
120	3×10^{-6}

Tabela 3.2 – Densidade atmosférica em atividade solar moderada (F10.7 = 150)(Picone *et al.*, 2002)

É possível notar que há um aumento de densidade considerável ao longo da missão de cerca de 6 ordens de magnitude. Este aumento exponencial é o que impulsiona o decaimento orbital rápido em baixas altitudes.

3.5.7 Aproximação por Altura de Escala

Na termosfera (> 90 km), a densidade segue aproximadamente:

$$\rho(h) \approx \rho_0 \exp\left(-\frac{h - h_0}{H}\right) \tag{3.56}$$

onde a **altura de escala** $H \approx$ 30–60 km (varia com temperatura e composição) (Vallado, 2013).

3.5.8 Conversão de Coordenadas

A posição no referencial inercial ECI deve ser convertida para o referencial fixo na Terra ECEF (Earth-Centered Earth-Fixed), e então para coordenadas geodésicas (latitude/-longitude).

3.5.8.1

Tempo sideral

$$\theta_{\text{sid}}(t) = \theta_0 + \omega_\oplus t \tag{3.57}$$

onde:

$$\theta_0 = 280.46061837 + 360.98564736628 \cdot d_{2000} \tag{3.58}$$

onde d_{2000} representa o número de dias Julianos desde J2000.0 (1 de janeiro de 2000, 12:00 TT), utilizado como época de referência para cálculos astronômicos.

3.5.8.2 Rotação ECI para ECEF

$$\mathbf{r}_{\text{ECEF}} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{r}_{\text{ECI}} \tag{3.59}$$

3.5.8.3 Conversão ECEF para geodésica

É utilizada solução iterativa das equações do elipsoide com parâmetros WGS-84 (National Imagery and Mapping Agency, 2000).

3.6 Modelo de Ruído dos Sensores

3.6.1 Ruído do Magnetômetro

O ruído do magnetômetro é modelado como ruído branco Gaussiano com desvio padrão constante:

$$\sigma_{\text{mag}} = 50 \text{ nT} \tag{3.60}$$

O valor de $\sigma_{\text{mag}} = 50$ nT foi escolhido por estar na faixa de valores de ruído típicos para magnetômetros espaciais utilizados em CubeSats (10–100 nT) (Wertz, 1978). Este valor é consistente com sensores comerciais como o MAG3110 (ruído típico de 25–50 nT) e HMC5883L (ruído típico de 20–100 nT)., além de ser o valor utilizado no código original de simulação do TRMM (Markley F. L.; Crassidis, 2014) que foi utilizado como base para a simulação realizada neste trabalho.

Para um CubeSat 12U equipado com sensores comerciais um magnetômetro com ruído de 50 nT é uma possibilidade existente na escolha de um sensor para uma missão real, uma vez que os magnetômetros utilizados em CubeSats modernos podem apresentar níveis de ruído em torno deste valor (Wertz, 1978).

3.6.2 Ruído do Giroscópio

O modelo de ruído de ruído usado para o giroscópio consiste de dois componentes, modelo padrão utilizado para giroscópios do tipo *RIG*, utilizados nessa simulação (Markley F. L.; Crassidis, 2014; Farrell, 2008): o *angle random walk* e o *random walk* do bias do giroscópio.

O *angle random walk*, como é chamado na literatura (Markley F. L.; Crassidis, 2014), é um ruído branco Gaussiano modelado como:

$$\eta_v \sim \mathcal{N}(0, \sigma_v^2) \tag{3.61}$$

onde $\sigma_v = 3.162 \times 10^{-7}$ rad/s ($\approx 1.09 \times 10^{-3}$ °/$\sqrt{\text{h}}$) é o coeficiente de Angle Random Walk. Ele representa o ruído que gera flutuações aleatórias instantâneas na medição.

Já o *random walk* do bias do giroscópio é modelado da seguinte forma (Markley F. L.; Crassidis, 2014):

$$\dot{\mathbf{b}}_g = \mathbf{n}_{wg} \tag{3.62}$$

onde $\mathbf{n}_{wg} \sim \mathcal{N}(0, \sigma_u^2 \mathbf{I})$ é um ruído branco Gaussiano com intensidade $\sigma_u = 3.162 \times 10^{-10}$ rad/s$^{1/2}$ ($\approx 1.09 \times 10^{-6}$ °/$\sqrt{\text{h}}$). Este representa a variação lenta do bias ao longo do tempo.

É possível notar que ambos os ruídos são Gaussianos e o que diferencia eles é a magnitude de cada coeficiente, uma diferença de mil vezes. Essa diferença de magnitude representa o fato de o ruído de medição instantâneo domina sobre a deriva lenta do bias em escalas de tempo curtas e é uma escolha específica herdada do código de simulação do TRMM (Markley F. L.; Crassidis, 2014). O bias inicial é $b_0 = 0.1$ °/h ($\approx 4.848 \times 10^{-7}$ rad/s).

Com esses coeficientes, o modelo de medição do giroscópio é dado por (Eq. 3.34):

$$\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega} + \boldsymbol{\eta}_v + \mathbf{b}_g \tag{3.63}$$

onde:

$\tilde{\boldsymbol{\omega}}$ Velocidade angular medida

$\boldsymbol{\omega}$ Velocidade angular verdadeira do satélite

$\boldsymbol{\eta}_v$ Ruído branco (ARW)

$\mathbf{b}_g$ Bias variável no tempo (random walk)

Os valores utilizados são consistentes com o que é proposto para simulações que utilizam giroscópios MEMS, que são comumente utilizados em CubeSats por causa de suas dimensões pequenas (Markley F. L.; Crassidis, 2014; Wertz, 1978). É importante destacar que a diferença de magnitude de mil vezes entre os dois coeficientes e o fato de ambos utilizarem o mesmo fator multiplicativo de 3.162 não têm motivo físico, mas foram uma escolha de projeto dos autores do código de simulação do TRMM e do livro no qual este trabalho se baseia (Markley F. L.; Crassidis, 2014). Apesar disso ambos os valores estão dentro da faixa típica para giroscópios MEMS (Wertz, 1978; Markley F. L.; Crassidis, 2014).

3.6.3 Justificativa do Modelo Simplificado

A escolha de utilizar ruído constante no modelo dos sensores se deu pelo objetivo central do trabalho, que é fazer uma comparação entre os diversos algoritmos de filtragem implementados neste trabalho. Adicionar mais realismo ao modelo dos sensores implicaria em fazer com que o ruído variasse no tempo, sendo influenciado pela altura e a temperatura do satélite. Isso aumentaria a complexidade do código de forma significativa, uma vez que a modelagem mais detalhadas dos fenômenos físicos que afetam os sensores teria que ser feita, aumentando ainda mais o escopo do trabalho. Além disso, é possível notar na literatura (Markley F. L.; Crassidis, 2014; Wertz, 1978; Shuster, 1993) que a utilização de ruído estacionário na modelagem dos sensores é uma abordagem amplamente utilizada quando se trata de comparação de algoritmos.

Outro motivo pelo qual o modelo simplificado foi escolhido se deve ao fato de que maior parte da missão se encontra em altitudes onde efeitos ambientais como temperatura e plasma ionosférico ainda não afetam de forma significativa os sensores (Wertz, 1978),

novamente aumentando a complexidade do código sem contribuir significantemente para a comparação entre os filtros.

3.6.4 Limitações

A escolha de utilizar o modelo de ruído constante acarreta em algumas limitações na simulação que devem ser levadas em consideração ao analisar os resultados do trabalho:

Efeitos ambientais não modelados: a variação de temperatura devido ao aquecimento aerodinâmico gerado pelo aumento da densidade atmosférica ao longo do decaimento, a interferência magnética do plasma presente na ionosfera e a própria degradação do sensor devido a ciclagem térmica e radiação são efeitos que mudam o ruído do sensor com o tempo, mas que não foram modelados neste trabalho devido a restrições de tempo e à complexidade necessária para implementar estes modelos de forma robusta suficiente para que a qualidade da simulação aumentasse, uma vez que uma implementação parcial desses fenômenos poderia diminuir a qualidade do trabalho, sendo preferido seguir a abordagem feita na literatura (Markley F. L.; Crassidis, 2014; Vallado, 2013; Wertz, 1978).

As conclusões sobre o desempenho de cada um dos filtros abordados neste trabalho são relativas somente ao cenário proposto em 3.4, em outras palavras, não é possível generalizar os resultados aqui obtidos para outros cenários, como no caso de ruídos não estacionários ou não Gaussianos. Ainda assim é possível verificar e validar algumas das vantagens fundamentais de cada filtro, como a não utilização de Jacobianos no EnSRF ou o mecanismo de adaptação de covariância de medição baseado em detecção de outliers no RAKF (Seção 4.4).

3.6.5 Injeção de *Outliers*

Para testar a robustez dos algoritmos de filtragem a erros de medição esporádicos e de grande magnitude, foi implementada uma estratégia de injeção de *outliers* na simulação, onde em 5% das medições, em um dado instante de tempo, o ruído aplicado é multiplicado por 10, ou seja, o ruído no magnetômetro naquele instante é de 500 nT.

Alguns dos motivos para implantação desta estratégia é simular situações onde pode haver interferência eletromagnética devido ao acionamento de rodas de reação ou magnetotorquers (Wertz, 1978), erros de conversão analógico-digital, uma vez que apesar de o sensor estar medindo um efeito de natureza contínua, ele precisa transformar esta leitura em informação digital para ser interpretada pelo computador de bordo, assim como efeitos causados pela alta radiação presente no ambiente seja vinda de tempestades solares ou devido a anomalia do Atlântico Sul.

Este teste é importante para avaliar como cada um dos filtros se comporta com dados incoerentes que podem aparecer durante o tempo de operação de um satélite. A escolha de 5% das medidas para receber estes ruídos condiz com cenários realistas para cubesats em órbita baixa, onde falhas de sensor e interferência ocorrem ocasionalmente, mas não são maioria no conjunto de dados (Wertz, 1978).

3.7 Modelo de Atitude Verdadeira

O satélite é modelado como seguindo uma atitude de apontamento ao nadir (*nadir-pointing*) (Wertz, 1978; Vallado, 2013), onde:

- $\hat{\mathbf{z}}_b$: Aponta para o centro da Terra (nadir)
- $\hat{\mathbf{y}}_b$: Ao longo da normal negativa da órbita (direção anti-momento)
- $\hat{\mathbf{x}}_b$: Completa o sistema destro (aproximadamente direção de velocidade)

3.7.1 Definição Matemática

$$\hat{\mathbf{z}}_b = -\frac{\mathbf{r}}{\|\mathbf{r}\|} \tag{3.64}$$

$$\hat{\mathbf{y}}_b = -\frac{\mathbf{r} \times \dot{\mathbf{r}}}{\|\mathbf{r} \times \dot{\mathbf{r}}\|} \tag{3.65}$$

$$\hat{\mathbf{x}}_b = \hat{\mathbf{y}}_b \times \hat{\mathbf{z}}_b \tag{3.66}$$

Matriz de cossenos diretores (DCM):

$$\mathbf{A}_{\text{ECI}\to\text{Body}} = \begin{bmatrix} \hat{\mathbf{x}}_b^T \\ \hat{\mathbf{y}}_b^T \\ \hat{\mathbf{z}}_b^T \end{bmatrix} \tag{3.67}$$

Extração de quatérnion:

O quatérnion $\mathbf{q}$ é extraído de $\mathbf{A}$ utilizando o método de Shepperd (Shepperd, 1978), que evita singularidades ao selecionar o componente com maior magnitude.

3.7.2 Velocidade Angular (Variante no Tempo)

Para um satélite apontando ao nadir em uma órbita em decaimento, a velocidade angular verdadeira é (Wertz, 1978):

$$\boldsymbol{\omega}_{\text{true}} = \begin{bmatrix} 0 \\ -n(t) \\ 0 \end{bmatrix} \tag{3.68}$$

onde $n(t)$ é a taxa orbital instantânea:

$$n(t) = \frac{\|\mathbf{h}(t)\|}{r(t)^2} \tag{3.69}$$

Momento angular:

$$\mathbf{h} = \mathbf{r} \times \dot{\mathbf{r}} \tag{3.70}$$

A Tabela 3.3 mostra a variação da taxa orbital.

Tabela 3.3 – Variação da taxa orbital ao longo da missão

Altitude (km)	n (rad/s)	Período (min)
400	0.001131	92.7
300	0.001163	90.2
200	0.001199	87.6
120	0.001235	84.8

Variação: ~9% de aumento na taxa angular ao longo da missão.

3.8 Validação e Justificativa Física

3.8.1 Validação do Decaimento Orbital

A energia mecânica total (Eq. 3.51) deve diminuir monotonicamente ao longo da simulação (Vallado, 2013) e a taxa de perda de energia (Eq. 3.52) deve ser sempre negativa, confirmando modelagem consistente do arrasto (Curtis, 2013). Isso deve fazer com que a taxa de decaimento do satélite aumente exponencialmente à medida que a altitude diminui (Vallado, 2013).

3.8.2 Validação do Modelo de Ruído

O modelo de ruído constante é validado por comparação direta com a literatura: a utilização de um ruído com valor constante é adequado para fazer uma analise de desempenho entre diferentes métodos de filtragem (Markley F. L.; Crassidis, 2014; Wertz, 1978) e o mesmo valor de ruído é utilizado no código da missão do satélite TRMM utilizado como base para este trabalho (Markley F. L.; Crassidis, 2014).

3.8.3 Validação dos Modelos de Sensores

O modelo magnético utilizado no código é o World Magnetic Model 2010 (Maus *et al.*, 2010), que fornece precisão de ~1–10 nT para o campo magnético da Terra em órbita LEO.

Essa precisão está bem abaixo do ruído de medição do magnetômetro de 50 nT. O modelo WMM foi desenvolvido pelo National Geophysical Data Center (NGDC) e é um dos modelos magnéticos mais utilizados para aplicações espaciais (Wertz, 1978).

O ruído aplicado no sensor solar, de $\sigma_{\text{sun}} = 0.05$ (0.87 mrad), foi retirado da literatura (Markley F. L.; Crassidis, 2014; Wertz, 1978), valor utilizado para reproduzir o ruído encontrado em sensores solares digitais mais grosseiros. Sensores solares de alta precisão existem e atingem uma precisão de até 0.001°, mas não são considerados neste trabalho.

Níveis de *random walk* e ruído branco são consistentes com giroscópios MEMS utilizados em pequenos satélites (Markley F. L.; Crassidis, 2014).

3.8.4 Comparação com o Exemplo Original TRMM

A cinemática de atitude e propagação de quatérnions, os modelos de medição de sensores (magnetômetro, sensor solar, giroscópio), o algoritmo QUEST e o filtro Alpha e a detecção de eclipse foram herdados do código exemplo do satélite TRMM (Markley F. L.; Crassidis, 2014).

O modelo de densidade atmosférica NRLMSISE-00 (Picone *et al.*, 2002), o modelo de arrasto (Vallado, 2013), propagação orbital RK4 foram implementados. Além disso, para os fins deste trabalho foram adicionados quatro filtros para comparação com os filtros herdados: o Filtro de Kalman Estendido, o Filtro de Kalman Adaptativo Baseado em Inovação, o Filtro de Kalman Ensemble e o Filtro de Kalman Raiz Quadrada de Ensemble. Estes serão apresentados na próxima seção.

4 Métodos de Estimação

4.1 QUEST

O QUEST resolve o problema de Wahba.

Grace Wahba propôs em 1965 (WAHBA, 1965) encontrar a matriz de rotação **A** (det $= +1$) que minimiza:

$$L(\mathbf{A}) = \frac{1}{2}\sum_{i=1}^{N} a_i \|\mathbf{b}_i - \mathbf{A}\mathbf{r}_i\|^2 \tag{4.1}$$

onde $L(\mathbf{A})$ é o erro entre vetores medidos $\mathbf{b}_i$ e referências rotacionadas $\mathbf{A}\mathbf{r}_i$, com pesos a_i para N vetores.

Fazendo uso da ortogonalidade da matriz A e a expansão da norma dos vetores unitários é possível reescrever a função de perda de outra forma (MARKLEY, 1993):

$$L(\mathbf{A}) = \lambda_0 - \mathrm{tr}(AB^T) \tag{4.2}$$

onde:

$$\lambda_0 = \sum_{i=1}^{n} a_i, \tag{4.3}$$

$$B = \sum_{i=1}^{n} a_i b_i r_i^T \tag{4.4}$$

Desta forma, o problema vira um problema de maximização de $\mathrm{tr}(AB^T)$. Existem duas abordagens: cálculo direto da matriz ou via quatérnions. O método conhecido como QUEST está incluído na segunda categoria, no qual a solução é derivada por meio da representação da matriz de atitude utilizando quatérnions (SHUSTER M. D.; OH, 1981).

Dessa forma, a solução para o problema de Wahba pode ser reescrita em termos de quatérnions e da matriz $K(B)$, conhecida como matriz de Davenport (DAVENPORT, 1968):

$$L(A(\mathbf{q})) = \lambda_0 - \mathbf{q}^T K(B)\mathbf{q} \tag{4.5}$$

onde:

$$K(B) = \sum_{i=1}^{N} a_i \left[\mathbf{b_i}\otimes\right]^T \left[\mathbf{r}_i\odot\right] = \begin{bmatrix} B + B^T - (\mathrm{tr}B)I_3 & \mathbf{z} \\ \mathbf{z}^T & \mathrm{tr}B \end{bmatrix} \tag{4.6}$$

$$\mathbf{z} \equiv \begin{bmatrix} B_{23} - B_{32} \\ B_{31} - B_{13} \\ B_{12} - B_{21} \end{bmatrix} = \sum_{i=1}^{N} a_i(\mathbf{b}_i \times \mathbf{r}_i) \tag{4.7}$$

Portanto, a solução é alcançada mediante a maximização do termo $\mathbf{q}^T K(B)\mathbf{q}$. O quatérnion ótimo tem que obedecer a seguinte restrição de norma unitária (SHUSTER M. D.; OH, 1981): $q^T q = 1$. Desta forma adicionamos um multiplicador de Lagrange para impor essa restrição de norma unitária:

$$\mathcal{L}(q,\lambda) = q^T K(B) q - \lambda(q^T q - 1) \tag{4.8}$$

Então derivamos em relação a q e igualamos a zero, obtendo a seguinte equação, que é um problema de autovalor, onde λ é um dos autovalores de $K(B)$ e q o autovetor correspondente:

$$K(B)q = \lambda q \tag{4.9}$$

Sabendo que o nosso objetivo é maximizar $q^T K(B)q$, é possível notar que a solução ótima para 4.9 é o maior λ de $K(B)$ (SHUSTER M. D.; OH, 1981).

Temos então que a solução ótima para a função de perda 4.5 se dá por (SHUSTER M. D.; OH, 1981):

$$L_{otimo} = \lambda_0 - \lambda_{max} \tag{4.10}$$

4.2 Filtro Alfa

O filtro alfa é um estimador recursivo de primeira ordem para atitude — caso particular do filtro alfa-beta (Benedict; Bordner, 1962) quando a velocidade angular vem diretamente do giroscópio (Junkins, 2011).

Sua principal aplicação consiste em integrar a leitura de sensores que apresentam ruído, com o objetivo de produzir uma estimativa suavizada e mais precisa (Simon, 2006). Para facilitar a compreensão do conceito, serão utilizados como exemplos os sensores que foram apresentados na seção 3.3. Serão consideradas as leituras do giroscópio para realizar a propagação da atitude e, adicionalmente, as leituras do magnetômetro e do sensor solar digital para efetuar a correção, aprimorando a precisão da estimativa. O processo se desenrola em duas vertentes principais: uma de predição baseada no modelo dinâmico e outra de correção baseada nas medições instantâneas.

Primeiramente, a atitude estimada no instante anterior é propagada para o instante atual utilizando a velocidade angular medida pelo giroscópio. Esta etapa de predição, que

resulta em $\mathbf{q}_{\text{prop}}$, representa a melhor estimativa da atitude futura baseada apenas no conhecimento do movimento do corpo. A propagação é realizada utilizando a forma discreta exata da cinemática de quatérnions, assumindo velocidade angular constante durante o intervalo Δt:

$$\mathbf{q}_{\text{prop}} = \mathbf{\Phi}(\boldsymbol{\omega}_{k-1}, \Delta t)\mathbf{q}_{k-1} \tag{4.11}$$

onde $\mathbf{\Phi}(\boldsymbol{\omega}, \Delta t)$ é a matriz de transição de estado do quatérnion definida na Eq. 3.25. Esta é a forma discreta exata da cinemática de quatérnions assumindo velocidade angular constante durante o intervalo Δt (Markley F. L.; Crassidis, 2014).

Embora métodos mais simples como a integração de Euler sejam frequentemente citados para filtros alfa devido à sua simplicidade computacional (Simon, 2006), a forma exata é preferível pois garante que o quatérnion propagado mantenha norma unitária sem precisar de um passo extra para realizar a normalização, além de eliminar o erro de truncamento $O(\Delta t^2)$ da aproximação de Euler. Para o passo de tempo utilizado ($\Delta t = 60$ s) e as velocidades angulares da missão ($\|\boldsymbol{\omega}\| \sim 10^{-3}$ rad/s, Tabela 3.3), a diferença entre os métodos seria pequena, mas a forma exata é matematicamente mais rigorosa.

Em paralelo, as medições vetoriais do magnetômetro e do sensor solar digital são utilizadas para calcular uma estimativa instantânea da atitude, aqui denominada $\mathbf{q}_{\text{meas}}$. Este quatérnion de medição representa a orientação do satélite no instante atual, conforme inferido pelos sensores (Wertz, 1978). Ele pode ser obtido utilizando o método QUEST como descrito na seção 4.1), que é a forma utilizada neste trabalho:

$$\mathbf{q}_{\text{meas}} = \text{QUEST}(\mathbf{b}_{\text{mag}}, \mathbf{b}_{\text{sun1}}, \mathbf{b}_{\text{sun2}}, \mathbf{r}_{\text{mag}}, \mathbf{r}_{\text{sun}}) \tag{4.12}$$

onde $\mathbf{b}_i$ são os vetores medidos no referencial do corpo e $\mathbf{r}_i$ são os vetores de referência no referencial inercial. Como demonstrado na Seção 4.1, o uso de múltiplos vetores é necessário pois um único vetor fornece apenas 2 graus de liberdade (Wertz, 1978), enquanto a determinação completa da atitude requer 3 graus de liberdade.

Com o quatérnion propagado ($\mathbf{q}_{\text{prop}}$) e o quatérnion medido ($\mathbf{q}_{\text{meas}}$) em mãos, a equação do filtro alfa é utilizada para fundir as duas informações. A fusão resulta em uma estimativa temporária, $\mathbf{q}_{\text{temp}}$, que é uma média ponderada entre a predição e a medição (Junkins, 2011):

$$\mathbf{q}_{\text{temp}} = (1 - \alpha)\mathbf{q}_{\text{prop}} + \alpha\mathbf{q}_{\text{meas}} \tag{4.13}$$

É crucial ressaltar que a equação 4.13 é uma interpolação linear, uma simplificação matemática para quatérnions. Como os quatérnions que representam rotações residem em uma hiperesfera quadridimensional (um espaço curvo), uma média ponderada linear não

garante que o resultado $\mathbf{q}_{\text{temp}}$ mantenha a norma unitária, condição indispensável para um quatérnion de atitude (Markley F. L.; Crassidis, 2014). A interpolação "correta"no espaço de rotações seria a Interpolação Linear Esférica (SLERP) (Shoemake, 1985), porém, sua complexidade computacional é maior e foge ao escopo de um filtro simples como o filtro alfa.

O valor do ganho α estabelece o equilíbrio crucial entre a confiança na predição do modelo e na nova medição, podendo variar entre 0 e 1. Um α que se encontra mais próximo de 0 diminui o ruído, mas resulta em uma estimativa cuja alteração é mais lenta, uma vez que concede maior peso à estimativa propagada. Por outro lado, um valor que se aproxime de 1 proporciona uma resposta mais ágil às mudanças nas leituras, mas, em contrapartida, também introduz mais flutuações e uma suavização menor da estimativa (Junkins, 2011) (Kalata, 1994).

