FÁBIO DORO ZANONI

DESENVOLVIMENTO DE FILTRO DE KALMAN ON-LINE COM REQUISITOS TEMPO-REAL PARA A NAVEGAÇÃO DE VANTS

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do titulo de bacharel em Engenharia Mecatrônica.

São Paulo

2007

FÁBIO DORO ZANONI

DESENVOLVIMENTO DE FILTRO DE KALMAN ON-LINE COM REQUISITOS TEMPO-REAL PARA A NAVEGAÇÃO DE VANTS

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do titulo de bacharel em Engenharia Mecatrônica.

Área de Concentração:
Engenharia Mecatrônica

Orientador:
Prof. Dr. Ettore Apolônio de Barros

São Paulo

2007

FICHA CATALOGRÁFICA

Zanoni, Fabio Doro
Desenvolvimento de Filtro de Kalman on-line com requisitos de "tempo real" para a navegação de vants / F.D. Zanoni. -- São Paulo, 2007.
80 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1.Filtros de Kalman (On-line; Desenvolvimento) I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos II.t.

DEDICATÓRIA

Dedico este trabalho a quem possa interessar e aos Amigos

AGRADECIMENTOS

Agradeço primeiramente aos meus pais (Devanir José Zanoni e Sonia Maria Doro Zanoni) por estar junto comigo em todos os momentos da minha vida me dando todo o apoio que eu precisava.

Ao meu irmão que me ajudou e esteve ao meu lado nos momentos que eu precisava.

Aos Amigos da Poli que esteve junto comigo nestes anos de dificuldades e de diversão.

Ao meu orientado (prof. Dr. Ettore Apolônio de Barros) que vem me ajudado desde minha iniciação científica e me dando oportunidade no ramo de navegação de veículos.

Ao aluno de Mestrado Giovani Aminati que me ajudou durante este trabalho até mesmo nos feriados e finais de semanas passados na Poli e pela paciência que teve em tentar me ensinar alguns métodos de estruturação de software.

Resumo

Este trabalho tem por objetivo o estudo, a implementação e os testes de bancada e de campo de técnica de calculo computacional com requisitos de tempo real das equações de RICCATI utilizadas no filtro de Kalman. Outro propósito deste trabalho é o estudo e a implementação da determinação de parâmetros modelos do filtro, principalmente as covariâncias dos sensores e do processo. Para isto, há a necessidade de se estudar o filtro de Kalman visando sua utilização no Veículo Aéreo Autônomo (UAV) e sua implementação no MATLAB, levando-se em conta alguns cuidados que se deve tomar ao utilizá-lo. Este filtro é freqüentemente usado para combinar dados obtidos de diferentes sensores em uma estimativa estatisticamente ótima e devido a isto ele utilizado no UAV para definir a sua posição, atitude e taxa de rotação. Mas para que se possa utilizá-lo com este objetivo há a necessidade de aplicá-lo em tempo real, para sua utilização durante o vôo do veículo. Conseqüentemente surgiu a necessidade de se estudar as principais características do Sistema de Tempo Real. A princípio utilizará o código desenvolvido no MATLAB para criar o código em tempo real e para isto utilizaremos Rhapsody, que a partir do modelo em MATLAB, torna possível gerar um código em C++ que será compilado pelo QNX Momentics, que por sua vez, embarcara o código no "target" (PC104). Desta maneira, com a implantação do Filtro de Kalman em Tempo Real, será possível determinar a posição, atitude e taxa de rotação do UAV, para que se possa controlá-lo.

Palavras Chave. Filtro de Kalman, Filtro de Kalman Estendido, Processo Estocástico.

ABSTRACT

This work has for objective the study, the implementation and the bench tests and of technique field of computational methods with requirements of real time of the equations of RICCATI used in the filter of Kalman. Another purpose of this work is the study and the implementation of the determination of parameters models of the filter, mainly the sensor's covariance and of the process. For this, there is the need to study the filter of Kalman seeking it use in the Autonomous Aerial Vehicle (UAV) and it implementation in MATLAB, being taken into account some cares that she should take when using it. This filter is frequently used to combine obtained data of different sensor in an estimate great statistically and due to this he used in UAV to define its position, attitude and rotation tax. But so that she can use it with this objective there is the need to apply it in real time, for it use during the flight of the vehicle. Consequently the need appeared of studying the main characteristics of the System of Real Time. At first it will use the code developed in MATLAB to create the code in real time and for this we will use Rhapsody, that starting from the model in MATLAB, it turns possible to generate a code in C++ that will be compiled by QNX Momentics, that for its time, it had embarked the code in the "target (PC104)." Of this it sorts things out, with the implantation of the Filter of Kalman in Real Time, it will be possible to determine the position, attitude and tax of rotation of UAV, so that she can control it.

Keywords. Kalman Filter, Extended Kalman Filter, Stochastic Process

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviatura

EKF	Filtro de Kalman Estendido (Extended Kalman Filter)
GPS	Sistema Global de Posicionamento (Global Positioning System)
IMU	Unidade Inercial de Medição (Inertial measurement Unit)
STR	Sistema de Tempo Real
UAV	Veículo Aéreo Autônomo
NED	Sistema de coordenadas North East Down
ECEF	Sistema de coordenadas Earth-Centered, Earth-Fixed

Lista de Símbolos

A	Matriz de transição de estados
B	Matriz de entradas de controle
H	Matriz de leitura dos sensores
G	Matriz de ruídos de medição
x_k	Vetor de estados
z_k	Vetor de medições do processo no instante t_k
w_k , v_k	Ruídos brancos de media nula do processo
u_k	Vetor entrada de controle
Q	Covariância do ruído de dinâmica
R	Covariância do ruído de medição
P	Matriz covariância dos estados
C	Matriz de ruídos da dinâmica
v	Velocidade
v_0	Velocidade inicial
a , a_x , a_y , a_z	Aceleração
p	Posição
p_0	Posição inicial
T	Intervalo de tempo no espaço discreto
θ	Posição Angular
θ_0	Posição Angular inicial
b^a	Erro de bias nos sistema de coordenadas da plataforma
a_e^n	Aceleração a ser utilizada para determinar a posição da plataforma no sistema NED
C_p^n	Matriz mudança de base das coordenada da plataforma para a coordenada no sistema NED
a_n	Aceleração medida pelo acelerômetro no sistema NED
b^n	Erro de bias do acelerômetro
g	Aceleração da gravidade
ω_{ie}^n	Vetor rotação da Terra projetado no sistema NED

ω_{en}^{n}	Vetor de velocidade angular do sistema NED em relação ao sistema ECEF
[a, b, c, d]	Parâmetros do quaternions
λ	Latitude
ϕ	Longitude
v_e^p	Velocidade da plataforma
Ω	Velocidade angular de rotação da Terra
v_e^n	Velocidade linear da plataforma no sistema de coordenadas NED
ω_{en}^{n}	Velocidade angular da plataforma no sistema de coordenadas NED
$[\omega_x,\ \omega_y,\ \omega_z]$	Velocidades angulares da plataforma no sistema de coordenadas da plataforma
$[b_{gx},\ b_{gy},\ b_{gz}]$	Erros de bias do giroscópio no sistema de coordenadas da plataforma
b_{a_y}	Erros de bias dos acelerômetros no sistema de coordenadas da plataforma
δ_x	Erros randômicos dos Estados e Sensores, onde sub-índice “x“ indica de qual sensor ou estado a quem o erro pertence
W_i	Densidade espectral de potência do ruído associados ao vetor u

1. Introdução

Desde os tempos mais remotos, os homens se deslocam por longas distâncias, tendo como desafio a aquisição de conhecimentos sobre uma maneira de se orientar corretamente. Esta habilidade demandava uma forma de navegação. Nossos ancestrais viajavam longas distâncias à procura de comida e abrigo seguro utilizando uma navegação puramente visual.

Após um período de desenvolvimento, passaram a ser utilizado como meios de navegação agulhas magnéticas e observações dos astros e estrelas através de sextantes.

No século XVIII, Isaac Newton definiu as leis da mecânica e gravitação, cujos fundamentos eram essenciais para desenvolvimento dos princípios de navegação, mas só alguns séculos depois que foram desenvolvidos sensores giométricos para medir posição e velocidade angular e os acelerômetros para medir as acelerações lineares.

Uma ferramenta utilizada atualmente para auxiliar na navegação por sensores é o Filtro de Kalman.

O Filtro de Kalman foi desenvolvido em 1960 e foi utilizado para realizar estimativas de trajetórias. Este filtro é freqüentemente usado para combinar dados obtidos de diferentes sensores em uma estimativa estatisticamente ótima, devido a isto ele utilizado no Veiculo Aéreo Autônomo (UAV) para definir a sua posição no espaço.

Mas para que se possa utilizá-lo com este objetivo há a necessidade de se aplicá-lo em tempo real, para que durante o vôo do veiculo possa determinar sua posição, velocidade, atitude, taxa de rotação.