Neste trabalho, o valor base utilizado é $\alpha_0 = 0.1$ (Markley F. L.; Crassidis, 2014), que proporciona um equilíbrio adequado entre a suavização do ruído e a capacidade de resposta às mudanças de atitude. Este valor está dentro da faixa típica de $0.1 \leq \alpha \leq 0.3$ utilizada em aplicações de estimação de atitude de satélites (Lefferts; Markley; Shuster, 1982). O valor de α varia com base na geometria dos sensores:

$$\alpha(k) = \alpha_0 \times \left\| \frac{\mathbf{b}_{\text{mag}}}{\|\mathbf{b}_{\text{mag}}\|} \times \frac{\mathbf{s}_{\text{sun}}}{\|\mathbf{s}_{\text{sun}}\|} \right\|^2 \tag{4.14}$$

onde $\mathbf{b}_{\text{mag}}$ e $\mathbf{s}_{\text{sun}}$ são os vetores de campo magnético e direção solar no referencial inercial. O termo $\|\mathbf{b}_{\text{mag}} \times \mathbf{s}_{\text{sun}}\|$ representa o seno do ângulo entre os vetores, atingindo valor máximo 1 quando os vetores são perpendiculares e mínimo 0 quando paralelos. Este ajuste geométrico é baseado no fato de que a observabilidade da atitude é máxima quando os vetores de medição são ortogonais, fornecendo informação independente sobre dois eixos de rotação (Lefferts; Markley; Shuster, 1982). Quando os vetores são paralelos ($\|\mathbf{b}_{\text{mag}} \times \mathbf{s}_{\text{sun}}\| = 0$), um grau de liberdade da atitude não é observável, e o filtro deve confiar mais na propagação do giroscópio, o que resulta em uma diminuição do valor de α. Quando perpendiculares, ambos os eixos fornecem informação máxima, justificando maior peso na medição, aumentando α.

A interpolação linear viola a norma unitária. Normalização restaura $\|\mathbf{q}\| = 1$ (Markley F. L.; Crassidis, 2014):

$$\mathbf{q}_k = \frac{\mathbf{q}_{\text{temp}}}{\|\mathbf{q}_{\text{temp}}\|} \tag{4.15}$$

4.3 Filtro de Kalman Estendido

O filtro de Kalman estendido representa a abordagem mais utilizada na estimação de sistemas não lineares, que é exatamente o caso quando se trata do desafio de determinar

a atitude de satélites. Essa escolha é justificada pelo fato de o filtro de Kalman clássico ter sido projetado para resolver problemas que envolvem apenas sistemas lineares (Markley F. L.; Crassidis, 2014). A dificuldade em sistemas não-lineares, como a cinemática de atitude, está na na propagação da matriz de covariância do erro, uma vez que uma distribuição de probabilidade, geralmente Gaussiana, ao ser propagada por uma função não-linear, muitas vezes não resulta em uma distribuição que ainda seja Gaussiana. O EKF contorna esse problema por meio da linearização do sistema em torno da estimativa de estado atual, utilizando uma expansão de série de Taylor de primeira ordem para obter as matrizes Jacobianas necessárias para a propagação da covariância.

Para estados que possuem restrições matemáticas, como o quatérnion de atitude que deve manter sua norma unitária ($\|\mathbf{q}\| = 1$), a aplicação do EKF requer uma consideração especial. Uma abordagem, conhecida como EKF aditivo, trata o quatérnion como um vetor de quatro elementos e aplica uma correção aditiva. Contudo, essa soma geralmente não mantém a norma unitária, precisando de um passo extra de normalização forçada ao final da atualização. Este passo, embora funcional, é uma projeção geométrica que não possui uma base estatística ótima e pode introduzir inconsistências no filtro.

Uma outra abordagem, chamada de EKF Multiplicativo ou MEKF, que é a versão utilizada no código deste trabalho, foi desenvolvido especificamente para contornar esse problema. Em vez de estimar um erro nos quatro componentes do quatérnion, o MEKF define o erro de atitude como um vetor de rotação infinitesimal de três elementos, $\delta\boldsymbol{\alpha}$, que pertence ao espaço euclidiano e não possui restrições. A correção da atitude é então realizada "multiplicando"o quatérnion estimado por um quatérnion de correção derivado desse vetor de erro. Essa metodologia respeita a geometria do grupo de rotações SO(3), preserva a norma unitária do quatérnion naturalmente e é considerada mais elegante e robusta para a estimação de atitude (Junkins, 2011).

4.3.1 EKF Multiplicativo (MEKF)

A utilização do filtro de Kalman estendido multiplicativo pode ser resumida nos seguintes passos (Lefferts; Markley; Shuster, 1982; Markley F. L.; Crassidis, 2014; Cordeiro, 2012):

4.3.1.1 Inicialização

Neste passo é definido o estado inicial do satélite, composto pelo quatérnion de atitude e pelo bias do giroscópio, e a matriz de covariância do erro, que quantifica a incerteza inicial.

-

- Vetor de estado estimado[1]: $\mathbf{x}_0 = [\mathbf{q}_0^T, \mathbf{b}_0^T]^T$
- Vetor de estado de erro:

$$\bar{\mathbf{x}} = \begin{bmatrix} \delta \boldsymbol{p} \\ \Delta \mathbf{b} \end{bmatrix} \tag{4.16}$$

  onde $\delta \boldsymbol{p} \in \mathbb{R}^3$ é o vetor de erro de atitude (parte vetorial do quatérnion de erro) e $\Delta \mathbf{b} \in \mathbb{R}^3$ é o erro de bias do giroscópio.

  O MEKF trabalha com um vetor de estado de erro reduzido de dimensão 6 ao invés de utilizar os 4 componentes do quatérnion de erro uma vez que devido à restrição de norma unitária, apenas 3 parâmetros independentes são necessários para representar o erro de atitude (Lefferts; Markley; Shuster, 1982).

- Matriz de covariância do erro (6×6): $\mathbf{P}_0^+ = E[\delta \mathbf{x}_0 \delta \mathbf{x}_0^T]$. Ela detalha a incerteza na atitude e no bias.

$$\mathbf{P}_0^+ = \begin{bmatrix} \mathbf{P}_{\alpha\alpha} & \mathbf{P}_{\alpha b} \\ \mathbf{P}_{b\alpha} & \mathbf{P}_{bb} \end{bmatrix}_0 \tag{4.17}$$

  onde $\mathbf{P}_{\alpha\alpha}$ é a covariância do erro de atitude (3×3), $\mathbf{P}_{bb}$ é a covariância do erro de bias (3×3), e $\mathbf{P}_{\alpha b}$ é a covariância cruzada.

- Matriz de covariância do ruído do processo (6×6): $\mathbf{Q}$.
- Matriz de covariância do ruído de medição ($m \times m$): $\mathbf{R}$.

4.3.1.2 Propagação

Nesta fase, o estado e sua incerteza são propagados para o próximo instante de tempo.

- Propagação do Estado Estimado: O quatérnion e o bias são propagados utilizando as equações cinemáticas e dinâmicas não-lineares. Esta propagação utiliza a velocidade angular medida, $\boldsymbol{\omega}_m$, corrigida pelo bias estimado, $\hat{\mathbf{b}}_{k-1}^+$, resultando na velocidade angular estimada $\hat{\boldsymbol{\omega}} = \boldsymbol{\omega}_m - \hat{\mathbf{b}}_{k-1}^+$.

$$\dot{\hat{\mathbf{q}}}(t) = \frac{1}{2}\boldsymbol{\Omega}(\hat{\boldsymbol{\omega}})\hat{\mathbf{q}}(t) \quad , \quad \dot{\hat{\mathbf{b}}}(t) = \mathbf{0} \tag{4.18}$$

  A integração dessas equações de t_{k-1} a t_k fornece o estado predito $\hat{\mathbf{x}}_k^- = [\hat{\mathbf{q}}_k^{-T}, \hat{\mathbf{b}}_k^{-T}]^T$.

- Propagação da Covariância do Erro: A matriz de covariância do erro é propagada através de um modelo linearizado. Este passo descreve como a incerteza do estado evolui devido à dinâmica do sistema e à adição de ruído de processo.

$$\mathbf{P}_k^- = \boldsymbol{\Phi}_{k-1}\mathbf{P}_{k-1}^+\boldsymbol{\Phi}_{k-1}^T + \mathbf{Q}_{k-1} \tag{4.19}$$

[1] Também denominado "estado nominal" em algumas referências, para distingui-lo do "estado de erro" que é resetado após cada atualização.

onde $\mathbf{\Phi}_{k-1}$ é a matriz de transição de estados do erro, obtida pela discretização da matriz de dinâmica do erro contínua, $\mathbf{F}(t)$:

$$\mathbf{F}(t) = \begin{bmatrix} -[\hat{\boldsymbol{\omega}}_\times] & -\mathbf{I}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \tag{4.20}$$

onde $[\hat{\boldsymbol{\omega}}_\times]$ é a matriz antissimétrica da velocidade angular estimada (Lefferts; Markley; Shuster, 1982).

A relação entre a matriz de transição de estados do erro completo (7 elementos) e a matriz de transição do erro reduzido (6 elementos) é dada pela matriz de transformação $\mathbf{S}^e(\hat{\mathbf{q}})$ (Lefferts; Markley; Shuster, 1982):

$$\bar{\mathbf{\Phi}}_{k-1} = \mathbf{S}^e(\hat{\mathbf{q}}_{k|k-1})\mathbf{\Phi}_{k-1}\mathbf{S}(\hat{\mathbf{q}}_{k-1|k-1}) \tag{4.21}$$

onde $\mathbf{S}^e$ extrai os componentes relevantes do estado completo para formar o estado reduzido. Na prática, essa transformação é implícita quando se trabalha diretamente com a representação de erro de 6 elementos.

4.3.1.3 Atualização

Quando uma nova medição $\mathbf{z}_k$ está disponível, ela é usada para corrigir o estado propagado.

- Cálculo da Jacobiana da Medição: A matriz Jacobiana da medição, $\mathbf{H}_k$, é calculada. Ela lineariza a função de medição em torno do estado predito e estabelece a relação entre o vetor de erro ($\delta\mathbf{x}$) e o residual da medição.
- Cálculo do Ganho de Kalman: O ganho de Kalman $\mathbf{K}_k$ é computado. Ele funciona como um peso ótimo que equilibra a confiança na predição do modelo (representada por $\mathbf{P}_k^-$) com a confiança na nova medição (representada por $\mathbf{R}_k$).

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k)^{-1} \tag{4.22}$$

Esta é a forma padrão do ganho de Kalman, que minimiza o erro médio quadrático da estimativa (Markley F. L.; Crassidis, 2014).

- Atualização do Estado de Erro: O ganho é usado para calcular a estimativa ótima do estado de erro com base na inovação (o residual entre a medição real e a predita).

$$\delta\hat{\mathbf{x}}_k = \begin{bmatrix} \delta\hat{\boldsymbol{\alpha}}_k \\ \delta\hat{\mathbf{b}}_k \end{bmatrix} = \mathbf{K}_k(\mathbf{z}_k - h(\hat{\mathbf{x}}_k^-)) \tag{4.23}$$

- Atualização da Covariância do Erro: A incerteza do estado de erro é reduzida, refletindo a informação ganha com a medição. A forma de Joseph é utilizada para garantir

estabilidade numérica e preservar a simetria positiva-definida da matriz (Markley F. L.; Crassidis, 2014):

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)^T + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T \tag{4.24}$$

A forma simplificada $\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-$ é matematicamente equivalente quando o ganho de Kalman é ótimo, mas a forma de Joseph é mais robusta a erros de arredondamento numérico.

4.3.1.4 Reset e normalização

Finalmente, a informação do estado de erro estimado é usada para corrigir o estado estimado. Este passo de reset é característico do MEKF e garante que a covariância do erro de atitude permaneça pequena (Lefferts; Markley; Shuster, 1982).

- Correção Multiplicativa da Atitude: O quatérnion predito é "rotacionado"pela pequena correção de atitude $\delta\hat{\boldsymbol{\alpha}}_k$. Primeiro, a magnitude do vetor de erro é calculada:

$$\delta\gamma = \|\delta\hat{\boldsymbol{\alpha}}_k\| \tag{4.25}$$

Em seguida, o vetor de erro é convertido em um quatérnion de erro usando a forma exata (Lefferts; Markley; Shuster, 1982):

$$\delta\mathbf{q}(\delta\hat{\boldsymbol{\alpha}}_k) = \begin{bmatrix} \frac{\sin(\delta\gamma/2)}{\delta\gamma}\delta\hat{\boldsymbol{\alpha}}_k \\ \cos(\delta\gamma/2) \end{bmatrix} \tag{4.26}$$

Para erros pequenos ($\delta\gamma \ll 1$), a aproximação de primeira ordem simplifica para:

$$\delta\mathbf{q}(\delta\hat{\boldsymbol{\alpha}}_k) \approx \begin{bmatrix} \frac{1}{2}\delta\hat{\boldsymbol{\alpha}}_k \\ 1 \end{bmatrix} \tag{4.27}$$

Finalmente, a multiplicação de quatérnions é realizada para aplicar a correção:

$$\hat{\mathbf{q}}_k^+ = \delta\mathbf{q}(\delta\hat{\boldsymbol{\alpha}}_k) \otimes \hat{\mathbf{q}}_k^- \tag{4.28}$$

- Correção Aditiva do Bias: O bias do giroscópio, que reside no espaço euclidiano, é corrigido de forma aditiva.

$$\hat{\mathbf{b}}_k^+ = \hat{\mathbf{b}}_k^- + \delta\hat{\mathbf{b}}_k \tag{4.29}$$

- Reset do Erro: O estado de erro é então resetado para zero ($\delta\mathbf{x} \rightarrow \mathbf{0}$), pois sua informação foi totalmente "absorvida"pelo estado estimado. A covariância atualizada, $\mathbf{P}_k^+$, é então usada como a incerteza inicial para o próximo ciclo de predição.

4.4 Robust Adaptive Kalman Filter

O Robust Adaptive Kalman Filter (RAKF) estende o EKF convencional com mecanismos de adaptação da covariância de medição baseados em detecção de outliers e avaliação de qualidade geométrica dos sensores (Mohamed; Schwarz, 1999; Hide; Moore; Smith, 2003). O objetivo é melhorar a robustez do filtro a medições anômalas sem descartar completamente informação utilizável, ao contrário de estratégias de rejeição binária de outliers (Yang; He; Xu, 2001; Karlgaard; Schaub, 2007).

4.4.1 Fundamentos da Adaptação de Covariância

O RAKF mantém a estrutura do MEKF apresentado na Seção 4.3, mas modifica dinamicamente a matriz de covariância de medição $\mathbf{R}_k$ com base em dois critérios complementares (Hide; Moore; Smith, 2003; Markley F. L.; Crassidis, 2014):

- Distância de Mahalanobis: Métrica estatística que detecta medições inconsistentes com o modelo de predição, amplamente utilizada em detecção de outliers (Maesschalck; Jouan-Rimbaud; Massart, 2000; Koch; Yang, 1998)
- Qualidade geométrica sensor-vetor: Avaliação da observabilidade baseada na ortogonalidade entre vetores medidos (magnetômetro e sensor solar), crítica para determinação precisa de atitude (Markley, 2003; SHUSTER M. D.; OH, 1981)

Esta abordagem dupla permite que o filtro adapte sua confiança nas medições considerando tanto inconsistências estatísticas quanto limitações geométricas da configuração de sensores (Karlgaard; Schaub, 2007; Markley F. L.; Crassidis, 2014).

4.4.2 Detecção de Outliers via Distância de Mahalanobis

A distância de Mahalanobis normaliza a inovação (residual entre medição e predição) pela covariância predita, fornecendo uma métrica adimensional de consistência (Maesschalck; Jouan-Rimbaud; Massart, 2000; Maybeck, 1979):

$$d_M = \sqrt{\mathbf{z}_k^T \mathbf{S}_k^{-1} \mathbf{z}_k} \tag{4.30}$$

onde $\mathbf{z}_k = \mathbf{y}_k - h(\hat{\mathbf{x}}_k^-)$ é a inovação e $\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k$ é a covariância da inovação (Kalman, 1960; Brown; Hwang, 1992).

Sob a hipótese de que o modelo está correto e os ruídos são gaussianos, d_M^2 segue uma distribuição qui-quadrado com n_z graus de liberdade (dimensão do vetor de medição) (Bar-Shalom; Li; Kirubarajan, 2001; Maybeck, 1979). Para medições vetoriais de 3 componentes, um threshold é estabelecido baseado no percentil 95% da distribuição χ_3^2 (Hide; Moore; Smith, 2003; Koch; Yang, 1998):

$$\chi^2_{\text{threshold}} = 5.0 \quad \text{(correspondente a 95\% de confiança)} \tag{4.31}$$

Quando $d_M > \chi^2_{\text{threshold}}$, a medição é considerada anômala e a covariância $\mathbf{R}_k$ é adaptada para reduzir seu peso no ganho de Kalman (Mohamed; Schwarz, 1999; Yang; He; Xu, 2001). Esta estratégia de adaptação suave é preferível à rejeição completa de medições, pois outliers podem conter informação parcial utilizável (Karlgaard; Schaub, 2007; Chang, 2012).

4.4.3 Qualidade Geométrica Sensor-Vetor

A observabilidade completa da atitude requer vetores de medição não-colineares (Markley, 2003; Lefferts; Markley; Shuster, 1982). Quando magnetômetro e sensor solar fornecem direções quase paralelas, a determinação do eixo perpendicular ao plano formado por ambos sofre degradação significativa (SHUSTER M. D.; OH, 1981; WAHBA, 1965). A qualidade geométrica é quantificada pelo produto vetorial normalizado entre o campo magnético terrestre e a direção solar (Pittelkau, 2003):

$$\gamma_{\text{geom}} = \left\| \frac{\mathbf{b}_{\text{mag}}}{\|\mathbf{b}_{\text{mag}}\|} \times \frac{\mathbf{s}_{\text{sun}}}{\|\mathbf{s}_{\text{sun}}\|} \right\| \tag{4.32}$$

onde $\gamma_{\text{geom}} \in [0, 1]$. Valores próximos a 1 indicam vetores perpendiculares (geometria ótima), enquanto valores próximos a 0 indicam vetores quase paralelos (perda de observabilidade em um eixo) (Markley, 2003). O threshold de geometria ruim é estabelecido empiricamente em (Hide; Moore; Smith, 2003; Markley F. L.; Crassidis, 2014):

$$q_{\text{threshold}} = 0.15 \tag{4.33}$$

Esta métrica é particularmente relevante para satélites em órbitas baixas, onde ocorrem configurações desfavoráveis durante travessias de eclipse ou em latitudes magnéticas específicas (Wertz, 1978; Pittelkau, 2003).

4.4.4 Fator de Escalonamento Adaptativo

A matriz de covariância de medição é escalonada por um fator $\lambda_k \geq 1.0$ que modula a confiança do filtro na medição atual (Hide; Moore; Smith, 2003; Mohamed; Schwarz, 1999):

$$\tilde{\mathbf{R}}_k = \lambda_k \mathbf{R}_k \tag{4.34}$$

Este escalonamento reduz o ganho de Kalman efetivo quando medições suspeitas ou condições geométricas desfavoráveis são detectadas (Karlgaard; Schaub, 2007; Yang; He; Xu, 2001). O fator λ_k é determinado pela lógica de adaptação multimodal (Chang, 2012):

$$\lambda_k = \begin{cases} 2.0 + 3.0 \times \min\left(1.0, \dfrac{d_M - \chi^2_{\text{threshold}}}{\chi^2_{\text{threshold}}}\right) & \text{se } d_M > \chi^2_{\text{threshold}} \\ 1.2 + 0.8 \times \dfrac{d_M - 0.8\chi^2_{\text{threshold}}}{0.2\chi^2_{\text{threshold}}} & \text{se } d_M > 0.8\chi^2_{\text{threshold}} \\ \max\left(\lambda_{\text{geom}}, 1.0\right) & \text{caso contrário} \end{cases} \tag{4.35}$$

onde o fator geométrico é calculado como (Pittelkau, 2003):

$$\lambda_{\text{geom}} = \begin{cases} 1.2 + 1.3\left(1.0 - \dfrac{\gamma_{\text{geom}}}{q_{\text{threshold}}}\right) & \text{se } \gamma_{\text{geom}} < q_{\text{threshold}} \\ 1.0 & \text{caso contrário} \end{cases} \tag{4.36}$$

Esta formulação implementa três regimes de adaptação hierárquicos, detalhados nas subseções seguintes.

4.4.4.1 Regime de Outlier Confirmado

Quando a distância de Mahalanobis excede significativamente o threshold ($d_M > 5.0$), a medição é classificada como outlier confirmado (Hide; Moore; Smith, 2003; Koch; Yang, 1998). Neste regime, o fator de escalonamento varia de 2.0 a 5.0 proporcionalmente ao excesso de distância de Mahalanobis (Mohamed; Schwarz, 1999):

$$\lambda_k = 2.0 + 3.0 \times \min\left(1.0, \frac{d_M - 5.0}{5.0}\right) \tag{4.37}$$

Este comportamento progressivo evita transições abruptas no ganho de Kalman, preservando estabilidade numérica (Chang, 2012; Bar-Shalom; Li; Kirubarajan, 2001). Para outliers extremos ($d_M \gg 10$), o fator satura em 5.0, reduzindo a covariância efetiva de medição em uma ordem de magnitude e essencialmente descartando a medição sem introduzir descontinuidades no filtro (Karlgaard; Schaub, 2007).

O crescimento não-linear reflete a interpretação probabilística: medições com $d_M = 7.5$ (1.5× o threshold) têm probabilidade desprezível ($< 0.1\%$) sob a hipótese nula de ruído gaussiano (Maesschalck; Jouan-Rimbaud; Massart, 2000; Maybeck, 1979). A adaptação suave permite recuperação gradual quando medições subsequentes retornam à normalidade, evitando divergência do filtro (Yang; He; Xu, 2001).

4.4.4.2 Regime de Medição Suspeita

Para distâncias de Mahalanobis moderadamente elevadas ($4.0 < d_M \leq 5.0$), a medição é tratada com cautela moderada através de crescimento linear suave de λ_k de 1.2 a 2.0 (Hide; Moore; Smith, 2003):

$$\lambda_k = 1.2 + 0.8 \times \frac{d_M - 4.0}{1.0} \tag{4.38}$$

Este regime intermediário reconhece que medições no intervalo $[4.0, 5.0]$ têm probabilidade marginal sob distribuição qui-quadrado (5-10%), podendo representar tanto outliers leves quanto flutuações estatísticas naturais (Bar-Shalom; Li; Kirubarajan, 2001). A adaptação conservadora (máximo de 2×) preserva informação útil enquanto limita impacto de anomalias transitórias (Chang, 2012; Karlgaard; Schaub, 2007).

A transição contínua entre regimes evita oscilações no ganho de Kalman quando medições flutuam próximo aos thresholds, fenômeno conhecido como "chattering"em sistemas de controle adaptativo (Åström; Wittenmark, 1995). A linearidade do fator neste intervalo simplifica análise de estabilidade e permite ajuste empírico baseado em dados de missão (Wertz, 1978).

4.4.4.3 Regime de Geometria Degradada

Quando a qualidade geométrica é insuficiente ($\gamma_{\text{geom}} < 0.15$), independentemente da distância de Mahalanobis, o fator de escalonamento aumenta inversamente proporcional à qualidade (Pittelkau, 2003; Markley, 2003):

$$\lambda_{\text{geom}} = 1.2 + 1.3 \left(1.0 - \frac{\gamma_{\text{geom}}}{0.15}\right) \tag{4.39}$$

resultando em $\lambda_k \in [1.2, 2.5]$ conforme γ_{geom} decresce de 0.15 a 0. Este regime reflete degradação fundamental de observabilidade: vetores de medição quase colineares produzem matriz de informação de Fisher mal-condicionada, tornando a solução do problema de Wahba numericamente instável (WAHBA, 1965; SHUSTER M. D.; OH, 1981).