Conseqüentemente surgiu à necessidade de se estudar as principais características do Sistema de Tempo Real. Onde o que marca este sistema é a sua corretitude dependente não apenas da lógica da computação, mas também do cumprimento do tempo na produção dos resultados.

Desta maneira, com a implantação do Filtro de Kalman em tempo Real, será possível determinar a posição do UAV, para que se possa controlá-lo.

2. Objetivo

O objetivo principal deste trabalho de formatura é o estudo e a implementação do filtro de Kalman, de campos de técnica de calculo computacional com requisitos de tempo real e testes de bancada para validá-lo. Outro propósito deste trabalho é o estudo e a implementação de um modelo e seus modos de determinação de parâmetros dele, principalmente os parâmetros de covariâncias dos sensores e do processo.

3. Estudo de Sistema de Tempo Real (STR)

Como o objetivo do trabalho é implementar o EKF em tempo real, há a necessidade de conhecer-lo mais profundamente. Este estudo ainda está sendo desenvolvido. A princípio, utiliza-se o código desenvolvido no MATLAB para criar o código em tempo real para isto utilizaremos Rhapsody, que a partir do modelo em MATLAB é possível gerar um código em C++ que será compilado pelo QNX Momentics que por sua vez embarcara o código no target (PC104).

Um sistema de real-tempo é qualquer sistema de processamento de informação que tem que responder a estímulos de contribuição externamente gerados dentro um finito e especifico período. Onde processos são programas completos em execução.

Para atender a todas as tarefas solicitadas, dentro das suas respectivas restrições de tempo, os STR necessitam invariavelmente de alguns mecanismos de gerência de atividades, classificando-as (ordenando) segundo algum critério de prioridade de atendimento. A associação entre Sistemas de Tempo Real e Programação Concorrente é inevitável. Por Programação Concorrente entenda-se que é um tipo particular de programação que processa várias atividades de forma paralela.

Outro conceito importante a se saber é o de Threads, no qual, são fluxos de controle independentes num mesmo processo, ela herda recursos, diminuindo o tempo de chaveamento entre as tarefas.

Um fator que se deve levar em consideração para STR é a carga computacional utilizada onde ela é dada pelo somatório dos tempos de computação das tarefas a fila de pronto. A carga é estática ou limitada quando os valores de temporização podem ser conhecidos em tempo de projeto, deste modo o sistema é previsível caso contrário o estamos trabalhando com carga dinâmica ou ilimitada onde ocorre quando não se podem determinar os valores de temporização em tempo de execução.

Para a melhor utilização destes recursos pode-se utilizar de o escalonamento que é a Criação de uma escala de execução (schedule), uma lista

ordenada que indique como tarefas terão acesso aos recursos, onde os critérios de temporização são fundamentais. Os escalonadores devem implementar políticas que garantam que todas as tarefas estejam oportunas.

O escalonamento pode ser classificado em:

- Quanto à preempção:
  - Preemptivas – podem ser interrompidas por outras mais prioritárias
  - Não-preemptivas – não podem ser interrompidas por outras mais prioritárias
- Quanto ao instante de utilização dos valores dos critérios:
  - Estática – utiliza valores determinados em tempo de design
  - Dinâmica – utiliza valores determinados em tempo de execução
- Quanto ao instante da produção da escala
  - Off-line – produzida em tempo de design
  - On-line – produzida em tempo de execução

Há também três tipos de abordagem do Escalonamento são elas:

- Com garantia em tempo de projeto
  - Previsibilidade determinista e carga computacional estática
- Com garantia em tempo de execução
  - Carga computacional não previsível.
  - Escala e testes de escalonabilidade são determinados em tempo de execução.
- Melhor esforço
  - Carga computacional não previsível.
  - Aplicação de regras de despacho.
  - Não são realizados testes.
  - São aplicadas para STR brandos.

4. Descrição das Ferramentas Utilizadas no Projeto

As ferramentas utilizadas para desenvolver este projeto foram: MATLAB/Simulink, Rhapsody e Momentics QNX.

No Simulink foi desenvolvido um programa do filtro de Kalman em diagramas de blocos. Após isto este programa é passado para C++ utilizando a função Real-Time Workshop do Simulink. Após isto, realiza-se a interação do Simulink com o Rhapsody.

O Rhapsody pertence à empresa I-Logix. Ele é um software de desenvolvimento no qual utiliza a linguagem UML para padronizar o software a ser desenvolvido e possibilita um entendimento visual do projeto.