Para $\gamma_{\text{geom}} \approx 0$ (vetores perfeitamente paralelos), o sistema tem apenas dois graus de liberdade observáveis, perdendo completamente informação sobre rotação em torno do eixo comum (Lefferts; Markley; Shuster, 1982; Markley, 2003). O escalonamento adaptativo de $\mathbf{R}_k$ compensa parcialmente esta perda elevando conservadoramente a incerteza reportada, prevenindo excesso de confiança do filtro em direções mal observadas (Pittelkau, 2003).

Este mecanismo é crítico em órbitas polares ou equatoriais, onde alinhamentos campo magnético-sol ocorrem periodicamente devido à geometria orbital (Wertz, 1978). Sem adaptação geométrica, o filtro pode apresentar divergência episódica durante travessias de configurações desfavoráveis (Markley F. L.; Crassidis, 2014).

4.4.5 Implementação no MEKF

O RAKF segue a estrutura do MEKF (Seção 4.3) com modificações estratégicas nas etapas de atualização para incorporar adaptação de covariância (Markley F. L.; Crassidis, 2014; Markley; Crassidis, 2014):

1. **Predição**: Idêntica ao MEKF, propagando $\hat{\mathbf{q}}_k^-$, $\hat{\mathbf{b}}_k^-$ e $\mathbf{P}_k^-$ através das equações de dinâmica de atitude e bias de giroscópio (Lefferts; Markley; Shuster, 1982).

2. **Cálculo preliminar da inovação**: Antes da adaptação, a covariância da inovação com $\mathbf{R}_k$ nominal é calculada (Kalman, 1960; Brown; Hwang, 1992):

$$\mathbf{S}_{\text{prelim}} = \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k \tag{4.40}$$

   Esta matriz é necessária para o cálculo da distância de Mahalanobis sem circularidade (Bar-Shalom; Li; Kirubarajan, 2001).

3. **Detecção e adaptação**: São calculados sequencialmente d_M (Eq. 4.30), γ_{geom} (Eq. 4.32), determinação de λ_k (Eq. 4.35), e atualização $\tilde{\mathbf{R}}_k = \lambda_k \mathbf{R}_k$ (Hide; Moore; Smith, 2003; Mohamed; Schwarz, 1999). Este passo é o que define a robustez adaptativa (Karlgaard; Schaub, 2007).

4. **Ganho de Kalman com R adaptado**: O ganho é calculado novamente utilizando a covariância de medição adaptada (Chang, 2012):

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \tilde{\mathbf{R}}_k \right)^{-1} \tag{4.41}$$

   O aumento de $\tilde{\mathbf{R}}_k$ reduz automaticamente a magnitude de $\mathbf{K}_k$, diminuindo o peso da medição anômala (Maybeck, 1979).

5. **Atualização**: É feita a correção multiplicativa do quatérnion ($\hat{\mathbf{q}}_k^+ = \delta\mathbf{q} \otimes \hat{\mathbf{q}}_k^-$) e atualização aditiva do bias ($\hat{\mathbf{b}}_k^+ = \hat{\mathbf{b}}_k^- + \delta\mathbf{b}$) conforme MEKF padrão (Lefferts; Markley; Shuster, 1982; Markley F. L.; Crassidis, 2014). A covariância é atualizada preferencialmente pela forma de Joseph para garantir simetria e definição positiva (Bierman, 1977).

A ordem de execução é crítica: d_M deve ser calculado antes da adaptação de $\mathbf{R}_k$ para evitar dependência circular entre métricas de detecção e parâmetros adaptativos (Yang; He; Xu, 2001; Chang, 2012).

4.4.6 Considerações de Estabilidade Numérica

Para garantir robustez numérica durante inversão de matrizes potencialmente mal-condicionadas, especialmente quando $\gamma_{\text{geom}} \approx 0$, é feita a regularização de Tikhonov (Tikhonov; Arsenin, 1977; Press *et al.*, 1992):

$$\mathbf{S}_{\text{reg}} = \frac{\mathbf{S} + \mathbf{S}^T}{2} + \varepsilon \mathbf{I} \tag{4.42}$$

com $\varepsilon = 10^{-6}$. A simetrização $(\mathbf{S}+\mathbf{S}^T)/2$ corrige erros de arredondamento acumulados em multiplicações matriciais sucessivas (Higham, 2002), enquanto a perturbação diagonal garante número de condição finito mesmo para matrizes singulares (Golub; Loan, 1996).

Adicionalmente, a covariância atualizada utiliza obrigatoriamente a forma de Joseph (Seção 4.7) (Bierman, 1977):

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)^T + \mathbf{K}_k\tilde{\mathbf{R}}_k\mathbf{K}_k^T \tag{4.43}$$

que preserva simetria e definição positiva mesmo com ganho não ótimo devido à adaptação (Grewal; Andrews, 2008). Saturação nos elementos da diagonal de $\mathbf{P}_k$ também é imposta (Markley F. L.; Crassidis, 2014):

$$P_{ii}(t) \leq (2)^2 \quad \text{para } i \in \{1,2,3\} \quad \text{(atitude)} \tag{4.44}$$

$$P_{ii}(t) \leq (0.5\ °/\text{hr})^2 \quad \text{para } i \in \{4,5,6\} \quad \text{(bias)} \tag{4.45}$$

prevenindo crescimento descontrolado da incerteza estimada durante sequências prolongadas de outliers ou geometria degradada persistente (Markley; Crassidis, 2014). Estes limites físicos refletem conhecimento a priori de performance dos sensores típicos de CubeSats (Wertz; Everett; Puschell, 2011).

4.5 Filtro de Kalman Ensemble

O filtro de Kalman Ensemble (EnKF) recebe essa denominação devido ao seu funcionamento baseado na utilização de um conjunto (ensemble) de amostras para a propagação dos estados, sem a necessidade de uma matriz de covariância. Essas características se tornam especialmente vantajosas em aplicações que apresentam alta não-linearidade, nas quais o Filtro de Kalman Estendido (EKF) não consegue lidar de maneira adequada. Além disso, ele se mostra eficaz em cenários onde existe uma grande incerteza nos estados do sistema, contribuindo para resultados mais confiáveis e robustos (Evensen, 1994; Evensen, 2003).

O Filtro de Kalman Ensemble é, em grande medida, calculado de maneira análoga ao Filtro de Kalman Estendido (EKF), apresentando, no entanto, algumas diferenças em aspectos particulares (Junkins, 2011):

Ao invés de utilizar apenas um vetor de estados como em 4.3.1.1, o EnKF utiliza um conjunto de amostras de estados:

$$\mathbf{X}_k = \left\{\mathbf{x}_k^{(1)}, \mathbf{x}_k^{(2)}, \ldots, \mathbf{x}_k^{(N)}\right\} \tag{4.46}$$

onde $x_k^{(i)}$ representa a i-ésima amostra do estado.

Enquanto no EKF um único estado é propagado utilizando a equação diferencial de atitude, no EnKF cada amostra é propagada independentemente e em paralelo:

$$\mathbf{x}_{k+1}^{(i)} = f(\mathbf{x}_k^{(i)}) + \mathbf{w}_k^{(i)} \tag{4.47}$$

O EKF utiliza a Jacobiana F para obter uma aproximação da covariância, já no EnKF a covariância é calculada de forma empírica a partir do conjunto de estados:

$$\mathbf{P}_{k+1} \approx \frac{1}{N-1} \sum_{i=1}^{N} (\mathbf{x}_k^{(i)} - \bar{\mathbf{x}}_k)(\mathbf{x}_k^{(i)} - \bar{\mathbf{x}}_k)^T \tag{4.48}$$

O ganho de Kalman é calculado analiticamente para realizar a atualização dos estados no EKF, já no EnKF o ganho de Kalman é calculado a partir da covariância do conjunto e todos os estados são atualizados individualmente:

$$\mathbf{K}_k = \mathbf{P}_k \mathbf{H}^T (\mathbf{H}\mathbf{P}_k \mathbf{H}^T + \mathbf{R}_k)^{-1} \tag{4.49}$$

Uma das desvantagens desse filtro reside no fato de que, apesar de ser significativamente mais robusto em comparação ao Filtro de Kalman Estendido (EKF), ele requer um maior número de amostras e exige um poder computacional mais elevado para funcionar de maneira eficiente. Isso pode tornar-se obstáculo dependendo do processador embarcado.

A implementação do EnKF utiliza $N = 50$ membros no ensemble, consistente com a escolha do EnSRF. O ruído do processo é amostrado de uma distribuição normal multivariada $\mathcal{N}(\mathbf{0}, \mathbf{Q})$ com:

$$\mathbf{Q} = \text{diag}\left(10^{-8}, 10^{-8}, 10^{-8}\right) \quad \text{rad}^2/\text{s}^2 \tag{4.50}$$

De forma similar, o ruído das medições é amostrado de $\mathcal{N}(\mathbf{0}, \mathbf{R})$ com:

$$\mathbf{R} = \text{diag}\left(10^{-4}, 10^{-4}, 10^{-4}\right) \tag{4.51}$$

A cada passo de tempo, as perturbações estocásticas são geradas independentemente para cada membro do ensemble, garantindo que as trajetórias explorem o espaço de estados de forma consistente com as incertezas especificadas. Esta abordagem permite ao EnKF capturar não-linearidades complexas no processo de reentrada atmosférica sem necessidade de linearização.

4.6 Filtro de Kalman Raiz Quadrada de Ensemble

O Filtro de Kalman Raiz Quadrada de Ensemble (EnSRF), do inglês *Ensemble Square Root Filter*, representa uma evolução das abordagens baseadas em conjuntos, concebida para aprimorar a estabilidade numérica e a precisão das estimativas em sistemas não-lineares. Ele emerge como uma alternativa ao EnKF tradicional, especificamente às suas formulações

estocásticas, ao adotar uma metodologia determinística para a etapa de atualização dos estados (Whitaker; Hamill, 2002).

A principal característica que o distingue de outras variantes do EnKF é a forma como a incerteza das medições é incorporada ao sistema. Em implementações estocásticas, é comum que a medição real seja artificialmente perturbada com um ruído aleatório para cada membro do conjunto, uma técnica que, embora funcional, pode introduzir erros de amostragem adicionais. O EnSRF, em contrapartida, contorna essa necessidade, utilizando a medição real de forma direta e aplicando uma transformação determinística para atualizar o conjunto de estados, o que pode resultar em estimativas mais consistentes.

A designação "Raiz Quadrada"em sua nomenclatura alude à maneira como a matriz de covariância do erro é manipulada. O EnSRF opera diretamente sobre a matriz de perturbações do conjunto — a "raiz quadrada"da covariância. Considerando N membros na previsão $\mathbf{X}^f$:

$$\mathbf{X}^f = \left[\mathbf{x}_1^f, \mathbf{x}_2^f, \ldots, \mathbf{x}_N^f\right] \tag{4.52}$$

A partir deste, a média da previsão ($\bar{\mathbf{x}}^f$) e a matriz de perturbações ($\mathbf{A}^f$) são calculadas, representando os desvios de cada membro em relação à média:

$$\bar{\mathbf{x}}^f = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i^f \tag{4.53}$$

$$\mathbf{A}^f = \mathbf{X}^f - \bar{\mathbf{X}}^f \tag{4.54}$$

onde $\bar{\mathbf{X}}^f$ é a matriz cujas colunas são todas iguais ao vetor médio $\bar{\mathbf{x}}^f$. A matriz de covariância do erro da previsão, $\mathbf{P}^f$, é então implicitamente representada por $\mathbf{P}^f = \frac{1}{N-1}\mathbf{A}^f(\mathbf{A}^f)^T$.

Na etapa de atualização, a média do conjunto é corrigida de forma análoga ao filtro de Kalman padrão:

$$\bar{\mathbf{x}}^a = \bar{\mathbf{x}}^f + \mathbf{K}(\mathbf{y}^o - H\bar{\mathbf{x}}^f) \tag{4.55}$$

onde $\mathbf{K}$ é o ganho de Kalman, $\mathbf{y}^o$ a medição real e H o operador de observação. O passo crucial do EnSRF reside na atualização determinística da matriz de perturbações:

$$\mathbf{A}^a = \mathbf{A}^f - \alpha\mathbf{K}H\mathbf{A}^f \tag{4.56}$$

O fator escalar α é um coeficiente de correção que garante que a covariância resultante seja matematicamente consistente, e sua forma exata depende da implementação específica do EnSRF. Finalmente, o novo conjunto de análise, $\mathbf{X}^a$, é reconstituído:

$$\mathbf{X}^a = \bar{\mathbf{X}}^a + \mathbf{A}^a \tag{4.57}$$

Apesar de suas vantagens em robustez numérica, o EnSRF, assim como outros filtros de conjunto, continua dependente de um número suficientemente grande de membros

no *ensemble* para representar adequadamente as estatísticas do erro, sendo este um fator limitante em sua aplicação.

A implementação do EnSRF neste trabalho utiliza um conjunto de $N = 50$ membros no ensemble. Este valor representa um compromisso entre precisão estatística e custo computacional. Ensembles menores ($N < 30$) apresentam subamostragem do espaço de estados, resultando em estimativas enviesadas das covariâncias. Ensembles maiores ($N > 100$) oferecem ganhos marginais em troca de custo computacional significativamente maior.

A média do ensemble para vetores de estado com quatérnion requer tratamento especial devido à natureza não-euclidiana de $\mathbb{S}^3$. O método de Markley (Markley *et al.*, 2007) é empregado para calcular o quatérnion médio através da matriz:

$$\mathbf{M} = \sum_{i=1}^{N} \mathbf{q}^{(i)} (\mathbf{q}^{(i)})^T \tag{4.58}$$

O quatérnion médio corresponde ao autovetor associado ao maior autovalor de $\mathbf{M}$. Esta abordagem garante consistência geométrica da média em $\mathbb{S}^3$.

As matrizes de covariância do processo e medição são mantidas constantes:

$$\mathbf{Q} = \text{diag}\left(10^{-8}, 10^{-8}, 10^{-8}\right) \quad \text{rad}^2/\text{s}^2 \tag{4.59}$$

para o ruído do giroscópio, e

$$\mathbf{R} = \text{diag}\left(10^{-4}, 10^{-4}, 10^{-4}\right) \tag{4.60}$$

para as medições vetoriais normalizadas de magnetômetro e sensor solar.

4.7 Considerações Numéricas Comuns

A implementação prática de filtros de estimação de estado requer atenção a aspectos de estabilidade numérica e consistência geométrica. Esta seção descreve técnicas aplicadas uniformemente em todos os filtros implementados.

4.7.1 Garantia de Semi-Definição Positiva

As matrizes de covariância devem ser simétricas e semi-definidas positivas (SPD) para manter significado físico. Erros de arredondamento acumulados podem violar essas propriedades. A função de regularização aplicada é:

$$\text{SPD}(\mathbf{P}) = \frac{\mathbf{P} + \mathbf{P}^T}{2} + \varepsilon \mathbf{I} \tag{4.61}$$

onde $\varepsilon = 10^{-10}$ é um valor de regularização pequeno. A simetrização corrige assimetrias numéricas, enquanto a adição de $\varepsilon\mathbf{I}$ garante autovalores positivos.

4.7.2 Forma de Joseph para Atualização da Covariância

Nos filtros baseados em Kalman (EKF, MEKF, RAKF), a atualização da covariância utiliza a forma de Joseph (Bucy; Joseph, 1969):

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)^T + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T \tag{4.62}$$

Esta forma é numericamente superior à forma simplificada $\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-$ pois garante simetria e positividade mesmo quando o ganho $\mathbf{K}_k$ não é ótimo devido a erros numéricos.

4.7.3 Média de quatérnions em Métodos Ensemble

O cálculo da média do ensemble em $\mathbb{S}^3$ requer o método de Markley (Markley *et al.*, 2007). Dada uma coleção de quatérnions $\{\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(N)}\}$, a matriz:

$$\mathbf{M} = \sum_{i=1}^{N} \mathbf{q}^{(i)}(\mathbf{q}^{(i)})^T \tag{4.63}$$

é formada e diagonalizada. O quatérnion médio corresponde ao autovetor associado ao maior autovalor de $\mathbf{M}$. Este procedimento evita o viés introduzido por médias aritméticas diretas, que não respeitam a geometria de $\mathbb{S}^3$.

4.7.4 Extração de Covariância do QUEST

O algoritmo QUEST fornece a estimativa ótima de atitude mas não produz diretamente a matriz de covariância. A covariância associada à solução QUEST é calculada através do Hessiano da função de perda de Wahba (Markley, 1988):

$$\mathbf{P}_{\text{QUEST}} = \left(\sum_{i=1}^{m} w_i \mathbf{I} - \tilde{\mathbf{B}}^T \tilde{\mathbf{B}} \right)^{-1} \tag{4.64}$$

onde w_i são os pesos das medições e $\tilde{\mathbf{B}}$ é a matriz de perfil de atitude. Esta covariância é utilizada para inicializar os filtros recursivos subsequentes.

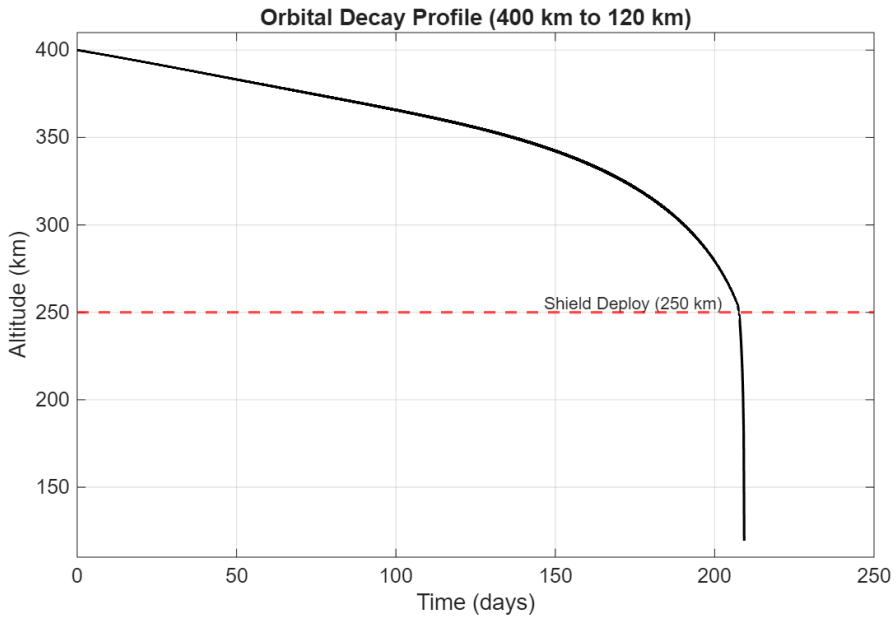
5 Resultados e Discussão

A simulação compara seis algoritmos de estimação durante reentrada de CubeSat 12U: um filtro instantâneo (QUEST) e cinco filtros recursivos (Alfa, EKF, RAKF, EnKF, EnSRF)

5.1 Perfil da Missão e Condições Orbitais

O satélite decai de 400 km até 120 km em 209 dias (Figura 5.1). O escudo abre em 250 km, aumentando a área frontal 5,2× (0,15 → 0,785 m^2).

Figura 5.1 – Perfil de decaimento orbital: altitude versus tempo da órbita inicial (400 km) até altitude final (120 km)



Fonte: Autoria própria

Taxa de decaimento varia: lenta na fase inicial (400-250 km) com densidade baixa e área reduzida, acelerando após abertura do escudo. A dinâmica orbital inclui variações de densidade atmosférica com latitude e altitude, modeladas pelo NRLMSISE-00, que causam modulações na taxa de decaimento não visíveis em modelos atmosféricos simplificados.

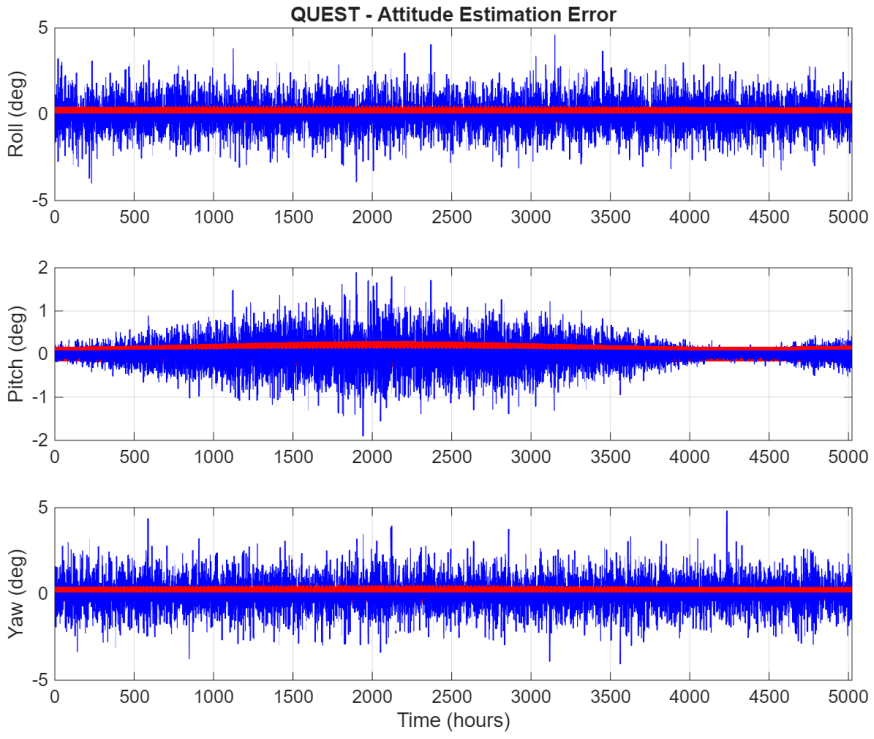
O sistema de medição é composto por magnetômetro triaxial (ruído gaussiano constante de 50 nT RMS) e dois sensores solares digitais (ruído de 0,05° RMS), operando com período de amostragem de 60 s (0,0167 Hz). Em 5% das medições do magnetômetro o ruído é multiplicado por 10 para gerar outliers sintéticos. O giroscópio de referência é modelado com $\sigma_v = 3{,}162 \times 10^{-7}$ rad/s$^{1/2}$ (angle random walk) e $\sigma_u = 3{,}162 \times 10^{-10}$ rad/s$^{1/2}$ (random

walk do bias), resultando em ruído discreto de aproximadamente $2{,}3 \times 10^{-6}$ °/s por eixo no passo de 60 s e bias inicial de 0,1°/h.

5.2 Desempenho do QUEST e Filtro Alfa

O QUEST é um filtro instantâneo que resolve o problema de Wahba a cada instante, enquanto o Filtro Alfa é um estimador recursivo que propaga a estimativa de atitude no tempo, embora não mantenha estimativa de covariância ou bias de giroscópio. A Figura 5.2 mostra o erro de atitude do QUEST ao longo da missão. O erro médio RMS é de 0,136 graus, refletindo a precisão do método de otimização de Wahba quando sensores de boa qualidade estão disponíveis.