No Rhapsody será desenvolvida, em linguagem UML, a fusão do filtro de Kalman desenvolvido no Simulink com o software da leitura dos sensores, este software este sendo desenvolvido por Amianti.G. Depois disto, o arquivo será compilado no Rhapsody, onde este código é enviado para o Momentics QNX.

No Momentics QNX, o código é enviado para o Pc104, hardware que será embarcado no UAV. Além desta função, este software ele é utilizado para realizar a interação entre o hardware PC104 e o computador onde está sedo desenvolvido o código que será embarcado, desta maneira, o Momentics QNX monitora os processamentos ocorridos no PC 104, sendo possível conhecer o tempo gasto para realizar o processamento do programa embarcado. Portanto, ele permite verificar se os requisitos de tempo real para o filtro estão sendo atingidos. Observe abaixo o diagrama que representa as interações entre os softwares.

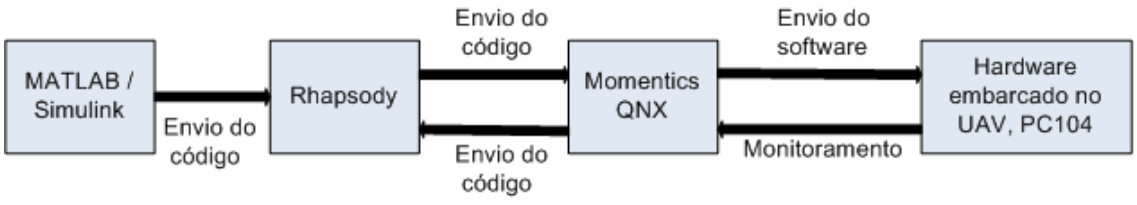

Figura 4.1 – Esquema das Integrações dos Softwares Utilizados

5. Formulação do Problema

Descreve-se a seguir o a formulação do problema da estimativa de trajetória através da navegação inercial. Inicialmente é descrito a teoria básica do Filtro de Kalman Discreto e a implementação do Filtro de Kalman Discreto no Simulink. Após isto, serão apresentados os resultados das simulações realizadas com o filtro construído no Simulink e um esquema descrevendo a sua implementação no Rhapsody.

5.1. Fusão Sensorial e sua Aplicação no Filtro de Kalman

Os Sistemas dinâmicos são afetados não somente pelos controles (determinísticos) impostos, mas também por perturbações que não se podem controlar e nem modelar deterministicamente devido a ruídos gerados pelo sistema por interferências externas que não são considerados no modelo.

Alem disto, podemos considerar que os sensores utilizados não são perfeitos, pois nunca nos fornecem leituras exatas sobre as variáveis desejadas, no qual sempre é introduzida a sua própria dinâmica, distorções associadas, que em geral não são levados em conta, alem da existência de ruídos que corrompem o sinal gerado por ele.

Levando-se em consideração isto, admitir um modelo determinístico é uma hipótese não adequada. Desta maneira há a necessidade de se trabalhar com modelos estocásticos, no qual trabalharemos com o Filtro de Kalman.

5.1.1. Fusão Sensorial

A fusão dos sensores (bússola, GPS e IMU) foi realizada de forma indireta, de modo a tornar o filtro mais robusto a falhas do filtro. Se a falha for

identificada, o filtro pode ser reinicializado ou reconfigurado, assim, aumenta a sua confiabilidade. Observe abaixo a estrutura do filtro utilizado na forma indireta.

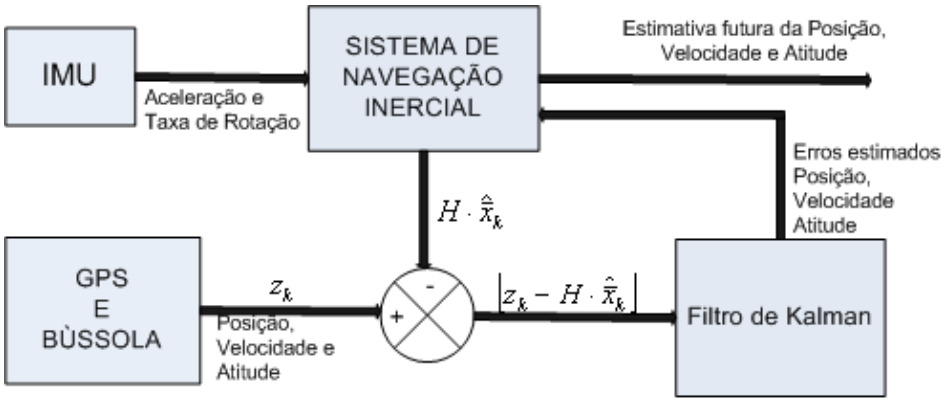

Figura 5.1 – Fusão sensorial indireta utilizando o Filtro de Kalman

Pode-se observar que o filtro desacopla a malha principal e opera de forma complementar aos dados fornecidos pela IMU.