Figura 5.2 – Erro de atitude do filtro QUEST nos três eixos (roll, pitch, yaw) com bandas de incerteza 3-sigma

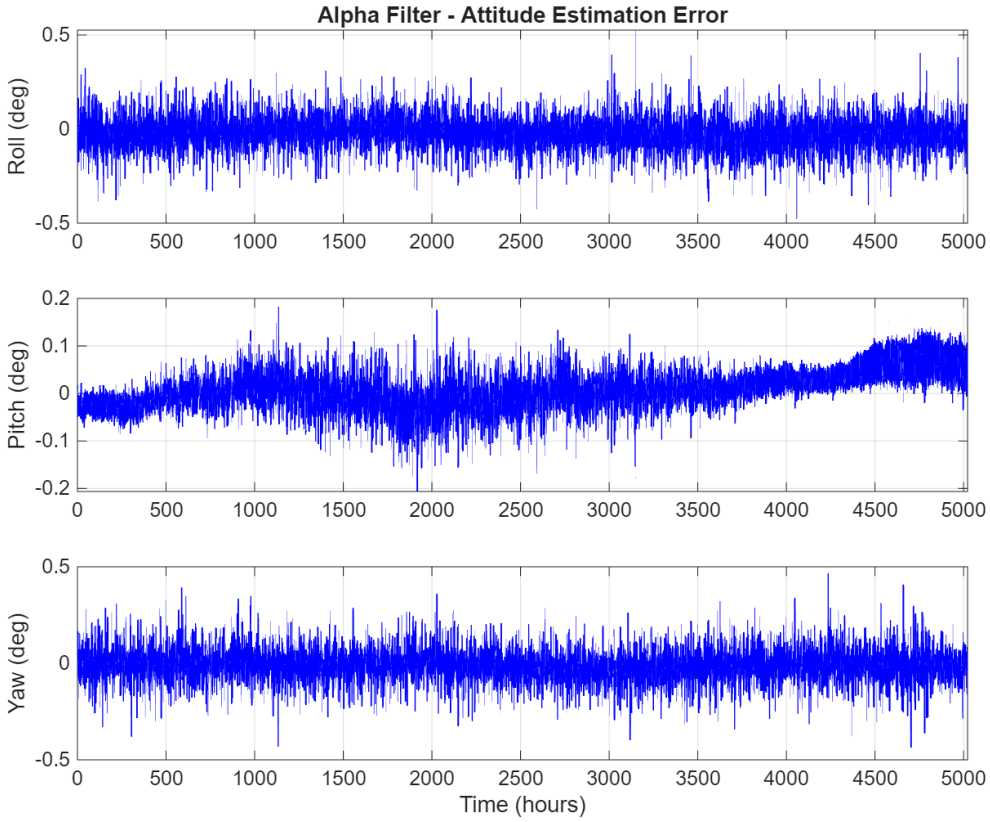


Fonte: Autoria própria

As bandas de 3-sigma mostradas na Figura 5.2 são calculadas a partir da covariância da solução QUEST, derivada da matriz de ganho de Davenport. O erro permanece dentro dos limites 3-sigma, validando a covariância do QUEST. Picos esporádicos de erro correlacionam-se com períodos de eclipse (ausência de medição do sensor solar) e com outliers injetados nos sensores de magnetômetro.

O Filtro Alfa, mostrado na Figura 5.3, apresenta desempenho similar ao QUEST em termos de erro absoluto, mas não fornece covariância de atitude estimada (portanto, sem bandas de incerteza nos gráficos). O Filtro Alfa é computacionalmente mais leve que QUEST, porém oferece menos informação estatística para fusão de dados ou análise de consistência.

Figura 5.3 – Erro de atitude do Filtro Alfa nos três eixos (roll, pitch, yaw)



Fonte: Autoria própria

Ambos os filtros instantâneos dependem exclusivamente da qualidade das medições atuais. Quando outliers severos ocorrem simultaneamente em ambos os sensores, não há mecanismo de suavização temporal para mitigar o impacto. Apesar disso, a taxa de outliers de 5% injetada na simulação não causa divergência perceptível nos filtros instantâneos, demonstrando robustez adequada para cenários de missão com sensores moderadamente confiáveis.

5.3 Desempenho dos Filtros Baseados em Kalman

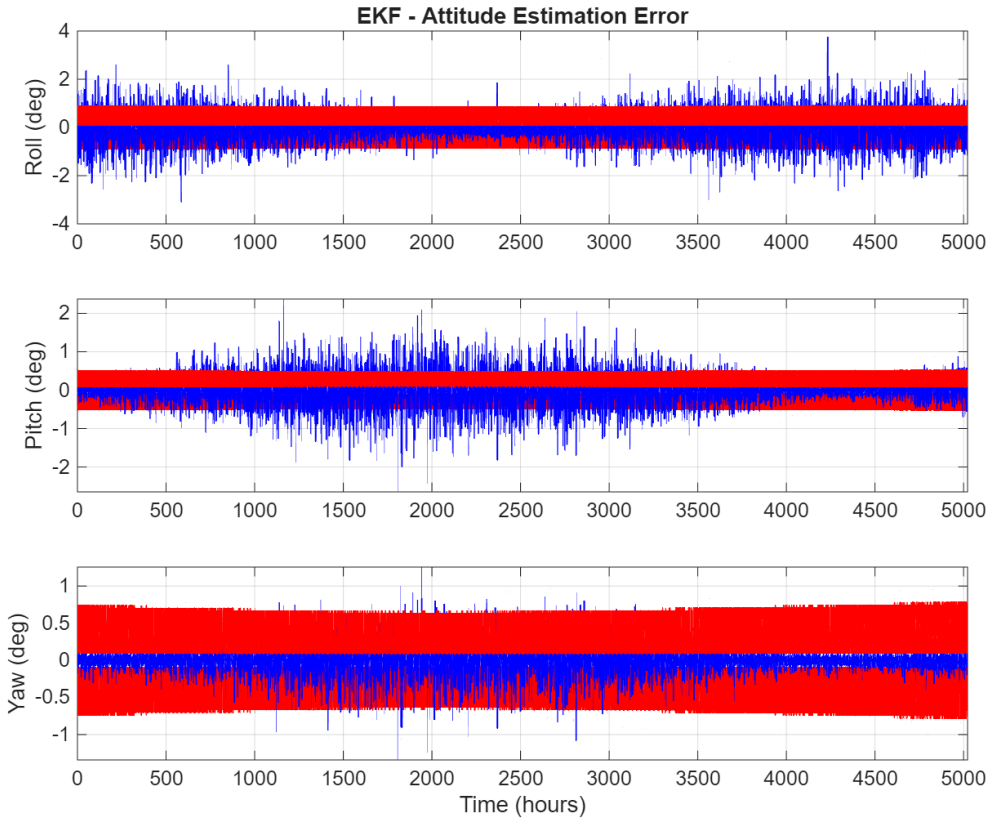
Os filtros baseados em Kalman incorporam propagação temporal da covariância e estimação de bias do giroscópio, proporcionando estimativas de atitude suavizadas e mais

robustas a ruídos de medição transitórios. Esta seção compara o desempenho de quatro implementações: EKF, RAKF, EnKF e EnSRF.

5.3.1 Extended Kalman Filter (EKF)

O EKF convencional, mostrado na Figura 5.4, apresenta erro médio RMS de 0,117 graus. Este nível de erro reflete o comportamento em presença de outliers não tratados: o EKF assume que todas as medições seguem o modelo gaussiano nominal e não adapta a covariância de medição **R** quando detecta inconsistências.

Figura 5.4 – Erro de atitude do EKF nos três eixos (roll, pitch, yaw) com bandas de incerteza 3-sigma



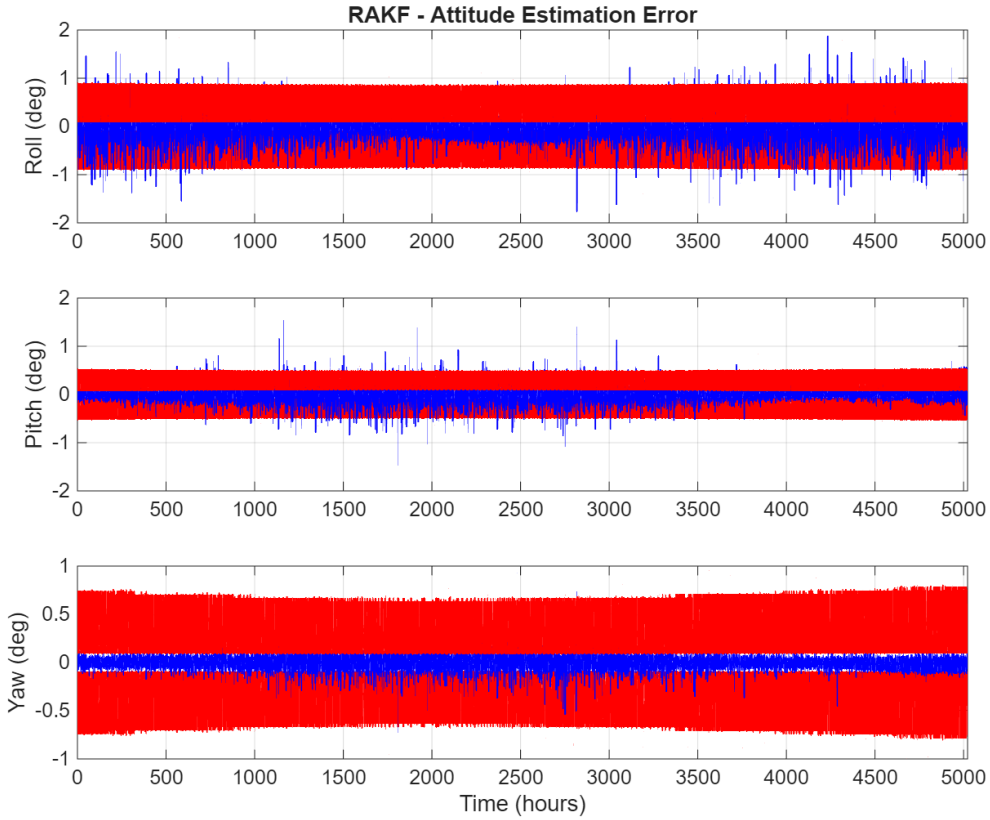
Fonte: Autoria própria

Picos de erro coincidem com outliers — pontos além das bandas 3-sigma. Após outliers, o EKF requer vários passos de tempo para reconverger, pois a atualização da covariância é gradual. Durante períodos sem outliers, o erro permanece consistentemente dentro das bandas de confiança, validando o modelo de propagação e a linearização de primeira ordem do MEKF.

5.3.2 Robust Adaptive Kalman Filter (RAKF)

O RAKF, apresentado na Figura 5.5, reduz o erro médio RMS para 0,099 graus, uma melhoria de 15% em relação ao EKF convencional. O mecanismo de adaptação detecta outliers via distância de Mahalanobis e escala a covariância de medição **R** por um fator λ_k que varia entre 1,0 (nominal) e 5,0 (outlier severo confirmado), com média de 1,12 ao longo da missão.

Figura 5.5 – Erro de atitude do RAKF nos três eixos (roll, pitch, yaw) com bandas de incerteza 3-sigma adaptadas



Fonte: Autoria própria

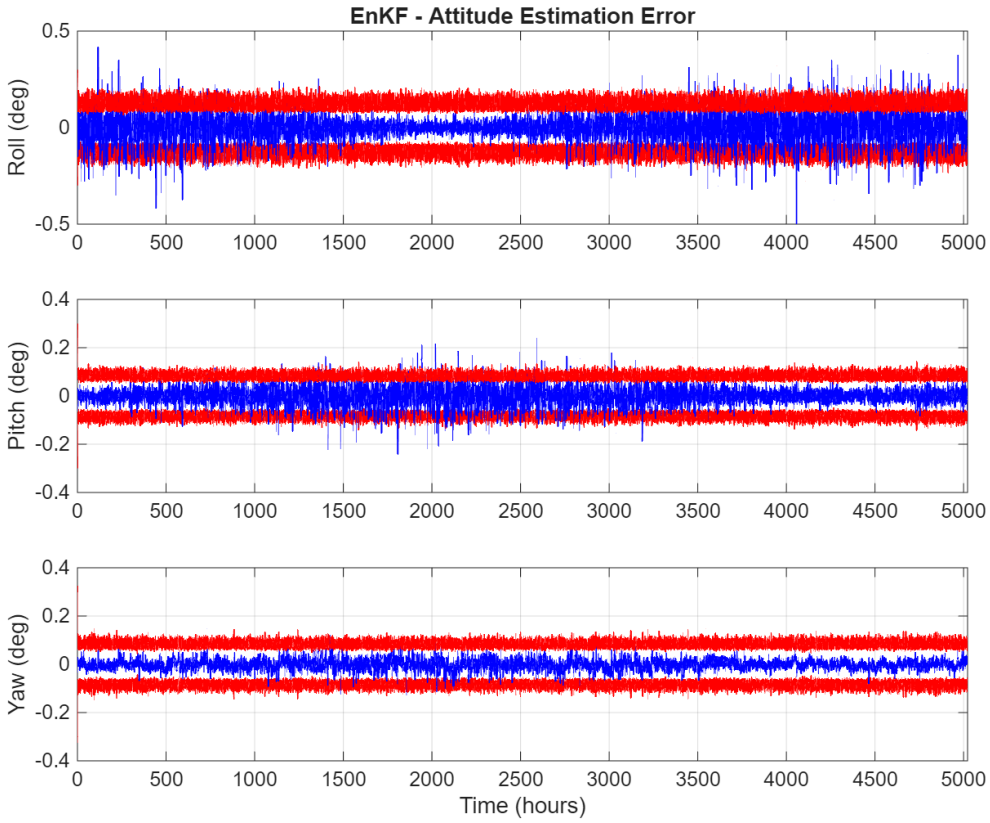
Picos de erro no RAKF (Figura 5.5) são menores que no EKF (Figura 5.4). Quando um outlier é detectado, o ganho de Kalman efetivo diminui automaticamente (devido ao aumento de **R**), reduzindo o peso da medição anômala na atualização de estado. Após a detecção, o fator λ_k retorna gradualmente ao valor nominal, permitindo reconvergência suave.

As bandas de 3-sigma do RAKF expandem-se momentaneamente durante eventos de outlier, refletindo o aumento da incerteza estimada quando medições suspeitas são processadas. Este comportamento é desejável: o filtro reconhece que não pode confiar plenamente na medição e aumenta conservadoramente sua incerteza reportada.

5.3.3 Ensemble Kalman Filter (EnKF)

O EnKF, mostrado na Figura 5.6, alcança erro médio RMS de 0,039 graus, o segundo melhor desempenho dentre todos os filtros testados. O método ensemble propaga 50 membros em paralelo, cada um com ruído amostrado independentemente, e estima a covariância empiricamente a partir da dispersão dos membros. Esta abordagem não lineariza a dinâmica, capturando não-gaussianidades introduzidas pela propagação multiplicativa de quaternions e pelo arrasto atmosférico variável.

Figura 5.6 – Erro de atitude do EnKF nos três eixos (roll, pitch, yaw) com bandas de incerteza 3-sigma derivadas do ensemble



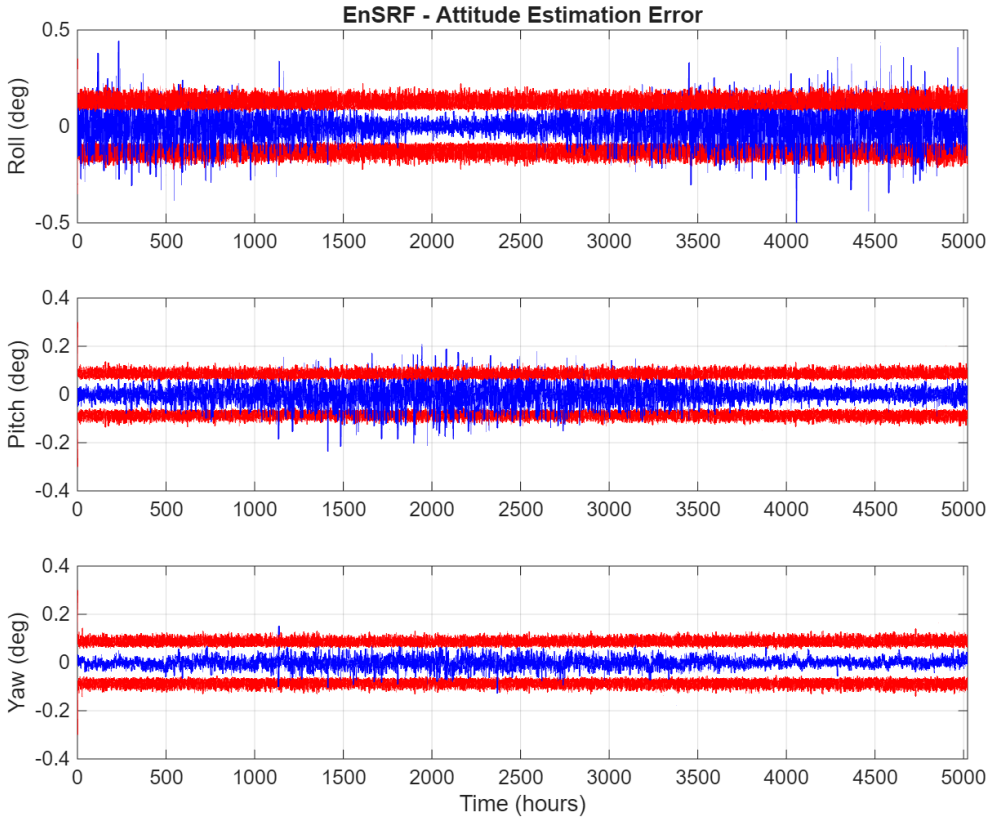
Fonte: Autoria própria

As bandas de incerteza do EnKF são geralmente mais largas que as do EKF, especialmente durante períodos de dinâmica acelerada (alta taxa de decaimento orbital). Isto reflete a capacidade do ensemble de capturar dispersão estatística real que a linearização de primeira ordem do EKF subestima. A consistência entre erro verdadeiro e bandas de 3-sigma é superior ao EKF, embora não tão conservadora quanto o RAKF durante outliers.

5.3.4 Ensemble Square Root Filter (EnSRF)

O EnSRF, apresentado na Figura 5.7, obtém erro médio RMS de 0,038 graus, o melhor desempenho dentre todos os filtros testados. A diferença fundamental entre EnKF e EnSRF está na forma de atualização: o EnSRF atualiza a matriz de covariância deterministicamente sem adicionar ruído artificial na etapa de medição, evitando a necessidade de inflação de covariância.

Figura 5.7 – Erro de atitude do EnSRF nos três eixos (roll, pitch, yaw) com bandas de incerteza 3-sigma derivadas do ensemble



Fonte: Autoria própria

A Figura 5.7 mostra comportamento qualitativo similar ao EnKF, porém com bandas de incerteza ligeiramente mais suaves (menor variabilidade temporal). O EnSRF é preferível ao EnKF em aplicações onde estabilidade numérica de longo prazo é crítica, pois não acumula ruído artificial nas atualizações. O custo computacional de ambos os métodos é comparável: EnKF requer 844 segundos e EnSRF requer 823 segundos para a simulação completa (Tabela 5.2).

5.4 Análise Comparativa e Custo Computacional

A Tabela 5.1 sumariza o desempenho quantitativo de todos os filtros. O Filtro Alpha apresenta o terceiro melhor erro médio (0,058 graus), enquanto o QUEST apresenta erro médio de 0,136 graus. Ambos não fornecem estimativa de bias de giroscópio e dependem exclusivamente da qualidade instantânea das medições.

Tabela 5.1 – Estatísticas de desempenho de estimação de atitude (RMS em graus)

Filtro	Média	Mediana	Desvio Padrão	Percentil 95
QUEST	0,136	0,085	0,236	0,302
Alpha	0,058	0,047	0,040	0,137
EKF	0,117	0,072	0,164	0,349
RAKF	0,099	0,069	0,100	0,287
EnKF	0,039	0,031	0,030	0,097
EnSRF	0,038	0,028	0,030	0,098

Fonte: Autoria própria

Dentre os filtros Kalman, o EnSRF e EnKF apresentam os melhores resultados, com erros médios de 0,038 e 0,039 graus respectivamente. O RAKF oferece desempenho intermediário (0,099 graus) com custo computacional significativamente menor que os métodos ensemble. O EKF convencional apresenta erro médio de 0,117 graus devido à ausência de mecanismos de robustez a outliers.

A Tabela 5.2 compara o custo computacional absoluto de cada filtro. O filtro Alpha é incluído para referência, apresentando tempo de execução de 4,14 segundos. O RAKF adiciona overhead de 99% sobre o EKF devido ao cálculo da distância de Mahalanobis e qualidade geométrica a cada passo. Os métodos ensemble são ordens de magnitude mais caros: EnKF requer 33 vezes mais tempo que o EKF, e EnSRF requer 32 vezes mais.

Tabela 5.2 – Custo computacional dos filtros baseados em Kalman

Filtro	Tempo (s)	Fator relativo ao EKF
Alpha	4,14	0,16×
EKF	25,36	1,00×
RAKF	50,58	1,99×
EnKF	843,93	33,27×
EnSRF	823,27	32,46×

Fonte: Autoria própria

Para aplicações embarcadas em CubeSats com processadores de baixa potência (e.g., ARM Cortex-M4 a 48 MHz), o custo dos métodos ensemble é proibitivo. O RAKF emerge como compromisso atrativo: oferece 15% de melhoria sobre o EKF com overhead modesto

de 99%, tornando-se viável para implementação em tempo real com taxa de atualização de 10 Hz típica de missões de atitude.

5.5 Discussão e Recomendações

Os resultados demonstram trade-offs claros entre precisão, robustez e eficiência computacional. Não existe um único filtro universalmente superior, a escolha deve considerar o contexto de aplicação.

Para missões críticas com processamento em solo (ground processing), os filtros ensemble (particularmente EnSRF) são recomendados. A precisão de 0,038° e robustez excepcional justificam o custo computacional quando recursos não são limitantes. Aplicações como controle de apontamento de telescópios espaciais ou reentrada guiada de alto valor se beneficiariam desta abordagem.

Em sistemas embarcados com processadores de baixo custo, o RAKF emerge como compromisso atrativo. O overhead de 99% sobre o EKF é compensado pela redução de 15% no erro médio. Para CubeSats em missões de demonstração tecnológica onde falhas ocasionais de sensores são esperadas, o RAKF oferece margem de segurança significativa.

O EKF convencional permanece válido para aplicações onde outliers são raros e recursos computacionais são extremamente limitados. Microcontroladores de 8 bits ou aplicações com restrições severas de energia podem se beneficiar da simplicidade do EKF.

O Filtro Alpha apresenta erro médio de 0,058°, superando RAKF (0,099°) e EKF (0,117°), revelando que para dinâmicas rotacionais lentas (como spin-stabilized CubeSats ou satélites em pointing mode), métodos simples podem ser competitivos. Entretanto, a ausência de estimação de incerteza e bias limita sua aplicabilidade.

Os filtros ensemble apresentam desempenho superior em precisão (0,038-0,039° vs. 0,099-0,117° dos filtros Kalman convencionais), mas com custo computacional 32-33 vezes maior que o EKF que limita sua aplicabilidade embarcada. O RAKF oferece equilíbrio interessante entre desempenho (15% melhor que EKF) e custo (99% de overhead), adequado para missões onde outliers esporádicos são esperados e recursos computacionais são moderados. O EKF convencional permanece a escolha para sistemas com restrições severas de processamento e cenários de baixa incidência de anomalias.

6 Conclusão

Este trabalho comparou seis algoritmos de estimação de atitude durante uma missão de reentrada atmosférica de um CubeSat 12U equipado com escudo térmico inflável, simulando o decaimento orbital de 400 km até 120 km de altitude ao longo de 209 dias. A comparação envolveu dois filtros instantâneos (QUEST e Filtro Alfa) e quatro filtros baseados em Kalman (EKF, RAKF, EnKF e EnSRF), avaliando precisão, robustez a outliers e custo computacional em condições de sensores imperfeitos e medições anômalas.

Os resultados revelam trade-offs fundamentais entre desempenho e viabilidade de implementação embarcada. Os filtros ensemble apresentaram os menores erros médios RMS (EnSRF: 0,038°, EnKF: 0,039°), confirmando que métodos ensemble capturam não-linearidades complexas da cinemática de atitude e dinâmica orbital com arrasto atmosférico variável sem necessidade de Jacobianos. A capacidade de representar distribuições não-gaussianas através da propagação de 50 membros independentes proporcionou robustez superior durante períodos de geometria sensor-vetor degradada e outliers esporádicos (5% das medições do magnetômetro). No entanto, o custo computacional de 823-844 segundos para a simulação completa (32-33 vezes maior que o EKF) inviabiliza implementação em processadores típicos de CubeSats, como ARM Cortex-M4 operando a 48 MHz, onde requisitos de tempo real exigem execução em milissegundos por ciclo de atualização.

O Filtro Alpha surpreendeu com o terceiro melhor desempenho (0,058°), superando RAKF (0,099°) e EKF (0,117°) apesar de não propagar covariância temporal ou estimar bias de giroscópio. Este resultado revela que para dinâmicas rotacionais lentas e previsíveis, como atitude nadir durante reentrada atmosférica, a fusão instantânea de medições vetoriais via QUEST pode ser competitiva com filtros recursivos mais complexos. A simplicidade computacional (4,14 segundos, apenas 16% do tempo do EKF) torna o Filtro Alpha atrativo para sistemas com recursos extremamente limitados, porém a ausência de estimativa de incerteza e bias limita sua aplicabilidade em sistemas que requerem fusão de dados com outros subsistemas ou análise de consistência estatística.

O RAKF emerge como solução de compromisso para missões com recursos computacionais moderados. O erro médio de 0,099° representa melhoria de 15% sobre o EKF convencional, com overhead de 99% no tempo de processamento (50,58 s vs. 25,36 s). O mecanismo de adaptação baseado em distância de Mahalanobis e qualidade geométrica demonstrou eficácia em mitigar impacto de outliers, com fator de escalonamento médio de 1,12 e máximo de 5,0 durante eventos anômalos. A qualidade geométrica média de 0,945 indica que configurações sensor-vetor favoráveis predominaram durante a missão, com mínimo de 0,636 durante alinhamentos temporários entre campo magnético e direção solar. Para CubeSats em missões de demonstração tecnológica onde falhas ocasionais de sensores

são esperadas, o RAKF oferece margem de segurança significativa sem exigir hardware especializado.

O EKF convencional apresentou erro médio de 0,117°, desempenho intermediário que reflete ausência de mecanismos de robustez. Outliers não tratados causam divergência temporária do filtro, com reconvergência lenta devido à propagação gradual da covariância. Este comportamento valida a necessidade de estratégias adaptativas em ambientes onde medições anômalas são inevitáveis, como interferência eletromagnética de magnetotorquers, erros de conversão analógico-digital ou radiação ionizante em anomalias geomagnéticas. O QUEST apresentou o maior erro médio (0,136°) dentre todos os filtros, com desvio padrão elevado (0,236°) indicando sensibilidade a outliers instantâneos sem mecanismo de suavização temporal.

As limitações deste trabalho devem ser consideradas ao interpretar os resultados. O modelo de ruído constante para sensores não captura degradação devida ao aquecimento aerodinâmico crescente durante reentrada, interferência de plasma ionosférico em altitudes abaixo de 200 km, ou ciclagem térmica acelerada. A inclusão desses efeitos ambientais dependentes de altitude e temperatura exigiria modelos fenomenológicos complexos de difícil validação sem dados de voo real, fugindo ao escopo de comparação algorítmica. Além disso, a simulação assume que o satélite mantém atitude de apontamento ao nadir durante toda a missão, cenário idealizado que não contempla perturbações de torque gravitacional diferencial, torque magnético residual ou torque aerodinâmico assimétrico após abertura do escudo em 250 km. Essas simplificações permitem isolar o desempenho intrínseco dos algoritmos de filtragem, mas não refletem completamente a complexidade de uma missão operacional.

Trabalhos futuros podem explorar diversas direções promissoras. A implementação dos filtros em hardware embarcado real (por exemplo, STM32H7 dual-core a 480 MHz) permitiria benchmarks de desempenho mais precisos, incluindo análise de consumo energético e latência de execução em tempo real. A validação experimental com dados de voo de missões reais de CubeSats em órbita baixa (como o MIRCA2) forneceria ground truth para calibração de modelos de ruído e verificação de robustez em condições não simuladas. A extensão para filtros híbridos, combinando RAKF para propagação rápida com atualizações periódicas de EnSRF em janelas de baixa carga computacional, pode balancear precisão e eficiência. Finalmente, a incorporação de modelos de arrasto atmosférico mais refinados (incluindo efeitos de atividade solar extrema e tempestades geomagnéticas) e torques perturbativos (gradiente de gravidade, pressão de radiação solar, interação magnética) aumentaria o realismo da simulação e permitiria avaliar o impacto dessas dinâmicas acopladas na convergência dos filtros.

A escolha do algoritmo de estimação de atitude para uma missão específica deve considerar o contexto completo de aplicação. Para processamento em solo com recursos

ilimitados, EnSRF é recomendado devido à precisão superior (0,038°) e estabilidade numérica de longo prazo. Em sistemas embarcados com processadores de médio desempenho e expectativa de falhas esporádicas de sensores, RAKF oferece o melhor compromisso entre desempenho (15% melhor que EKF) e viabilidade de implementação (overhead de 99%). Para sistemas com recursos extremamente limitados onde estimativa de bias não é crítica, o Filtro Alpha apresenta alternativa eficiente (16% do custo computacional do EKF). O EKF convencional permanece válido apenas em cenários onde outliers são raros e recursos computacionais são moderadamente limitados, oferendo base sólida para extensões adaptativas como o RAKF.

Referências

ÅSTRÖM, K. J.; WITTENMARK, B. **Adaptive control**. 2nd. ed. Reading, MA: Addison-Wesley, 1995. Citado na p. 56.

BAR-SHALOM, Y.; LI, X.-R.; KIRUBARAJAN, T. **Estimation with applications to tracking and navigation**. New York: John Wiley & Sons, 2001. Citado nas pp. 53, 55, 56 e 57.

BASSANO, E.; SAVINO, R.; RICHIELLO, C.; RUSSO, G.; AURIGEMMA, R.; PUNZO, F. Irene - italian re-entry nacelle for microgravity experiments. *In*: . [*S.l.: s.n.*], 2011. Citado na p. 19.

BENEDICT, T. R.; BORDNER, G. W. Synthesis of an optimal set of radar track-while-scan smoothing equations. **IRE Transactions on Automatic Control**, v. 7, n. 4, p. 27–32, 1962. Citado na p. 46.

BIERMAN, G. J. **Factorization methods for discrete sequential estimation**. New York: Academic Press, 1977. Citado nas pp. 57 e 58.

BROWN, R. G.; HWANG, P. Y. **Introduction to random signals and applied Kalman filtering**. 2nd. ed. New York: John Wiley & Sons, 1992. Citado nas pp. 53 e 57.

BUCY, R. S.; JOSEPH, P. D. Filtering for stochastic processes with applications to guidance. **Interscience**, New York, 1969. Citado na p. 62.

BURDEN, R. L.; FAIRES, J. D.; BURDEN, A. M. **Numerical Analysis**. 10th. ed. [*S.l.*]: Cengage Learning, 2015. Citado na p. 35.

CASSELL, A.; SMITH, B.; WERCINSKI, P.; GHASSEMIEH, S.; HIBBARD, K.; NELESSEN, A.; CUTTS, J. ADEPT, a mechanically deployable Re-Entry Vehicle system, enabling interplanetary CubeSAT and small satellite missions. *In*: . [*s.n.*], 2018. Disponível em: https://ntrs.nasa.gov/api/citations/20190002830/downloads/20190002830.pdf. Citado nas pp. 8, 18 e 19.

CHANG, G. Huber-based novel robust unscented Kalman filter. **IET Science, Measurement & Technology**, IET, v. 6, n. 6, p. 502–509, 2012. Citado nas pp. 54, 55, 56 e 57.

CORDEIRO, T. F. K. **Fusão decentralizada com filtragem de Kalman estendida para estimação de atitude e deriva com rede de sensores inerciais e magnetômetro de baixo custo**. Dissertação (Mestrado) — Instituto Tecnológico de Aeronáutica, São José dos Campos, 2012. Citado na p. 49.

CUBESATSHOP. **SSOC-D60 2-Axis digital sun sensor - CubeSatShop.com**. 5 2023. 2023. Disponível em: https://www.cubesatshop.com/product/ssoc-d60-2-axis-digital-sun-sensor/. Citado nas pp. 8 e 30.

CURTIS, H. D. **Orbital Mechanics for Engineering Students**. 3rd. ed. Oxford, UK: Butterworth-Heinemann, 2013. ISBN 978-0-08-097747-8. Citado nas pp. 35 e 43.

DAVENPORT, P. B. **A vector approach to the algebra of rotations with applications**. [*S.l.*], 1968. Citado na p. 45.

DIMINO, I.; VENDITTOZZI, C.; SILVA, W. R.; AMEDURI, S.; CONCILIO, A. A morphing deployable mechanism for Re-Entry capsule Aeroshell. **Applied Sciences**, v. 13, n. 5, p. 2783, 2 2023. Disponível em: https://doi.org/10.3390/app13052783. Citado nas pp. 8 e 20.

ESPER, J. **CUBESAT Application for Planetary Entry (CAPE) Missions: Micro-Return Capsule (MIRCA)**. 2025. 2025. Disponível em: https://ntrs.nasa.gov/api/citations/20160010298/downloads/20160010298.pdf. Acesso em: 17 fev. 2025. Citado na p. 18.

EVENSEN, G. Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. **Journal of Geophysical Research**, v. 99, n. C5, p. 10143–10162, 1994. Citado na p. 58.

EVENSEN, G. The ensemble kalman filter: Theoretical formulation and practical implementation. **Ocean Dynamics**, v. 53, n. 4, p. 343–367, 2003. Citado na p. 58.

FARRELL, J. **Attitude determination by Kalman filtering**. [*S.l.: s.n.*], 1970. v. 6. 419–430 p. Citado na p. 16.

FARRELL, J. A. **Aided Navigation: GPS with High Rate Sensors**. [*S.l.*]: McGraw Hill Professional, 2008. Citado nas pp. 32 e 39.

FRAUNHOFER. **Gyrocompass: The use of a fully integrated precision MEMS gyroscope in a standalone North-Finder - Fraunhofer ENAS**. 2025. 2025. Disponível em: https://www.enas.fraunhofer.de/en/Business_Units/Smart_Systems/Inertial_Components_and_Systems/High-Precision_MEMS_Angular_Rate_Sensor/Gyrocompass_The_use_of_a_fully_integrated_precision_MEMS_gyroscope_in_a_standalone_north-finder.html. Citado nas pp. 8 e 32.

Goddard Space Flight Center. **A Summary of the Upper Atmosphere Research Satellite (UARS)**. 2025. 2025. Disponível em: https://uars.gsfc.nasa.gov/uars-science/science_highlights1.html. Acesso em: 17 fev. 2025. Citado na p. 17.

GOLUB, G. H.; LOAN, C. F. V. **Matrix computations**. 3rd. ed. Baltimore: Johns Hopkins University Press, 1996. Citado na p. 57.

GREWAL, M. S.; ANDREWS, A. P. **Kalman filtering: Theory and practice using MATLAB**. 3rd. ed. Hoboken, NJ: John Wiley & Sons, 2008. Citado na p. 58.

HAMILTON, W. R. On quaternions; or on a new system of imaginaries in algebra. **The London Edinburgh and Dublin Philosophical Magazine and Journal of Science**, v. 25, n. 163, p. 10–13, 1844. Citado na p. 24.

HIDE, C.; MOORE, T.; SMITH, M. Adaptive Kalman filtering for low-cost INS/GPS. **The Journal of Navigation**, Cambridge University Press, v. 56, n. 1, p. 143–152, 2003. Disponível em: https://www.cambridge.org/core/journals/journal-of-navigation/article/abs/adaptive-kalman-filtering-for-lowcost-insgps/7D17D4A198ED56F06C7F590E90467BC3. Citado nas pp. 53, 54, 55 e 57.

HIGHAM, N. J. **Accuracy and stability of numerical algorithms**. 2nd. ed. Philadelphia: SIAM, 2002. Citado na p. 57.

JUNKINS, J. L. C. J. L. **Optimal Estimation of Dynamic Systems**. 2nd. ed. [*S.l.: s.n.*], 2011. 750 p. Citado nas pp. 46, 47, 48, 49 e 58.

KALATA, P. R. The tracking index: A generalized parameter for alpha-beta and alpha-beta-gamma target trackers. **IEEE Transactions on Aerospace and Electronic Systems**, v. 30, n. 2, p. 471–488, 1994. Citado na p. 48.

KALMAN, R. E. A new approach to linear filtering and prediction problems. **Transactions of the ASME–Journal of Basic Engineering**, v. 82, n. Series D, p. 35–45, 1960. Citado nas pp. 53 e 57.

KARLGAARD, C. D.; SCHAUB, H. Nonlinear regression huber-kalman filtering and fixed-interval smoothing. **Journal of Guidance, Control, and Dynamics**, v. 30, n. 2, p. 405–412, 2007. Citado nas pp. 53, 54, 55, 56 e 57.

KOCH, K. R.; YANG, Y. Robust Kalman filter for rank deficient observation models. **Journal of Geodesy**, Springer, v. 72, n. 8, p. 436–441, 1998. Citado nas pp. 53 e 55.

LEFFERTS, E. J.; MARKLEY, F. L.; SHUSTER, M. D. Kalman filtering for spacecraft attitude estimation. **Journal of Guidance, Control, and Dynamics**, v. 5, n. 5, p. 417–429, 1982. Citado nas pp. 48, 49, 50, 51, 52, 54, 56 e 57.

MAESSCHALCK, R. D.; JOUAN-RIMBAUD, D.; MASSART, D. L. The Mahalanobis distance. **Chemometrics and Intelligent Laboratory Systems**, Elsevier, v. 50, n. 1, p. 1–18, 2000. Citado nas pp. 53 e 55.

MARKLEY, F. L. Attitude determination and parameter estimation using vector observations: Theory. *In*: **The Journal of the Astronautical Sciences**. [*S.l.: s.n.*], 1988. v. 36, n. 3, p. 245–258. Citado na p. 62.

MARKLEY, F. L. Attitude determination using vector observations: A fast optimal matrix algorithm. **The Journal of the Astronautical Sciences**, v. 41, n. 2, 1993. Citado na p. 45.

MARKLEY, F. L. Attitude error representations for Kalman filtering. **Journal of Guidance, Control, and Dynamics**, v. 26, n. 2, p. 311–317, 2003. Citado nas pp. 53, 54 e 56.

MARKLEY, F. L.; CHENG, Y.; CRASSIDIS, J. L.; OSHMAN, Y. Averaging quaternions. **Journal of Guidance, Control, and Dynamics**, v. 30, n. 4, p. 1193–1197, 2007. Citado nas pp. 61 e 62.

MARKLEY, F. L.; CRASSIDIS, J. L. **Fundamentals of spacecraft attitude determination and control**. New York: Springer, 2014. Citado nas pp. 56 e 58.

MARKLEY F. L.; CRASSIDIS, J. L. **Fundamentals of Spacecraft Attitude Determination and Control**. 1st. ed. [*S.l.: s.n.*], 2014. 1–495 p. Citado nas pp. 15, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 39, 40, 41, 43, 44, 47, 48, 49, 51, 52, 53, 54, 56, 57 e 58.

MAUS, S.; MACMILLAN, S.; MCLEAN, S.; HAMILTON, B.; THOMSON, A.; NAIR, M.; ROLLINS, C. The US/UK world magnetic model for 2010-2015. **NOAA Technical Report NESDIS/NGDC**, 2010. National Geophysical Data Center, Boulder, CO. Citado na p. 43.

MAYBECK, P. S. **Stochastic models, estimation, and control**. New York: Academic Press, 1979. v. 1. Citado nas pp. 53, 55 e 57.

MOHAMED, A. H.; SCHWARZ, K. P. Adaptive Kalman filtering for INS/GPS. **Journal of Geodesy**, Springer, v. 73, n. 4, p. 193–203, 1999. Disponível em: https://link.springer.com/article/10.1007/s001900050236. Citado nas pp. 53, 54, 55 e 57.

MUNGIGUERRA, S. Mini-IRENE, a successful re-entry flight of a deployable heatshield capsule. **Materials research proceedings**, v. 37, p. 530–533, 10 2023. Disponível em: https://doi.org/10.21741/9781644902813-116. Citado nas pp. 8 e 19.

National Imagery and Mapping Agency. **Department of Defense World Geodetic System 1984: Its Definition and Relationships with Local Geodetic Systems**. 3rd. ed. [*S.l.*], 2000. Citado na p. 38.

PICONE, J. M. *et al.* NRLMSISE-00 empirical model of the atmosphere. **Journal of Geophysical Research**, v. 107, n. A12, 2002. Citado nas pp. 9, 33, 34, 37 e 44.

PITTELKAU, M. E. Rotation vector in attitude estimation. *In*: **AIAA Guidance, Navigation, and Control Conference**. [*S.l.: s.n.*], 2003. p. 5542. Citado nas pp. 54, 55 e 56.

PRESS, W. H.; TEUKOLSKY, S. A.; VETTERLING, W. T.; FLANNERY, B. P. **Numerical recipes in C: The art of scientific computing**. 2nd. ed. Cambridge: Cambridge University Press, 1992. Citado na p. 57.

SATCATALOG. **Magnetômetro de 3 eixos**. 2024. 2024. Acessado em: 04 nov. 2025. Disponível em: https://www.satcatalog.com/component/tam-1-series/. Citado nas pp. 8 e 31.

SHEPPERD, S. W. Quaternion from rotation matrix. **Journal of Guidance and Control**, v. 1, n. 3, p. 223–224, 1978. Citado na p. 42.

SHOEMAKE, K. Animating rotation with quaternion curves. *In*: ACM. **ACM SIGGRAPH Computer Graphics**. [*S.l.*], 1985. v. 19, n. 3, p. 245–254. Citado na p. 48.

SHUSTER, M. D. A Survey of Attitude Representations. **The Journal of the Astronautical Sciences**, v. 41, n. 4, p. 439–517, 1993. Citado nas pp. 21, 26 e 40.

SHUSTER M. D.; OH, S. D. Three-axis attitude determination from vector observations. **Journal of Guidance and Control**, v. 4, n. 1, p. 70–77, 1981. Citado nas pp. 45, 46, 53, 54 e 56.

SIMON, D. **Optimal state estimation Kalman, HOO and nonlinear approches**. [*s.n.*], 2006. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.705.3475. Citado nas pp. 46 e 47.

TIKHONOV, A. N.; ARSENIN, V. Y. **Solutions of ill-posed problems**. Washington, DC: V. H. Winston & Sons, 1977. Citado na p. 57.

VALLADO, D. A. **Fundamentals of Astrodynamics and Applications**. 4th. ed. [*S.l.*]: Microcosm Press, 2013. Citado nas pp. 33, 34, 35, 36, 38, 41, 42, 43 e 44.

WAHBA, G. A least squares estimate of satellite attitude. **SIAM Review**, v. 7, n. 3, p. 409, 1965. Citado nas pp. 45, 54 e 56.

WERTZ, J. **Spacecraft Attitude Determination and Control**. 1st. ed. [*S.l.: s.n.*], 1978. 1–858 p. Citado nas pp. 29, 30, 31, 39, 40, 41, 42, 43, 44, 47, 54 e 56.

WERTZ, J. R.; EVERETT, D. F.; PUSCHELL, J. J. **Space mission engineering: The new SMAD**. Hawthorne, CA: Microcosm Press, 2011. Citado na p. 58.

WHITAKER, J. S.; HAMILL, T. M. Ensemble Data Assimilation without Perturbed Observations. **Monthly Weather Review**, v. 130, n. 7, p. 1913–1924, 2002. Citado na p. 60.

XIE, Y. e. a. **Spacecraft Dynamics and Control**. 1st. ed. [*S.l.: s.n.*], 2022. 160 p. Citado nas pp. 8, 21, 22, 23 e 28.

YANG, Y.; HE, H.; XU, G. Adaptively robust filtering for kinematic geodetic positioning. **Journal of Geodesy**, Springer, v. 75, n. 2-3, p. 109–116, 2001. Citado nas pp. 53, 54, 55 e 57.

Anexos

Anexo A – Códigos

```
clear;
close all;
clc;

mu = 398600.64;
RUN_QUEST = true;
RUN_ALPHA = true;
RUN_EKF = true;
RUN_RAKF = true;
RUN_ENKF = true;
RUN_ENSRF = true;

if ~(RUN_QUEST || RUN_ALPHA || RUN_EKF || RUN_RAKF || RUN_ENKF || RUN_ENSRF)
    error('Pelo menos um filtro deve estar habilitado!');
end

fprintf('\n========== FILTROS HABILITADOS ==========\n');
if RUN_QUEST, fprintf(' [X] QUEST\n'); else, fprintf(' [ ] QUEST
    ↪ (DESABILITADO)\n'); end
if RUN_ALPHA, fprintf(' [X] Filtro Alfa\n'); else, fprintf(' [ ] Filtro Alfa
    ↪ (DESABILITADO)\n'); end
if RUN_EKF, fprintf(' [X] EKF\n'); else, fprintf(' [ ] EKF (DESABILITADO)\n'); end
if RUN_RAKF, fprintf(' [X] RAKF\n'); else, fprintf(' [ ] RAKF (DESABILITADO)\n');
    ↪ end
if RUN_ENKF, fprintf(' [X] EnKF\n'); else, fprintf(' [ ] EnKF (DESABILITADO)\n');
    ↪ end
if RUN_ENSRF, fprintf(' [X] EnSRF\n'); else, fprintf(' [ ] EnSRF
    ↪ (DESABILITADO)\n'); end
fprintf('=========================================\n\n');

dt = 60;
max_time = 86400 * 365 * 2;

yr = 2011; mth = 3; day = 31;
hr = 2; minute = 32; sec = 41;
start_datetime = datetime(yr, mth, day, hr, minute, sec, 'TimeZone', 'UTC');

```

```
mass_sat = 24;
shield_diameter = 1.0;
shield_radius = shield_diameter / 2;
area_drag_phase2 = pi * shield_radius^2;
cd_drag_phase2 = 1.9;
area_drag_phase1 = 0.15;
cd_drag_phase1 = 2.2;
deploy_altitude_km = 250;
sat_params = struct( ...
    'mass', mass_sat, ...
    'deploy_altitude_km', deploy_altitude_km, ...
    'shield_closed', struct('area', area_drag_phase1, 'Cd', cd_drag_phase1), ...
    'shield_open', struct('area', area_drag_phase2, 'Cd', cd_drag_phase2));

f107_avg = 150;
f107_daily = 150;
ap_daily = 15;
atm_params = struct('f107a', f107_avg, 'f107', f107_daily, 'ap', ap_daily);

earth_radius_km = 6378.137;
initial_altitude_km = 400;
target_altitude_km = 120;
a = earth_radius_km + initial_altitude_km;
initial_position = [a, 0, 0];
initial_velocity = [0, sqrt(mu / a), 0];
initial_state = [initial_position, initial_velocity];

fprintf('Iniciando simulação de decaimento orbital de 400 km para 120 km...\n');
fprintf('Nota: Decaimento em 400 km de altitude élento devido  baixa densidade
    ↪ atmosférica.\n');
fprintf('A simulação pode levar tempo considerável.\n\n');

[t,pos,vel] = propagateDecayOrbit(dt, max_time, initial_state, start_datetime, ...
    target_altitude_km, mu, earth_radius_km, sat_params, atm_params);
length_t = numel(t);
tf = t(end);
alt_vec = sqrt(sum(pos.^2,2)) - earth_radius_km;

fprintf('Propagação orbital completa: %.2f dias\n', tf/86400);
fprintf('Total de passos de tempo: %d\n\n', length_t);

```

```
q = zeros(length_t, 4);
qe = zeros(length_t, 4);
sig3 = zeros(length_t, 3);
qer = zeros(length_t, 4);
q_alp = zeros(length_t, 4);
alpha = zeros(length_t, 1);
sig3qer = zeros(length_t, 3);
P_meas = NaN(3, 3, length_t);
sig3alp = zeros(length_t, 3);

h_vec = cross(pos,vel,2);
h_norm = sqrt(sum(h_vec.^2, 2));
r_norm = sqrt(sum(pos.^2, 2));
orbital_rate = h_norm ./ max(r_norm.^2, realmin);
ang_vel=[zeros(length_t,1) -orbital_rate zeros(length_t,1)];

sigu = sqrt(10) * 1e-10;
sigv = sqrt(10) * 1e-7;
num_g = dt * [1 1];
den_g = 2 * [1 -1];
[phi_g, gam_g, c_g, d_g] = tf2ss(num_g, den_g);

bias_init = 0.1 * pi / 180 / 3600 / dt;
noise_scale = sigu / sqrt(dt);
bias1 = dlsim(phi_g, gam_g, c_g, d_g, noise_scale * randn(length_t, 1), bias_init);
bias2 = dlsim(phi_g, gam_g, c_g, d_g, noise_scale * randn(length_t, 1), bias_init);
bias3 = dlsim(phi_g, gam_g, c_g, d_g, noise_scale * randn(length_t, 1), bias_init);
bias = [bias1, bias2, bias3];

arw_component = sigv^2 / dt;
rw_component = (sigu^2 * dt) / 12;
gyro_noise_var = arw_component + rw_component;
gyro_noise = sqrt(gyro_noise_var) * randn(length_t, 3);
wgm = ang_vel + gyro_noise + bias;

for i = 1:length_t
    pos_norm = norm(pos(i,:));
    lz = -pos(i,:) / pos_norm;

    h_vec_i = cross(pos(i,:), vel(i,:));
    h_norm_i = norm(h_vec_i);
```

```
    ly = -h_vec_i / h_norm_i;
    lx = cross(ly, lz);

    true_att = [lx; ly; lz];

    if i == 1
        q(1,:) = extract(true_att);
    else
        vel_norm = norm(ang_vel(i,:));
        if vel_norm > 1e-10
            half_angle = 0.5 * vel_norm * dt;
            co = cos(half_angle);
            si = sin(half_angle);

            axis_n = ang_vel(i,:) / vel_norm;
            qw1 = axis_n(1) * si;
            qw2 = axis_n(2) * si;
            qw3 = axis_n(3) * si;
            qw4 = co;

            om = [qw4, qw3, -qw2, qw1;
                 -qw3, qw4, qw1, qw2;
                  qw2, -qw1, qw4, qw3;
                 -qw1, -qw2, -qw3, qw4];
            q_prev = q(i-1, 1:4)';
            q_new = om * q_prev;
            q(i, 1:4) = q_new';
        else
            q(i,:) = q(i-1,:);
        end
    end
end

mag_i=mag_field(pos,yr,mth,day,hr,minute,sec,10,dt,1);
mag_b=i2b(q,mag_i);

sun_i=solar(yr,mth,day,hr,minute,sec,dt,tf);
sun_b=i2b(q,sun_i);
t1=[-0.5736 0 -0.8192;0.4096 0.866 -0.2868;0.7094 -0.5 -0.4967];
t2=[-0.5736 0 0.8192;-0.4096 0.866 -0.2868;-0.7094 -0.5 -0.4967];
sun_sen1=(t1*sun_b')';
```

```
155 sun_sen2=(t2*sun_b')';
156 sun_avail1=eclipse(pos,sun_sen1);
157 sun_avail2=eclipse(pos,sun_sen2);
158 avail1=find(sun_avail1==1);
159 avail2=find(sun_avail2==1);
160 sun_avail=zeros(length_t,1);
161 sun_avail(avail1)=1;sun_avail(avail2)=1;
162
163 sig_sun=0.05*pi/180;
164 alp1 = -sun_sen1(:,1) ./ sun_sen1(:,3);
165 beta1 = -sun_sen1(:,2) ./ sun_sen1(:,3);
166 alp1 = alp1 + sig_sun * randn(length_t, 1);
167 beta1 = beta1 + sig_sun * randn(length_t, 1);
168 sensor_vec = [alp1, beta1, -ones(length_t, 1)];
169 sensor_norm = sqrt(sum(sensor_vec.^2, 2));
170 sun_sen1m = sensor_vec ./ repmat(sensor_norm, 1, 3);
171
172 sun_sen1m(:,1) = sign(sun_sen1(:,1)) .* abs(sun_sen1m(:,1));
173 sun_sen1m(:,2) = sign(sun_sen1(:,2)) .* abs(sun_sen1m(:,2));
174 sun_sen1m(:,3) = sign(sun_sen1(:,3)) .* abs(sun_sen1m(:,3));
175 sun_b1m = (t1 \ sun_sen1m')';
176
177 alp2 = -sun_sen2(:,1) ./ sun_sen2(:,3);
178 beta2 = -sun_sen2(:,2) ./ sun_sen2(:,3);
179 noise2_alpha = sig_sun * randn(length_t, 1);
180 noise2_beta = sig_sun * randn(length_t, 1);
181 alp2 = alp2 + noise2_alpha;
182 beta2 = beta2 + noise2_beta;
183 sensor2_vec = [alp2, beta2, -ones(length_t, 1)];
184 norm2 = sqrt(sum(sensor2_vec.^2, 2));
185 sun_sen2m = sensor2_vec ./ repmat(norm2, 1, 3);
186
187 sun_sen2m(:,1) = sign(sun_sen2(:,1)) .* abs(sun_sen2m(:,1));
188 sun_sen2m(:,2) = sign(sun_sen2(:,2)) .* abs(sun_sen2m(:,2));
189 sun_sen2m(:,3) = sign(sun_sen2(:,3)) .* abs(sun_sen2m(:,3));
190 sun_b2m = (t2 \ sun_sen2m')';
191
192 sig_tam = 50;
193 mag_bm = mag_b + sig_tam * randn(length_t, 3);
194 fprintf('\n=== Modelo de Ruído do Magnetmetro ===\n');
195 fprintf('Ruído gaussiano constante:  = %.1f nT\n\n', sig_tam);
```

```
outlier_prob = 0.05;
outlier_magnitude = 10;
outlier_mask = rand(length_t, 1) < outlier_prob;
num_outliers = sum(outlier_mask);

for i = 1:length_t
    if outlier_mask(i)
        outlier_noise = outlier_magnitude * sig_tam * randn(1,3);
        mag_bm(i,:) = mag_b(i,:) + outlier_noise;
    end
end

fprintf('Injeção de Outliers: %d medições (%.1f%%) corrompidas\n', num_outliers,
    ↪ 100*outlier_prob);
fprintf('Magnitude do ruído outlier: %.1fx nível nominal\n\n', outlier_magnitude);

for i = 1:length_t
    sun_b1_cross = -crossm(sun_b1m(i,:));
    sun_b2_cross = -crossm(sun_b2m(i,:));
    mag_bm_cross = -crossm(mag_bm(i,:));
    sun_i_cross = crossm(sun_i(i,:));
    mag_i_cross = crossm(mag_i(i,:));

    om_dss1 = [sun_b1_cross sun_b1m(i,:)'; -sun_b1m(i,:) 0];
    om_dss2 = [sun_b2_cross sun_b2m(i,:)'; -sun_b2m(i,:) 0];
    om_tam = [mag_bm_cross mag_bm(i,:)'; -mag_bm(i,:) 0];
    gam_dss = [sun_i_cross sun_i(i,:)'; -sun_i(i,:) 0];
    gam_tam = [mag_i_cross mag_i(i,:)'; -mag_i(i,:) 0];
    gam_tamr = gam_tam;
    if sun_avail1(i) == 1 & sun_avail2(i) == 1
        sig_tam_i = sig_tam;
        weight_sun = 1 / sig_sun^2;
        weight_mag = 1 / sig_tam_i^2;

        k_mat = -weight_sun * om_dss1 * gam_dss ...
               -weight_sun * om_dss2 * gam_dss ...
               -weight_mag * om_tam * gam_tamr;

        [v_eig, e_eig] = eig(k_mat);
        [ii, jj] = max(diag(e_eig));
        q_opt = v_eig(:, jj)';
```

```
236         qer(i,:) = q_opt / norm(q_opt);
237 
238         sun1_cross = crossm(sun_b1m(i,:));
239         sun2_cross = crossm(sun_b2m(i,:));
240         mag_cross = crossm(mag_bm(i,:));
241         meas_mat = weight_sun * sun1_cross^2 + weight_sun * sun2_cross^2 +
                ↪ weight_mag * mag_cross^2;
242      p_bad=regularized_inverse(meas_mat, 1e-6);
243      P_meas(:,:,i)=p_bad;
244      sig3qer(i,:)=(sqrt(diag(p_bad)))'*3*180/pi;
245      if dot(qer(i,:), q(i,:)) < 0, qer(i,:) = -qer(i,:); end
246  end
247     if sun_avail1(i) == 1 & sun_avail2(i) == 0
248         sig_tam_i = sig_tam;
249         k_mat = -om_dss1 * gam_dss / sig_sun^2 - om_tam * gam_tamr / sig_tam_i^2;
250 
251         [v_eig, e_eig] = eig(k_mat);
252         [~, jj] = max(diag(e_eig));
253         qer(i,:) = v_eig(:, jj)' / norm(v_eig(:, jj));
254 
255         meas_mat = crossm(sun_b1m(i,:))^2 / sig_sun^2 + crossm(mag_bm(i,:))^2 /
                ↪ sig_tam_i^2;
256      p_bad=regularized_inverse(meas_mat, 1e-6);
257      P_meas(:,:,i)=p_bad;
258      sig3qer(i,:)=(sqrt(diag(p_bad)))'*3*180/pi;
259      if dot(qer(i,:), q(i,:)) < 0, qer(i,:) = -qer(i,:); end
260  end
261     if sun_avail1(i) == 0 & sun_avail2(i) == 1
262         sig_tam_i = sig_tam;
263 
264         sun_weight = 1 / sig_sun^2;
265         mag_weight = 1 / sig_tam_i^2;
266         k_mat = -sun_weight * om_dss2 * gam_dss - mag_weight * om_tam * gam_tamr;
267 
268         [v_eig, e_eig] = eig(k_mat);
269         diagonal_vals = diag(e_eig);
270         [~, max_idx] = max(diagonal_vals);
271 
272         eigenvector = v_eig(:, max_idx);
273         qer(i,:) = eigenvector' / norm(eigenvector);
274 
```

```
275         meas_mat = sun_weight * crossm(sun_b2m(i,:))^2 + mag_weight *
                ↪ crossm(mag_bm(i,:))^2;
276       p_bad=regularized_inverse(meas_mat, 1e-6);
277       P_meas(:,:,i)=p_bad;
278       sig3qer(i,:)=(sqrt(diag(p_bad)))'*3*180/pi;
279       if dot(qer(i,:), q(i,:)) < 0, qer(i,:) = -qer(i,:); end
280    end
281    if sun_avail1(i) == 0 & sun_avail2(i) == 0
282       qer(i,:)=NaN;
283       P_meas(:,:,i)=NaN(3);
284       sig3qer(i,:)=NaN(1,3);
285    end
286
287 end
288
289 if RUN_QUEST
290     qerr=quat_err(q,qer);errer=qerr(:,1:3)*2*180/pi;
291 end
292 k0 = find(~any(isnan(qer),2), 1, 'first');
293 if isempty(k0)
294     error('Nenhuma solução QUEST válida para inicializar os filtros');
295 end
296 q0 = qer(k0,:);
297 if ~RUN_QUEST
298     errer = NaN(length_t, 3);
299 end
300 if ~RUN_ALPHA
301     errea = NaN(length_t, 3);
302     q_alp = NaN(length_t, 4);
303     sig3alp = NaN(length_t, 3);
304 end
305 if ~RUN_EKF
306     ekf_error = NaN(length_t, 3);
307     q_ekf = NaN(length_t, 4);
308     sig3ekf = NaN(length_t, 3);
309     b_ekf = NaN(length_t, 3);
310 end
311 if ~RUN_RAKF
312     rakf_error = NaN(length_t, 3);
313     q_rakf = NaN(length_t, 4);
314     sig3rakf = NaN(length_t, 3);
```

```
315     b_rakf = NaN(length_t, 3);
316     lambda_rakf = NaN(length_t, 1);
317     mahal_dist_hist = NaN(length_t, 1);
318     geom_quality_hist = NaN(length_t, 1);
319     outlier_flags = false(length_t, 1);
320     Rk_rakf_hist = NaN(3, 3, length_t);
321 end
322 if ~RUN_ENKF
323     enkf_error = NaN(length_t, 3);
324     q_enkf = NaN(length_t, 4);
325     sig3enkf = NaN(length_t, 3);
326     b_enkf = NaN(length_t, 3);
327 end
328 if ~RUN_ENSRF
329     ensrf_error = NaN(length_t, 3);
330     q_ensrf = NaN(length_t, 4);
331     sig3ensrf = NaN(length_t, 3);
332     b_ensrf = NaN(length_t, 3);
333 end
334
335 if RUN_ALPHA
336     q_alp(1,:) = q0;
337     alpha0 = 0.1;
338     mag_norm = mag_i(1,:) / norm(mag_i(1,:));
339     sun_norm = sun_i(1,:) / norm(sun_i(1,:));
340     geom_factor = norm(cross(mag_norm, sun_norm))^2;
341     alpha(1) = alpha0 * geom_factor;
342
343     process_cov_alpha = diag([(0.005*pi/180)^2, (0.005*pi/180)^2,
            ↪ (0.005*pi/180)^2]);
344     Q_alp = process_cov_alpha;
345
346     if sun_avail(1) == 1 && all(isfinite(diag(P_meas(:,:,1))))
347         P_alp = P_meas(:,:,1);
348     else
349         P_alp = (0.5 * pi/180)^2 * eye(3);
350     end
351     sig3alp(1,:) = sqrt(diag(P_alp))' * 3 * 180/pi;
352
353     fprintf('Executando Filtro Alfa...\n');
354     tic;
```

```
for i = 1:length_t-1
    ww=norm(wgm(i,:));
    if ww>1e-8
        co=cos(0.5*ww*dt); si=sin(0.5*ww*dt);
        n=wgm(i,:)/ww; qw1=n(1)*si; qw2=n(2)*si; qw3=n(3)*si; qw4=co;
    else
        qw1=0; qw2=0; qw3=0; qw4=1;
    end
    om=[qw4 qw3 -qw2 qw1; -qw3 qw4 qw1 qw2; qw2 -qw1 qw4 qw3; -qw1 -qw2 -qw3 qw4];
    q_prop=(om*q_alp(i,:)'); q_prop=q_prop/norm(q_prop);

    if sun_avail(i+1) == 1
        alpha(i+1)=alpha0*norm(cross( ...
            mag_i(i+1,:)/norm(mag_i(i+1,:)), ...
            sun_i(i+1,:)/norm(sun_i(i+1,:)) ))^2;
    else
        alpha(i+1)=0;
    end
    a=alpha(i+1);

    F=eye(3)-crossm(wgm(i,:))*dt;
    P_prop=F*P_alp*F'+Q_alp*dt;
    Rz=P_meas(:,:,i+1);
    if a>0 && all(isfinite(diag(Rz)))
        P_alp=(1-a)^2*P_prop + a^2*Rz;
    else
        P_alp=P_prop;
    end
    sig3alp(i+1,:)=(sqrt(diag(P_alp)))'*3*180/pi;

    if a>0
        q_meas = qer(i+1,:);
        if dot(q_meas, q_prop') < 0, q_meas = -q_meas; end
        q_alp(i+1,:)=(1-a)*q_prop' + a*q_meas;
    else
        q_alp(i+1,:)=q_prop';
    end
    q_alp(i+1,:)=q_alp(i+1,:)/norm(q_alp(i+1,:));

end

```

```
396     time_alpha = toc;
397     fprintf('Filtro Alfa completo em %.2f segundos\n', time_alpha);
398     qerr_a=quat_err(q,q_alp);errea=qerr_a(:,1:3)*2*180/pi;
399 else
400     time_alpha = 0;
401 end
402 
403 quat_mult = @(q, r) [q(4)*r(1:3) + r(4)*q(1:3) + cross(q(1:3), r(1:3)), ...
404                      q(4)*r(4) - dot(q(1:3), r(1:3))];
405 quat_inv = @(q) [-q(1:3), q(4)];
406 normalize_quat = @(q) q / norm(q);
407 delta_to_quat = @(delta) normalize_quat([0.5*delta, 1]);
408 
409 attitude_noise = [(0.01*pi/180)^2 (0.005*pi/180)^2 (0.005*pi/180)^2];
410 bias_noise = (sigu)^2 * eye(3);
411 Q = blkdiag(diag(attitude_noise), bias_noise);
412 R_nominal = diag([(0.1 * pi/180)^2, (0.1 * pi/180)^2, (0.1 * pi/180)^2]);
413 
414 if RUN_EKF
415     fprintf('Executando EKF...\n');
416     tic
417 
418     q_ekf = zeros(length_t, 4);
419     q_ekf(1,:) = q0;
420 
421     att_cov_init = (0.1*pi/180)^2 * eye(3);
422     bias_cov_init = (0.1*pi/180/3600)^2 * eye(3);
423     P = blkdiag(att_cov_init, bias_cov_init);
424     b_ekf = zeros(length_t, 3);
425 
426 sig3ekf = zeros(length_t, 3);
427 sig3ekf(1,:) = (sqrt(diag(P(1:3,1:3))))' * 3 * 180/pi;
428 ekf_error = zeros(length_t, 3);
429 
430 for i = 1:length_t-1
431     omega = wgm(i,:) - b_ekf(i,:);
432     ww = norm(omega);
433 
434     if ww > 1e-8
435         half_angle = 0.5 * ww * dt;
436         co = cos(half_angle);
```

```
437         si = sin(half_angle);
438         n = omega / ww;
439         dq = [n(1)*si, n(2)*si, n(3)*si, co];
440     else
441         dq = [0, 0, 0, 1];
442     end
443 
444     q_pred = quat_mult(q_ekf(i,:), dq);
445     q_pred = normalize_quat(q_pred);
446 
447     F = [eye(3)-crossm(omega)*dt -eye(3)*dt; zeros(3,3) eye(3)];
448     P_pred = F*P*F' + Q*dt;
449 
450     if ~any(isnan(qer(i+1,:)))
451         q_tilde = quat_mult(qer(i+1,:), quat_inv(q_pred));
452         if q_tilde(4) < 0, q_tilde = -q_tilde; end
453         z = 2 * q_tilde(1:3)';
454 
455         H = [eye(3) zeros(3)];
456         Rk = measurement_covariance_from_quest(P_meas, i+1, R_nominal);
457         Rk = ensure_spd(Rk, (0.01*pi/180)^2*eye(3));
458 
459         S = H*P_pred*H' + Rk;
460         K = (P_pred*H')/S;
461 
462         delta = K*z;
463         delta_theta = delta(1:3);
464         delta_bias = delta(4:6);
465 
466         I6 = eye(6);
467         P = (I6 - K*H)*P_pred*(I6 - K*H)' + K*Rk*K';
468         P = (P + P')/2;
469 
470         dq_update = delta_to_quat(delta_theta');
471         q_update = quat_mult(dq_update, q_pred); q_update =
            ↪ normalize_quat(q_update);
472         b_ekf(i+1,:) = b_ekf(i,:) + delta_bias';
473     else
474         q_update = q_pred;
475         P = P_pred;
476         b_ekf(i+1,:) = b_ekf(i,:);
```

```
477     end
478
479     q_ekf(i+1,:) = q_update;
480     sig3ekf(i+1,:) = (sqrt(diag(P(1:3,1:3))))' * 3 * 180/pi;
481
482     q_err_ekf = quat_mult(q(i+1,:), quat_inv(q_update));
483     if q_err_ekf(4) < 0
484         q_err_ekf = -q_err_ekf;
485     end
486     ekf_error(i+1,:) = 2 * q_err_ekf(1:3) * 180/pi;
487 end
488
489     time_ekf = toc;
490     fprintf('EKF completo em %.2f segundos\n', time_ekf);
491 else
492     time_ekf = 0;
493 end
494
495 if RUN_RAKF
496     fprintf('Executando RAKF (EKF adaptativo robusto com rejeição de
        ↪ outliers)...\n');
497     tic
498
499     q_rakf = zeros(length_t,4);
500     q_rakf(1,:) = q0;
501 b_rakf = zeros(length_t,3);
502 P_rakf = blkdiag(diag([(0.1*pi/180)^2 (0.1*pi/180)^2 (0.1*pi/180)^2]),
    ↪ (0.1*pi/180/3600)^2*eye(3));
503 Q_rakf = Q;
504
505 sig3rakf = zeros(length_t,3);
506 sig3rakf(1,:) = (sqrt(diag(P_rakf(1:3,1:3))))' * 3 * 180/pi;
507 rakf_error = zeros(length_t,3);
508 lambda_rakf = ones(length_t,1);
509 mahal_dist_hist = NaN(length_t,1);
510 geom_quality_hist = NaN(length_t,1);
511 outlier_flags = false(length_t,1);
512 Rk_rakf_hist = zeros(3,3,length_t);
513
514 chi2_threshold = 5.0;
515 geom_threshold = 0.15;
```

```
R_floor_rakf = (0.01*pi/180)^2 * eye(3);
eps_regularization = 1e-6;
Rk_rakf_hist(:,:,1) = R_nominal;

for i = 1:length_t-1
    omega = wgm(i,:) - b_rakf(i,:);
    ww = norm(omega);
    if ww > 1e-8
        co = cos(0.5 * ww * dt);
        si = sin(0.5 * ww * dt);
        n = omega / ww;
        dq = [n(1)*si, n(2)*si, n(3)*si, co];
    else
        dq = [0, 0, 0, 1];
    end
    q_pred = quat_mult(q_rakf(i,:), dq);
    q_pred = normalize_quat(q_pred);

    F = [eye(3)-crossm(omega)*dt -eye(3)*dt; zeros(3,3) eye(3)];
    P_pred = F*P_rakf*F' + Q_rakf*dt;

    if ~any(isnan(qer(i+1,:)))
        q_tilde = quat_mult(qer(i+1,:), quat_inv(q_pred));
        if q_tilde(4) < 0
            q_tilde = -q_tilde;
        end
        z = 2 * q_tilde(1:3)';

        H = [eye(3) zeros(3)];
        Rk = measurement_covariance_from_quest(P_meas, i+1, R_nominal);
        Rk = ensure_spd(Rk, R_floor_rakf);

        S_prelim = H*P_pred*H' + Rk;

        try
            S_inv = pinv(S_prelim);
            z_norm_sq = z' * S_inv * z;
            if isfinite(z_norm_sq) && z_norm_sq >= 0
                mahal_dist = sqrt(z_norm_sq);
            else
                mahal_dist = norm(z) / sqrt(trace(S_prelim));
```

```
557             end
558         catch
559             mahal_dist = norm(z) / sqrt(trace(S_prelim));
560         end
561         mahal_dist_hist(i+1) = mahal_dist;
562 
563         if sun_avail(i+1) == 1
564             mag_norm = mag_i(i+1,:) / norm(mag_i(i+1,:));
565             sun_norm = sun_i(i+1,:) / norm(sun_i(i+1,:));
566             geom_quality = norm(cross(mag_norm, sun_norm));
567         else
568             geom_quality = 0;
569         end
570         geom_quality_hist(i+1) = geom_quality;
571 
572         Rk_adapted = Rk;
573         scale_factor = 1.0;
574         if isfinite(mahal_dist) && mahal_dist > 0
575             if mahal_dist > chi2_threshold
576                 outlier_flags(i+1) = true;
577                 excess_ratio = min(1.0, (mahal_dist - chi2_threshold) /
                        ↪ chi2_threshold);
578                 scale_factor = max(scale_factor, 2.0 + 3.0 * excess_ratio);
579             elseif mahal_dist > chi2_threshold * 0.8
580                 scale_factor = max(scale_factor, 1.2 + 0.8 * (mahal_dist -
                        ↪ chi2_threshold*0.8)/(chi2_threshold*0.2));
581             end
582 
583             if geom_quality < geom_threshold
584                 geom_scale = 1.2 + 1.3 * (1.0 - geom_quality/geom_threshold);
585                 scale_factor = max(scale_factor, geom_scale);
586             end
587         end
588 
589         lambda_rakf(i+1) = scale_factor;
590         Rk = scale_factor * Rk_adapted;
591 
592         Rk_rakf_hist(:,:,i+1) = Rk;
593 
594         S = H*P_pred*H' + Rk;
595         K = (P_pred*H') * pinv(S);
```

```
596
597         delta = K*z;
598         delta_theta = delta(1:3);
599         delta_bias = delta(4:6);
600
601         I6 = eye(6);
602         P_rakf = (I6 - K*H)*P_pred*(I6 - K*H)' + K*Rk*K';
603         P_rakf = (P_rakf + P_rakf')/2;
604
605         max_att_var = (2*pi/180)^2;
606         max_bias_var = (0.5*pi/180/3600)^2;
607         for j = 1:3
608             P_rakf(j,j) = min(P_rakf(j,j), max_att_var);
609             P_rakf(j+3,j+3) = min(P_rakf(j+3,j+3), max_bias_var);
610         end
611
612         P_rakf = (P_rakf + P_rakf')/2;
613
614         dq_update = delta_to_quat(delta_theta');
615         q_update = quat_mult(dq_update, q_pred);
616         q_update = normalize_quat(q_update);
617         b_rakf(i+1,:) = b_rakf(i,:) + delta_bias';
618     else
619         q_update = q_pred;
620         P_rakf = P_pred;
621         b_rakf(i+1,:) = b_rakf(i,:);
622         lambda_rakf(i+1) = 1.0;
623         Rk_rakf_hist(:,:,i+1) = Rk_rakf_hist(:,:,i);
624         mahal_dist_hist(i+1) = NaN;
625         geom_quality_hist(i+1) = NaN;
626     end
627
628     q_rakf(i+1,:) = q_update;
629     sig3rakf(i+1,:) = (sqrt(diag(P_rakf(1:3,1:3))))' * 3 * 180/pi;
630
631     q_err_rakf = quat_mult(q(i+1,:), quat_inv(q_update));
632     if q_err_rakf(4) < 0
633         q_err_rakf = -q_err_rakf;
634     end
635     rakf_error(i+1,:) = 2 * q_err_rakf(1:3) * 180/pi;
636 end
```

```
637
638     time_rakf = toc;
639     fprintf('RAKF completo em %.2f segundos\n', time_rakf);
640     fprintf(' Outliers detectados: %d (%.1f%%)\n', sum(outlier_flags),
            ↪ 100*sum(outlier_flags)/length_t);
641 else
642     time_rakf = 0;
643 end
644
645 if RUN_ENSRF
646     fprintf('Executando EnSRF...\n');
647     tic
648
649     quat_to_dcm = @(q) ...
650         [q(1)^2 - q(2)^2 - q(3)^2 + q(4)^2, 2*(q(1)*q(2) + q(3)*q(4)),
                ↪ 2*(q(1)*q(3) - q(2)*q(4));
651          2*(q(1)*q(2) - q(3)*q(4)), -q(1)^2 + q(2)^2 - q(3)^2 + q(4)^2,
                ↪ 2*(q(2)*q(3) + q(1)*q(4));
652          2*(q(1)*q(3) + q(2)*q(4)), 2*(q(2)*q(3) - q(1)*q(4)), -q(1)^2 - q(2)^2 +
                ↪ q(3)^2 + q(4)^2];
653
654     N_ensrf = 50;
655
656 q_ensrf = zeros(length_t, 4);
657 q_ensrf(1,:) = q0;
658
659 Q_ensrf = diag([(0.01 * pi/180)^2, (0.005 * pi/180)^2, (0.005 * pi/180)^2]);
660 R_att = diag([(0.1 * pi/180)^2, (0.1 * pi/180)^2, (0.1 * pi/180)^2]);
661 R_floor = (0.1*pi/180)^2 * eye(3);
662
663 P_init = diag([(0.1 * pi/180)^2, (0.1 * pi/180)^2, (0.1 * pi/180)^2]);
664 delta_theta_ensemble = mvnrnd(zeros(1,3), P_init, N_ensrf)';
665 q_ensemble = zeros(N_ensrf, 4);
666 for j = 1:N_ensrf
667     dq_j = delta_to_quat(delta_theta_ensemble(:,j)');
668     q_ensemble(j, :) = quat_mult(dq_j, q_ensrf(1,:));
669     q_ensemble(j, :) = normalize_quat(q_ensemble(j, :));
670 end
671 b_ensemble = mvnrnd(zeros(1,3), (0.1 * pi/180/3600)^2 * eye(3), N_ensrf);
672 b_ensrf = zeros(length_t, 3);
673 b_ensrf(1,:) = mean(b_ensemble, 1);
```

```
674 Q_bias_rw = (sigu)^2 * eye(3);
675
676 sig3ensrf = zeros(length_t, 3);
677 ensrf_error = zeros(length_t, 3);
678 sig3ensrf(1,:) = sqrt(diag(P_init))' * 3 * 180/pi;
679
680 for i = 1:length_t-1
681
682     process_noise = mvnrnd(zeros(1,3), Q_ensrf, N_ensrf)';
683     bias_noise = mvnrnd(zeros(1,3), Q_bias_rw * dt, N_ensrf)';
684     q_ensemble_pred = zeros(N_ensrf, 4);
685     b_ensemble_pred = zeros(N_ensrf, 3);
686     for j = 1:N_ensrf
687         omega_j = wgm(i,:) - b_ensemble(j,:);
688         ww_j = norm(omega_j);
689         if ww_j > 1e-8
690             co_j = cos(0.5 * ww_j * dt); si_j = sin(0.5 * ww_j * dt); n_j = omega_j
                ↪ / ww_j;
691             dq_prop = [n_j(1)*si_j, n_j(2)*si_j, n_j(3)*si_j, co_j];
692         else
693             dq_prop = [0 0 0 1];
694         end
695         q_j_propagated = quat_mult(q_ensemble(j,:), dq_prop);
696         dq_noise = delta_to_quat(process_noise(:,j)');
697         q_ensemble_pred(j,:) = quat_mult(dq_noise, q_j_propagated);
698         q_ensemble_pred(j,:) = normalize_quat(q_ensemble_pred(j,:));
699         b_ensemble_pred(j,:) = b_ensemble(j,:) + bias_noise(:,j)';
700     end
701     q_ensemble = q_ensemble_pred;
702     b_ensemble = b_ensemble_pred;
703
704     meas_valid = ~any(isnan(qer(i+1,:)));
705     if ~meas_valid
706         q_ensrf(i+1,:) = mean_quat(q_ensemble);
707         b_ensrf(i+1,:) = mean(b_ensemble, 1);
708         delta_X_final = zeros(6, N_ensrf);
709         for j=1:N_ensrf
710             q_err_j = quat_mult(q_ensemble(j,:), quat_inv(q_ensrf(i+1,:)));
711             if q_err_j(4) < 0, q_err_j = -q_err_j; end
712             delta_X_final(1:3,j) = 2 * q_err_j(1:3)';
713             delta_X_final(4:6,j) = (b_ensemble(j,:) - b_ensrf(i+1,:))';
```

```
        end
        P_final = (1/(N_ensrf-1)) * (delta_X_final * delta_X_final');
        sig3ensrf(i+1,:) = (sqrt(diag(P_final(1:3,1:3))))' * 3 * 180/pi;
        q_err_ensrf = quat_mult(q(i+1,:), quat_inv(q_ensrf(i+1,:)));
        if q_err_ensrf(4) < 0, q_err_ensrf = -q_err_ensrf; end
        ensrf_error(i+1,:) = 2 * q_err_ensrf(1:3) * 180/pi;
        continue;
    end

    Rk = R_att;
    Rk = ensure_spd(Rk, R_floor);
    [Rk_chol,pR] = chol(Rk,'lower');
    if pR ~= 0
        Rk = ensure_spd(Rk, R_floor + 1e-9*eye(3));
        Rk_chol = chol(Rk,'lower');
    end

    b_mean_pred = mean(b_ensemble, 1);
    q_mean_pred = mean_quat(q_ensemble);

    X_a = zeros(6, N_ensrf);
    for j = 1:N_ensrf
        q_err_j = quat_mult(q_ensemble(j,:), quat_inv(q_mean_pred));
        if q_err_j(4) < 0, q_err_j = -q_err_j; end
        X_a(1:3,j) = 2 * q_err_j(1:3)';
        X_a(4:6,j) = (b_ensemble(j,:) - b_mean_pred)';
    end

    Y_pred_ensemble = zeros(3, N_ensrf);
    for j = 1:N_ensrf
        q_err_meas = quat_mult(qer(i+1,:), quat_inv(q_ensemble(j,:)));
        if q_err_meas(4) < 0, q_err_meas = -q_err_meas; end
        Y_pred_ensemble(:,j) = 2 * q_err_meas(1:3)';
    end
    y_mean_pred = mean(Y_pred_ensemble, 2);
    Y_a = Y_pred_ensemble - y_mean_pred;
    y_true = zeros(3,1);
    Sf = X_a / sqrt(N_ensrf - 1);
    Sy = Y_a / sqrt(N_ensrf - 1);
    Sy_tilde = Rk_chol \ Sy;
    M = eye(N_ensrf) + Sy_tilde' * Sy_tilde;
```

```
    M = (M+M')/2;
    [LT,pchol] = chol(M,'lower');
    jitter = 1e-9;
    iter = 0;
    while pchol>0 && iter < 5
        M = M + (jitter * 2^iter)*eye(N_ensrf);
        [LT,pchol] = chol(M,'lower');
        iter = iter + 1;
    end
    if pchol>0, LT = chol(M + 1e-6*eye(N_ensrf),'lower'); end
    T = LT \ eye(N_ensrf);

    d = y_true - y_mean_pred;
    d_tilde = Rk_chol \ d;
    wa = LT'\(LT \ (Sy_tilde' * d_tilde));
    delta_mean = Sf * wa;
    delta_theta_mean = delta_mean(1:3)';
    delta_bias_mean = delta_mean(4:6)';
    q_mean_updated = quat_mult(delta_to_quat(delta_theta_mean), q_mean_pred);
    q_mean_updated = normalize_quat(q_mean_updated);
    b_mean_updated = b_mean_pred + delta_bias_mean;

    X_a_updated = X_a * T;
    for j = 1:N_ensrf
        delta_theta_j = X_a_updated(1:3,j)';
        delta_bias_j = X_a_updated(4:6,j)';
        dq_update = delta_to_quat(delta_theta_j);
        q_ensemble(j,:) = quat_mult(dq_update, q_mean_updated);
        q_ensemble(j,:) = normalize_quat(q_ensemble(j,:));
        b_ensemble(j,:) = b_mean_updated + delta_bias_j;
    end

    q_ensrf(i+1,:) = mean_quat(q_ensemble);
    b_ensrf(i+1,:) = mean(b_ensemble, 1);

    delta_X_final = zeros(6, N_ensrf);
    for j=1:N_ensrf
        q_err_j = quat_mult(q_ensemble(j,:), quat_inv(q_ensrf(i+1,:)));
        if q_err_j(4) < 0, q_err_j = -q_err_j; end
        delta_X_final(1:3,j) = 2 * q_err_j(1:3)';
        delta_X_final(4:6,j) = (b_ensemble(j,:) - b_ensrf(i+1,:))';
```

```
796     end
797     P_final = (1/(N_ensrf-1)) * (delta_X_final * delta_X_final');
798     sig3ensrf(i+1,:) = (sqrt(diag(P_final(1:3,1:3))))' * 3 * 180/pi;
799 
800     q_err_ensrf = quat_mult(q(i+1,:), quat_inv(q_ensrf(i+1,:)));
801     if q_err_ensrf(4) < 0, q_err_ensrf = -q_err_ensrf; end
802     ensrf_error(i+1,:) = 2 * q_err_ensrf(1:3) * 180/pi;
803 end
804 
805     time_ensrf = toc;
806     fprintf('EnSRF completo em %.2f segundos\n', time_ensrf);
807 else
808     time_ensrf = 0;
809 end
810 
811 if RUN_ENKF
812     fprintf('Executando EnKF...\n');
813     tic;
814     N_enkf = 50;
815 
816 q_enkf = zeros(length_t,4);
817 q_enkf(1,:) = q0;
818 Q_enkf = diag([(0.01*pi/180)^2, (0.005*pi/180)^2, (0.005*pi/180)^2]);
819 P_init_enkf = diag([(0.1*pi/180)^2, (0.1*pi/180)^2, (0.1*pi/180)^2]);
820 
821 delta_theta_ensemble = mvnrnd(zeros(1,3), P_init_enkf, N_enkf)';
822 q_ensemble_enkf = zeros(N_enkf,4);
823 for j = 1:N_enkf
824     dq_j = delta_to_quat(delta_theta_ensemble(:,j)');
825     q_ensemble_enkf(j,:) = normalize_quat(quat_mult(dq_j, q_enkf(1,:)));
826 end
827 b_ensemble_enkf = mvnrnd(zeros(1,3), (0.1*pi/180/3600)^2 * eye(3), N_enkf);
828 b_enkf = zeros(length_t,3);
829 b_enkf(1,:) = mean(b_ensemble_enkf, 1);
830 Q_bias_enkf = (sigu)^2 * eye(3);
831 
832 sig3enkf = zeros(length_t,3);
833 enkf_error = zeros(length_t,3);
834 sig3enkf(1,:) = sqrt(diag(P_init_enkf))' * 3 * 180/pi;
835 
836 for i = 1:length_t-1
```

```
    proc_noise = mvnrnd(zeros(1,3), Q_enkf, N_enkf)';
    bias_noise = mvnrnd(zeros(1,3), Q_bias_enkf * dt, N_enkf)';
    for j = 1:N_enkf
        omega_j = wgm(i,:) - b_ensemble_enkf(j,:);
        ww_j = norm(omega_j);
        if ww_j > 1e-8
            co_j = cos(0.5*ww_j*dt); si_j = sin(0.5*ww_j*dt); n_j = omega_j/ww_j;
            dq_prop = [n_j(1)*si_j, n_j(2)*si_j, n_j(3)*si_j, co_j];
        else
            dq_prop = [0 0 0 1];
        end
        q_p = quat_mult(q_ensemble_enkf(j,:), dq_prop);
        dq_n = delta_to_quat(proc_noise(:,j)');
        q_ensemble_enkf(j,:) = normalize_quat(quat_mult(dq_n, q_p));
        b_ensemble_enkf(j,:) = b_ensemble_enkf(j,:) + bias_noise(:,j)';
    end

    meas_valid = ~any(isnan(qer(i+1,:)));
    if ~meas_valid
        q_enkf(i+1,:) = mean_quat(q_ensemble_enkf);
        b_enkf(i+1,:) = mean(b_ensemble_enkf, 1);
        DX = zeros(6, N_enkf);
        for j = 1:N_enkf
            q_err = quat_mult(q_ensemble_enkf(j,:), quat_inv(q_enkf(i+1,:)));
            if q_err(4) < 0, q_err = -q_err; end
            DX(1:3,j) = 2 * q_err(1:3)';
            DX(4:6,j) = (b_ensemble_enkf(j,:) - b_enkf(i+1,:))';
        end
        P_samp = (DX*DX')/(N_enkf-1);
        sig3enkf(i+1,:) = sqrt(diag(P_samp(1:3,1:3)))' * 3 * 180/pi;
        q_err_enkf = quat_mult(q(i+1,:), quat_inv(q_enkf(i+1,:)));
        if q_err_enkf(4) < 0, q_err_enkf = -q_err_enkf; end
        enkf_error(i+1,:) = 2 * q_err_enkf(1:3) * 180/pi;
        continue;
    end

    Rk = R_att;
    Rk = ensure_spd(Rk, R_floor);
    [Rk_chol,pR] = chol(Rk,'lower');
    if pR ~= 0
        Rk = ensure_spd(Rk, R_floor + 1e-9*eye(3));
```

```
878         Rk_chol = chol(Rk,'lower');
879     end
880 
881     b_mean_pred = mean(b_ensemble_enkf, 1);
882     q_mean_pred = mean_quat(q_ensemble_enkf);
883 
884     X_a = zeros(6, N_enkf);
885     for j = 1:N_enkf
886         q_err = quat_mult(q_ensemble_enkf(j,:), quat_inv(q_mean_pred));
887         if q_err(4) < 0, q_err = -q_err; end
888         X_a(1:3,j) = 2 * q_err(1:3)';
889         X_a(4:6,j) = (b_ensemble_enkf(j,:) - b_mean_pred)';
890     end
891 
892     y_true = zeros(3,1);
893     m = 3;
894     Y_pred = zeros(m, N_enkf);
895     for j = 1:N_enkf
896         q_err_meas = quat_mult(qer(i+1,:), quat_inv(q_ensemble_enkf(j,:)));
897         if q_err_meas(4) < 0, q_err_meas = -q_err_meas; end
898         Y_pred(:,j) = 2 * q_err_meas(1:3)';
899     end
900     y_bar = mean(Y_pred, 2);
901     Y_a = Y_pred - y_bar;
902 
903     P_xy = (X_a * Y_a') / (N_enkf - 1);
904     P_yy = (Y_a * Y_a') / (N_enkf - 1) + Rk;
905     P_yy = ensure_spd(P_yy, 1e-12*eye(3));
906     K = P_xy / P_yy;
907 
908     obs_pert = y_true + Rk_chol*randn(m, N_enkf);
909     for j = 1:N_enkf
910         innov = obs_pert(:,j) - Y_pred(:,j);
911         delta_state = K * innov;
912         delta_theta = delta_state(1:3);
913         delta_bias = delta_state(4:6);
914         dq_upd = delta_to_quat(delta_theta');
915         q_ensemble_enkf(j,:) = normalize_quat(quat_mult(dq_upd,
            ↪ q_ensemble_enkf(j,:)));
916         b_ensemble_enkf(j,:) = b_ensemble_enkf(j,:) + delta_bias';
917     end
```

```
    q_enkf(i+1,:) = mean_quat(q_ensemble_enkf);
    b_enkf(i+1,:) = mean(b_ensemble_enkf, 1);

    DX = zeros(6, N_enkf);
    for j = 1:N_enkf
        q_err = quat_mult(q_ensemble_enkf(j,:), quat_inv(q_enkf(i+1,:)));
        if q_err(4) < 0, q_err = -q_err; end
        DX(1:3,j) = 2 * q_err(1:3)';
        DX(4:6,j) = (b_ensemble_enkf(j,:) - b_enkf(i+1,:))';
    end
    P_samp = (DX*DX')/(N_enkf-1);
    sig3enkf(i+1,:) = sqrt(diag(P_samp(1:3,1:3)))' * 3 * 180/pi;

    q_err_enkf = quat_mult(q(i+1,:), quat_inv(q_enkf(i+1,:)));
    if q_err_enkf(4) < 0, q_err_enkf = -q_err_enkf; end
    enkf_error(i+1,:) = 2 * q_err_enkf(1:3) * 180/pi;
end

    time_enkf = toc;
    fprintf('EnKF completo em %.2f segundos\n', time_enkf);
else
    time_enkf = 0;
end

d_2000 = 367*yr - floor(7*(yr+floor((mth+9)/12))/4) + floor(275*mth/9) + ...
    (hr + minute/60 + (sec + t)/3600)/24 + day - 730531.5;
theta = (280.46061837 + 360.98564736628*d_2000)*pi/180;
r_ecef = [cos(theta).*pos(:,1) + sin(theta).*pos(:,2), ...
          -sin(theta).*pos(:,1) + cos(theta).*pos(:,2), ...
          pos(:,3)];
[lat,long,height] = ecef2llh(r_ecef);

fprintf('\nSalvando dados para análise...\n');
data_dir = fullfile(fileparts(fileparts(mfilename('fullpath'))), 'data',
    ↪ 'results');
if exist(data_dir, 'dir') ~= 7
    mkdir(data_dir);
end

save_vars = {'t', 'tf', 'pos', 'vel', 'alt_vec', 'lat', 'long', ...
             'earth_radius_km', 'target_altitude_km', 'dt', 'q', 'outlier_mask',
```

```
                ↪ 'sun_avail', ...
958             'deploy_altitude_km', 'orbital_rate', 'bias', 'RUN_QUEST',
                ↪ 'RUN_ALPHA', 'RUN_EKF', ...
959             'RUN_RAKF', 'RUN_ENKF', 'RUN_ENSRF'};
960 if RUN_QUEST
961     save_vars = [save_vars, {'qer', 'sig3qer', 'errer'}];
962 end
963 if RUN_ALPHA
964     save_vars = [save_vars, {'q_alp', 'sig3alp', 'errea', 'time_alpha'}];
965 end
966 if RUN_EKF
967     save_vars = [save_vars, {'q_ekf', 'sig3ekf', 'ekf_error', 'b_ekf',
        ↪ 'time_ekf'}];
968 end
969 if RUN_RAKF
970     save_vars = [save_vars, {'q_rakf', 'sig3rakf', 'rakf_error', 'lambda_rakf',
        ↪ 'mahal_dist_hist', ...
971                          'geom_quality_hist', 'outlier_flags', 'Rk_rakf_hist',
                             ↪ 'b_rakf', 'time_rakf'}];
972 end
973 if RUN_ENKF
974     save_vars = [save_vars, {'q_enkf', 'sig3enkf', 'enkf_error', 'b_enkf',
        ↪ 'time_enkf'}];
975 end
976 if RUN_ENSRF
977     save_vars = [save_vars, {'q_ensrf', 'sig3ensrf', 'ensrf_error', 'b_ensrf',
        ↪ 'time_ensrf'}];
978 end
979
980 save(fullfile(data_dir,'plot_data.mat'), save_vars{:});
981 fprintf('Dados salvos em %s\n', fullfile(data_dir,'plot_data.mat'));
982
983 fprintf('\n=======================================\n');
984 fprintf('RESUMO DE DESEMPENHO DOS FILTROS\n');
985 fprintf('=======================================\n');
986 fprintf('Erro RMS Total (todos os eixos combinados):\n');
987
988 filters = {};
989 errors = {};
990 if RUN_QUEST
991     filters{end+1} = 'QUEST';
```

```
992     errors{end+1} = errer;
993 end
994 if RUN_ALPHA
995     filters{end+1} = 'Alpha';
996     errors{end+1} = errea;
997 end
998 if RUN_EKF
999     filters{end+1} = 'EKF';
1000    errors{end+1} = ekf_error;
1001 end
1002 if RUN_RAKF
1003    filters{end+1} = 'RAKF';
1004    errors{end+1} = rakf_error;
1005 end
1006 if RUN_ENKF
1007    filters{end+1} = 'EnKF';
1008    errors{end+1} = enkf_error;
1009 end
1010 if RUN_ENSRF
1011    filters{end+1} = 'EnSRF';
1012    errors{end+1} = ensrf_error;
1013 end
1014
1015 for i = 1:numel(filters)
1016    err = errors{i};
1017    overall_rms = sqrt(mean(sum(err.^2,2),'omitnan'));
1018    fprintf(' %s: %.4f graus\n', filters{i}, overall_rms);
1019 end
1020 fprintf('========================================\n');
1021 fprintf('Simulação completa!\n');
1022
1023 fprintf('\n========================================\n');
1024 fprintf('GERANDO GRFICOS DE ANLISE\n');
1025 fprintf('NOTA: Alguns gráficos requerem filtros específicos habilitados.\n');
1026 fprintf(' Para gráficos completos, habilite todos os filtros.\n');
1027 fprintf('========================================\n');
1028
1029 fig_dir = fullfile(fileparts(fileparts(mfilename('fullpath'))), 'data', 'figures');
1030 if exist(fig_dir, 'dir') ~= 7
1031    mkdir(fig_dir);
1032 end
```

```
1033
1034 t_hours = t / 3600;
1035 t_days = t / 86400;
1036
1037 colors = struct();
1038 colors.quest = [0.8, 0.4, 0.0];
1039 colors.alpha = [0.0, 0.5, 0.8];
1040 colors.ekf = [0.2, 0.7, 0.2];
1041 colors.rakf = [0.9, 0.2, 0.4];
1042 colors.enkf = [0.6, 0.0, 0.6];
1043 colors.ensrf = [0.0, 0.7, 0.7];
1044
1045 fprintf('Gerando Figura 1: Altitude vs Tempo...\n');
1046 fig1 = figure('Position', [100, 100, 800, 500]);
1047
1048 plot(t_days, alt_vec, 'k-', 'LineWidth', 1.5);
1049 hold on;
1050 yline(deploy_altitude_km, 'r--', 'LineWidth', 1.5);
1051 deploy_text_x = t_days(end) * 0.7;
1052 deploy_text_y = deploy_altitude_km + 5;
1053 text(deploy_text_x, deploy_text_y, 'Shield Deploy (250 km)', 'FontSize', 10);
1054 xlabel('Time (days)', 'FontSize', 12);
1055 ylabel('Altitude (km)', 'FontSize', 12);
1056 title('Orbital Decay Profile (400 km to 120 km)', 'FontSize', 14);
1057 grid on;
1058 altitude_limits = [target_altitude_km - 10, initial_altitude_km + 10];
1059 ylim(altitude_limits);
1060 set(gca, 'FontSize', 12);
1061
1062 saveas(fig1, fullfile(fig_dir, 'fig1_altitude.png'));
1063 fprintf(' Salvo: fig1_altitude.png\n');
1064
1065 fprintf('Gerando Figura 2: Erro de Atitude do QUEST...\n');
1066 fig2 = figure('Position', [100, 100, 900, 700]);
1067
1068 subplot(3,1,1)
1069 plot(t_hours, -sig3qer(:,1), 'r', t_hours, errer(:,1), 'b', t_hours, sig3qer(:,1),
        ↪ 'r', 'LineWidth', 1.0);
1070 ylabel('Roll (deg)');
1071 title('QUEST - Attitude Estimation Error');
1072 set(gca, 'FontSize', 12);
```

```
grid on;
xlim([0 t_hours(end)]);

subplot(3,1,2)
plot(t_hours, errer(:,2), 'b', t_hours, -sig3qer(:,2), 'r', t_hours, sig3qer(:,2),
    ↪ 'r', 'LineWidth', 1.0);
ylabel('Pitch (deg)');
grid on;
set(gca, 'FontSize', 12);
xlim([0 t_hours(end)]);

subplot(3,1,3)
plot(t_hours, -sig3qer(:,3), 'r', 'LineWidth', 1.0);
hold on;
plot(t_hours, errer(:,3), 'b', 'LineWidth', 1.0);
plot(t_hours, sig3qer(:,3), 'r', 'LineWidth', 1.0);
xlabel('Time (hours)');
ylabel('Yaw (deg)');
set(gca, 'FontSize', 12);
grid on;
xlim([0 t_hours(end)]);

saveas(fig2, fullfile(fig_dir, 'fig2_quest_error.png'));
fprintf(' Salvo: fig2_quest_error.png\n');

fprintf('Gerando Figura 3: Erro de Atitude do Filtro Alfa...\n');
fig3 = figure('Position', [100, 100, 900, 700]);

subplot(3,1,1)
plot(t_hours, errea(:,1), 'b', 'LineWidth', 1.0);
set(gca, 'FontSize', 12);
ylabel('Roll (deg)');
title('Alpha Filter - Attitude Estimation Error');
grid on;
xlim([0 t_hours(end)]);

subplot(3,1,2)
plot(t_hours, errea(:,2), 'b', 'LineWidth', 1.0);
set(gca, 'FontSize', 12);
ylabel('Pitch (deg)');
grid on;
```

```
xlim([0 t_hours(end)]);

subplot(3,1,3)
plot(t_hours, errea(:,3), 'b', 'LineWidth', 1.0);
set(gca, 'FontSize', 12);
ylabel('Yaw (deg)');
xlabel('Time (hours)');
grid on;
xlim([0 t_hours(end)]);

saveas(fig3, fullfile(fig_dir, 'fig3_alpha_error.png'));
fprintf(' Salvo: fig3_alpha_error.png\n');

fprintf('Gerando Figura 4: Erro de Atitude do EKF...\n');
fig4 = figure('Position', [100, 100, 900, 700]);

subplot(3,1,1)
plot(t_hours, -sig3ekf(:,1), 'r', t_hours, ekf_error(:,1), 'b', t_hours,
    ↪ sig3ekf(:,1), 'r', 'LineWidth', 1.0);
set(gca, 'FontSize', 12);
ylabel('Roll (deg)');
title('EKF - Attitude Estimation Error');
grid on;
xlim([0 t_hours(end)]);

subplot(3,1,2)
plot(t_hours, -sig3ekf(:,2), 'r', t_hours, ekf_error(:,2), 'b', t_hours,
    ↪ sig3ekf(:,2), 'r', 'LineWidth', 1.0);
set(gca, 'FontSize', 12);
ylabel('Pitch (deg)');
grid on;
xlim([0 t_hours(end)]);

subplot(3,1,3)
plot(t_hours, -sig3ekf(:,3), 'r', t_hours, ekf_error(:,3), 'b', t_hours,
    ↪ sig3ekf(:,3), 'r', 'LineWidth', 1.0);
set(gca, 'FontSize', 12);
ylabel('Yaw (deg)');
xlabel('Time (hours)');
grid on;
xlim([0 t_hours(end)]);
```

```
1151
1152 saveas(fig4, fullfile(fig_dir, 'fig4_ekf_error.png'));
1153 fprintf(' Salvo: fig4_ekf_error.png\n');
1154
1155 fprintf('Gerando Figura 5: Erro de Atitude do RAKF...\n');
1156 fig5 = figure('Position', [100, 100, 900, 700]);
1157
1158 subplot(3,1,1)
1159 plot(t_hours, -sig3rakf(:,1), 'r', t_hours, rakf_error(:,1), 'b', t_hours,
        ↪ sig3rakf(:,1), 'r', 'LineWidth', 1.0);
1160 set(gca, 'FontSize', 12);
1161 ylabel('Roll (deg)');
1162 title('RAKF - Attitude Estimation Error');
1163 grid on;
1164 xlim([0 t_hours(end)]);
1165
1166 subplot(3,1,2)
1167 plot(t_hours, -sig3rakf(:,2), 'r', t_hours, rakf_error(:,2), 'b', t_hours,
        ↪ sig3rakf(:,2), 'r', 'LineWidth', 1.0);
1168 set(gca, 'FontSize', 12);
1169 ylabel('Pitch (deg)');
1170 grid on;
1171 xlim([0 t_hours(end)]);
1172
1173 subplot(3,1,3)
1174 plot(t_hours, -sig3rakf(:,3), 'r', t_hours, rakf_error(:,3), 'b', t_hours,
        ↪ sig3rakf(:,3), 'r', 'LineWidth', 1.0);
1175 set(gca, 'FontSize', 12);
1176 ylabel('Yaw (deg)');
1177 xlabel('Time (hours)');
1178 grid on;
1179 xlim([0 t_hours(end)]);
1180
1181 saveas(fig5, fullfile(fig_dir, 'fig5_rakf_error.png'));
1182 fprintf(' Salvo: fig5_rakf_error.png\n');
1183
1184 fprintf('Gerando Figura 6: Erro de Atitude do EnKF...\n');
1185 fig6 = figure('Position', [100, 100, 900, 700]);
1186
1187 subplot(3,1,1)
1188 plot(t_hours, -sig3enkf(:,1), 'r', t_hours, enkf_error(:,1), 'b', t_hours,
```

```
    ↪ sig3enkf(:,1), 'r', 'LineWidth', 1.0);
1189 set(gca, 'FontSize', 12);
1190 ylabel('Roll (deg)');
1191 title('EnKF - Attitude Estimation Error');
1192 grid on;
1193 xlim([0 t_hours(end)]);
1194
1195 subplot(3,1,2)
1196 plot(t_hours, -sig3enkf(:,2), 'r', t_hours, enkf_error(:,2), 'b', t_hours,
    ↪ sig3enkf(:,2), 'r', 'LineWidth', 1.0);
1197 set(gca, 'FontSize', 12);
1198 ylabel('Pitch (deg)');
1199 grid on;
1200 xlim([0 t_hours(end)]);
1201
1202 subplot(3,1,3)
1203 plot(t_hours, -sig3enkf(:,3), 'r', t_hours, enkf_error(:,3), 'b', t_hours,
    ↪ sig3enkf(:,3), 'r', 'LineWidth', 1.0);
1204 set(gca, 'FontSize', 12);
1205 ylabel('Yaw (deg)');
1206 xlabel('Time (hours)');
1207 grid on;
1208 xlim([0 t_hours(end)]);
1209
1210 saveas(fig6, fullfile(fig_dir, 'fig6_enkf_error.png'));
1211 fprintf(' Salvo: fig6_enkf_error.png\n');
1212
1213 fprintf('Gerando Figura 7: Erro de Atitude do EnSRF...\n');
1214 fig7 = figure('Position', [100, 100, 900, 700]);
1215
1216 subplot(3,1,1)
1217 plot(t_hours, -sig3ensrf(:,1), 'r', t_hours, ensrf_error(:,1), 'b', t_hours,
    ↪ sig3ensrf(:,1), 'r', 'LineWidth', 1.0);
1218 set(gca, 'FontSize', 12);
1219 ylabel('Roll (deg)');
1220 title('EnSRF - Attitude Estimation Error');
1221 grid on;
1222 xlim([0 t_hours(end)]);
1223
1224 subplot(3,1,2)
1225 plot(t_hours, -sig3ensrf(:,2), 'r', t_hours, ensrf_error(:,2), 'b', t_hours,
```

```
    ↪ sig3ensrf(:,2), 'r', 'LineWidth', 1.0);
set(gca, 'FontSize', 12);
ylabel('Pitch (deg)');
grid on;
xlim([0 t_hours(end)]);

subplot(3,1,3)
plot(t_hours, -sig3ensrf(:,3), 'r', t_hours, ensrf_error(:,3), 'b', t_hours,
    ↪ sig3ensrf(:,3), 'r', 'LineWidth', 1.0);
set(gca, 'FontSize', 12);
ylabel('Yaw (deg)');
xlabel('Time (hours)');
grid on;
xlim([0 t_hours(end)]);

saveas(fig7, fullfile(fig_dir, 'fig7_ensrf_error.png'));
fprintf(' Salvo: fig7_ensrf_error.png\n');

fprintf('\nSalvando estatísticas detalhadas em CSV...\n');

rms_quest = sqrt(sum(errer.^2, 2));
rms_alpha = sqrt(sum(errea.^2, 2));
rms_ekf = sqrt(sum(ekf_error.^2, 2));
rms_rakf = sqrt(sum(rakf_error.^2, 2));
rms_enkf = sqrt(sum(enkf_error.^2, 2));
rms_ensrf = sqrt(sum(ensrf_error.^2, 2));

all_filters = {'QUEST', 'Alpha', 'EKF', 'RAKF', 'EnKF', 'EnSRF'};
all_errors = {rms_quest, rms_alpha, rms_ekf, rms_rakf, rms_enkf, rms_ensrf};

num_filters = length(all_filters);
stats_mean = zeros(num_filters, 1);
stats_std = zeros(num_filters, 1);
stats_max = zeros(num_filters, 1);
stats_p95 = zeros(num_filters, 1);
stats_p50 = zeros(num_filters, 1);

for i = 1:num_filters
    current_err = all_errors{i};
    stats_mean(i) = mean(current_err, 'omitnan');
    stats_std(i) = std(current_err, 'omitnan');
```

```
1265     stats_max(i) = max(current_err);
1266     stats_p95(i) = prctile(current_err, 95);
1267     stats_p50(i) = prctile(current_err, 50);
1268 end
1269 
1270 T = table(all_filters', stats_mean, stats_std, stats_p50, stats_p95, stats_max, ...
1271           'VariableNames', {'Filter', 'Mean_deg', 'Std_deg', 'Median_deg',
                  ↪ 'P95_deg', 'Max_deg'});
1272 
1273 csv_file = fullfile(data_dir, 'performance_statistics.csv');
1274 writetable(T, csv_file);
1275 fprintf(' Salvo: performance_statistics.csv\n');
1276 T_timing = table({'Alpha'; 'EKF'; 'RAKF'; 'EnKF'; 'EnSRF'}, ...
1277                  [time_alpha; time_ekf; time_rakf; time_enkf; time_ensrf], ...
1278                  'VariableNames', {'Filter', 'Time_seconds'});
1279 csv_timing = fullfile(data_dir, 'timing_statistics.csv');
1280 writetable(T_timing, csv_timing);
1281 fprintf(' Salvo: timing_statistics.csv\n');
1282 
1283 lambda_valid = lambda_rakf(isfinite(lambda_rakf));
1284 mahal_valid = mahal_dist_hist(isfinite(mahal_dist_hist));
1285 geom_valid = geom_quality_hist(isfinite(geom_quality_hist));
1286 num_outliers_detected = sum(outlier_flags);
1287 outlier_rate = 100 * num_outliers_detected / length_t;
1288 
1289 diag_table = table( ...
1290     mean(lambda_valid), median(lambda_valid), max(lambda_valid),
          ↪ min(lambda_valid), ...
1291     mean(mahal_valid), median(mahal_valid), max(mahal_valid), ...
1292     mean(geom_valid), median(geom_valid), min(geom_valid), ...
1293     num_outliers_detected, outlier_rate, ...
1294     'VariableNames', {'Lambda_mean','Lambda_median','Lambda_max','Lambda_min', ...
1295                       'Mahal_mean','Mahal_median','Mahal_max', ...
1296                       'GeomQuality_mean','GeomQuality_median','GeomQuality_min', ...
1297                       'Outliers_detected','Outlier_rate_percent'});
1298 writetable(diag_table, fullfile(data_dir, 'rakf_diagnostics.csv'));
1299 fprintf(' Salvo: rakf_diagnostics.csv\n');
1300 
1301 summary_file = fullfile(data_dir, 'analysis_summary.txt');
1302 [fid,msg] = fopen(summary_file, 'w');
1303 if fid == -1
```

```
1304     warning('N\u00e3o foi poss\u00edvel abrir arquivo de resumo: %s', msg);
1305 else
1306     fprintf(fid, "Resumo de Desempenho dos Filtros de Atitude\n");
1307     fprintf(fid, "Dura\u00e7\u00e3o da simula\u00e7\u00e3o: %.2f dias (%.2f
         ↪ horas)\n", tf/86400, tf/3600);
1308     fprintf(fid, "Passo de tempo: %.1f s, amostras: %d\n\n", dt, length_t);
1309     fprintf(fid, "Filtro\tRMS Total (graus)\tRMS Mediano (graus)\tPercentil 95
         ↪ (graus)\tTempo Execu\u00e7\u00e3o (s)\n");
1310     runtime_lookup = containers.Map( ...
1311         {'QUEST','Alpha','EKF','RAKF','EnKF','EnSRF'}, ...
1312         [NaN, time_alpha, time_ekf, time_rakf, time_enkf, time_ensrf]);
1313     for idx = 1:numel(filters)
1314         err = errors{idx};
1315         err_mag = sqrt(sum(err.^2,2));
1316         overall_rms = sqrt(mean(err_mag.^2,'omitnan'));
1317         median_err = median(err_mag,'omitnan');
1318         p95_err = prctile(err_mag,95);
1319         t_run = runtime_lookup(filters{idx});
1320         fprintf(fid, "%s\t%.4f\t%.4f\t%.4f\t%.2f\n", filters{idx}, overall_rms,
             ↪ median_err, p95_err, t_run);
1321     end
1322     fprintf(fid, "\nEstat\u00edsticas adaptativas RAKF:\n");
1323     fprintf(fid, " Fator de atenua\u00e7\u00e3o: m\u00e9dia  = %.2f, m\u00e1x  =
         ↪ %.2f\n", mean(lambda_valid), max(lambda_valid));
1324     fprintf(fid, " Outliers detectados: %d (%.1f%%)\n", num_outliers_detected,
         ↪ outlier_rate);
1325     fprintf(fid, " Qualidade geom\u00e9trica: m\u00e9dia = %.3f, m\u00edn =
         ↪ %.3f\n", mean(geom_valid), min(geom_valid));
1326     fclose(fid);
1327 end
1328 fprintf(' Salvo: analysis_summary.txt\n');
1329
1330 fprintf('\n========================================\n');
1331 fprintf('TODOS OS GRFICOS E DADOS GERADOS COM SUCESSO\n');
1332 fprintf('========================================\n');
1333 fprintf('Figuras salvas em: %s\n', fig_dir);
1334 fprintf('Dados salvos em: %s\n', data_dir);
1335 fprintf('\n');
1336
1337 function [time_hist,pos_hist,vel_hist] = propagateDecayOrbit(dt, max_prop_time,
         ↪ initial_state, start_datetime, target_altitude_km, mu, earth_radius_km,
```

```
    ↪ sat_params, atm_params)
1338    if dt <= 0
1339        error('O passo de tempo dt deve ser positivo.');
1340    end
1341
1342    max_steps = floor(max_prop_time/dt) + 1;
1343    state_hist = zeros(max_steps,6);
1344    time_hist = zeros(max_steps,1);
1345
1346    state_hist(1,:) = initial_state;
1347    time_hist(1) = 0;
1348
1349    omega_earth = 7.2921159e-5;
1350
1351    idx = 1;
1352    while idx < max_steps
1353        current_state = state_hist(idx,:);
1354        current_time = time_hist(idx);
1355        current_state_col = current_state(:);
1356        r = current_state_col(1:3);
1357        alt_km = norm(r) - earth_radius_km;
1358
1359        if alt_km <= target_altitude_km
1360            break;
1361        end
1362
1363        k1 = orbitalDynamics(current_state_col, mu, earth_radius_km, sat_params,
                ↪ atm_params, current_time, start_datetime, omega_earth);
1364        k2 = orbitalDynamics(current_state_col + 0.5*dt*k1, mu, earth_radius_km,
                ↪ sat_params, atm_params, current_time + 0.5*dt, start_datetime,
                ↪ omega_earth);
1365        k3 = orbitalDynamics(current_state_col + 0.5*dt*k2, mu, earth_radius_km,
                ↪ sat_params, atm_params, current_time + 0.5*dt, start_datetime,
                ↪ omega_earth);
1366        k4 = orbitalDynamics(current_state_col + dt*k3, mu, earth_radius_km,
                ↪ sat_params, atm_params, current_time + dt, start_datetime,
                ↪ omega_earth);
1367
1368        next_state_col = current_state_col + dt/6*(k1 + 2*k2 + 2*k3 + k4);
1369        if any(~isfinite(next_state_col))
1370            warning('Estado n\u00e3o finito encontrado durante propaga\u00e7\u00e3o
```

```
                ↪ orbital no passo %d.', idx);
1371            break;
1372        end
1373        next_state = next_state_col.';
1374
1375        idx = idx + 1;
1376        state_hist(idx,:) = next_state;
1377        time_hist(idx) = current_time + dt;
1378
1379        if mod(idx, 360) == 0
1380            fprintf('Tempo: %.2f horas (%.1f dias), Altitude: %.2f km\n',
                ↪ current_time/3600, current_time/86400, alt_km);
1381        end
1382    end
1383
1384    state_hist = state_hist(1:idx,:);
1385    time_hist = time_hist(1:idx);
1386
1387    pos_hist = state_hist(:,1:3);
1388    vel_hist = state_hist(:,4:6);
1389 end
1390
1391 function state_dot = orbitalDynamics(state_col, mu, earth_radius_km, sat_params,
    ↪ atm_params, current_time, start_datetime, omega_earth)
1392    r = state_col(1:3);
1393    v = state_col(4:6);
1394
1395    r_norm = norm(r);
1396    if r_norm == 0
1397        error('Vetor de posi\u00e7\u00e3o zero encontrado na din\u00e2mica
            ↪ orbital.');
1398    end
1399
1400    current_datetime = start_datetime + seconds(current_time);
1401    lat_deg = rad2deg(asin(max(min(r(3) / r_norm,1),-1)));
1402    lon_deg = rad2deg(atan2(r(2), r(1)));
1403    alt_km = max(r_norm - earth_radius_km, 0);
1404    rho = computeAtmosDensity(alt_km, lat_deg, lon_deg, current_datetime,
        ↪ atm_params);
1405
1406    if alt_km > sat_params.deploy_altitude_km
```

```
        Cd_curr = sat_params.shield_closed.Cd;
        area_curr = sat_params.shield_closed.area;
    else
        Cd_curr = sat_params.shield_open.Cd;
        area_curr = sat_params.shield_open.area;
    end

    r_m = r * 1e3;
    v_m = v * 1e3;
    omega_vec = [0; 0; omega_earth];
    v_atm = cross(omega_vec, r_m);
    v_rel = v_m - v_atm;
    v_rel_mag = norm(v_rel);

    grav_acc_km = -mu * r / r_norm^3;
    if v_rel_mag > 0
        drag_acc_m = -0.5 * rho * Cd_curr * area_curr / sat_params.mass *
            ↪ v_rel_mag * v_rel;
        drag_acc_km = (drag_acc_m(:)) / 1000;
    else
        drag_acc_km = zeros(3,1);
    end

    state_dot = reshape([v(:); grav_acc_km + drag_acc_km], [], 1);
end

function rho = computeAtmosDensity(alt_km, lat_deg, lon_deg, current_datetime,
    ↪ atm_params)
    year_val = year(current_datetime);
    doy_val = day(current_datetime,'dayofyear');
    sec_of_day = hour(current_datetime)*3600 + minute(current_datetime)*60 +
        ↪ second(current_datetime);

    alt_m = min(max(alt_km*1000, 0), 1e6);
    [~, density_matrix] = atmosnrlmsise00(alt_m, lat_deg, lon_deg, year_val,
        ↪ doy_val, sec_of_day, ...
        atm_params.f107a, atm_params.f107, atm_params.ap);

    if ~isnumeric(density_matrix) || size(density_matrix,2) < 6
        error('Sa\u00edda de densidade inesperada de atmosnrlmsise00.');
    end
```

```
    rho = density_matrix(end,6);
end

function R_cov = measurement_covariance_from_quest(P_meas, idx, R_default)
    R_cov = R_default;
    if idx >= 1 && idx <= size(P_meas,3)
        candidate = P_meas(:,:,idx);
        if all(isfinite(candidate(:)))
            R_cov = ensure_spd((candidate + candidate')/2, 1e-12*eye(3));
            return;
        end
    end
end

function A_spd = ensure_spd(A, floor_mat)
    n = size(A,1);
    if nargin < 2
        floor_mat = zeros(n);
    elseif isscalar(floor_mat)
        floor_mat = floor_mat * eye(n);
    else
        floor_mat = (floor_mat + floor_mat')/2;
    end

    A_sym = (A + A')/2 + floor_mat;
    diag_abs = abs(diag(A_sym));
    diag_abs(~isfinite(diag_abs)) = 0;
    base_scale = max(1, max(diag_abs));
    jitter = 1e-12 * base_scale;
    p = 1;
    iter = 0;
    while p ~= 0 && iter < 7
        [~,p] = chol(A_sym,'lower');
        if p == 0
            break;
        end
        A_sym = A_sym + (10^iter) * jitter * eye(n);
        iter = iter + 1;
    end

    if p ~= 0
```

```
1485         A_sym = A_sym + 1e-6*eye(n);
1486     end
1487 
1488     A_spd = (A_sym + A_sym')/2;
1489 end
1490 
1491 function invA = regularized_inverse(M, lambda)
1492     M_sym = (M + M')/2;
1493     n = size(M_sym,1);
1494     if nargin < 2
1495         lambda = 1e-9;
1496     end
1497     lambda = max(lambda, eps);
1498     M_reg = M_sym + lambda*eye(n);
1499     inv_raw = -pinv(M_reg);
1500     invA = ensure_spd(inv_raw);
1501 end
```

Código A.1 – Código de Estimação de Atitude