Os dados referentes às características das UMI estão no Anexo B.

5.1.2. Estudo do Filtro de Kalman

O Filtro de Kalman tem por finalidade de realizar uma estimação de estados. Ele é um algoritmo computacional que tem como entradas todas as medidas disponíveis sobre o sistema e como saídas as componentes do vetor de estado do sistema.

O filtro realiza conjuntamente a fusão sensorial, gerando assim as estimativas dos estados e efetuando a correção destas estimativas através de sinais medidos da Bússola e GPS.

Quando algumas condições são satisfeitas, o filtro de Kalman pode ser considerado um estimador ótimo e minimiza a covariância do erro estimado, pois, a incerteza no final do processo é a menor entre todas as outras geradas por outras aproximações.

O filtro pode ser empregado no sentido de tentar estimar o estado $\mathrm{x} \in R^n$ de um processo controlado em instantes discretos de tempos, que é governado

por uma equação de diferenças linear estocástico e uma equação de medição $z \in R^m$, observem as duas equações abaixo:

$$x_{k+1} = A \cdot x_k + B \cdot u_k + w_k \tag{5.1}$$

$$z_k = H \cdot x_k + G \cdot v_k \tag{5.2}$$

Onde, A é a matriz de transição de estados, B é a matriz de entradas de controle, H é a Matriz de leitura dos sensores, G é a matriz de ruídos de medição, x_k é o vetor de estados, z_k é o vetor de medições do processo no instante t_k, w_k e v_k são ruídos brancos de media nula do processo e da medição, respectivamente e u_k é vetor entrada de controle.

O Filtro de Kalman realiza a estimativa de um processo utilizando uma forma de controle por realimentação. O filtro estima o estado do processo em um instante de tempo e após isto, faz-se a realimentação a partir de medições. Deste modo, as equações do filtro são divididas em dois grupos, propagação e atualização.

As equações de propagação correspondem em projetar a frente (com relação ao tempo) o estado corrente e a estimativa da covariância do erro de modo a obter uma estimativa para o próximo instante de tempo. As equações de atualização são responsáveis pela realimentação, ou seja, elas incorporam uma nova medição ao estado estimado a priori para obter uma estimativa melhorada a posteriori.

O filtro de Kalman discreto corresponde a uma sucessão de propagação e atualizações. Teremos, então:

a) Ciclo de Propagação

- Média: $\bar{x}_{k+1} = A \cdot \hat{x}_k + B \cdot u_k$ (5.3)
- Covariância: $\bar{P}_{k+1} = A \cdot \hat{P}_k \cdot A^T + C \cdot Q \cdot C^T$ (5.4)

b) Ciclo de Atualização

- Ganho de Kalman: $K_k = \bar{P}_k \cdot H^T \left[H \cdot \bar{P}_k \cdot H^T + G \cdot R \cdot G^T \right]^{-1}$ (5.5)

- Média: $\hat{\mathrm{x}}_{\mathrm{k}} = \hat{\bar{x}}_k + K \cdot \left[z_k - H \cdot \hat{\bar{x}}_k \right]$ (5.6)
- Covariância: $\hat{\mathrm{P}}_{\mathrm{k}} = \overline{P}_k - K_k \cdot H \cdot \overline{P}_k$ (5.7)

Onde, Q é a covariância do ruído de dinâmica, $G \cdot R \cdot G^T$ é a covariância do ruído de medição, P é a matriz covariância dos estados, C é a matriz de ruídos da dinâmica, o sobrescrito – é para valores na propagação e ^ para valores imediatamente após a atualização.

O primeiro passo do algoritmo é dado pelo ciclo de atualização, onde primeiramente se calcula o ganho de Kalman do filtro, depois a média e a covariância. O segundo passo é dado pelo ciclo de propagação, onde se calcula a média e a covariância realizando assim, uma estimativa posteriori (em um instante de tempo futuro). Após cada ciclo de atualização e propagação o processo é repetido através da estimativa posteriori anterior que é utilizada para predizer uma nova estimativa a priori. Observe a figura 5.2 que apresenta o ciclo do algoritmo do Filtro de Kalman discreto.

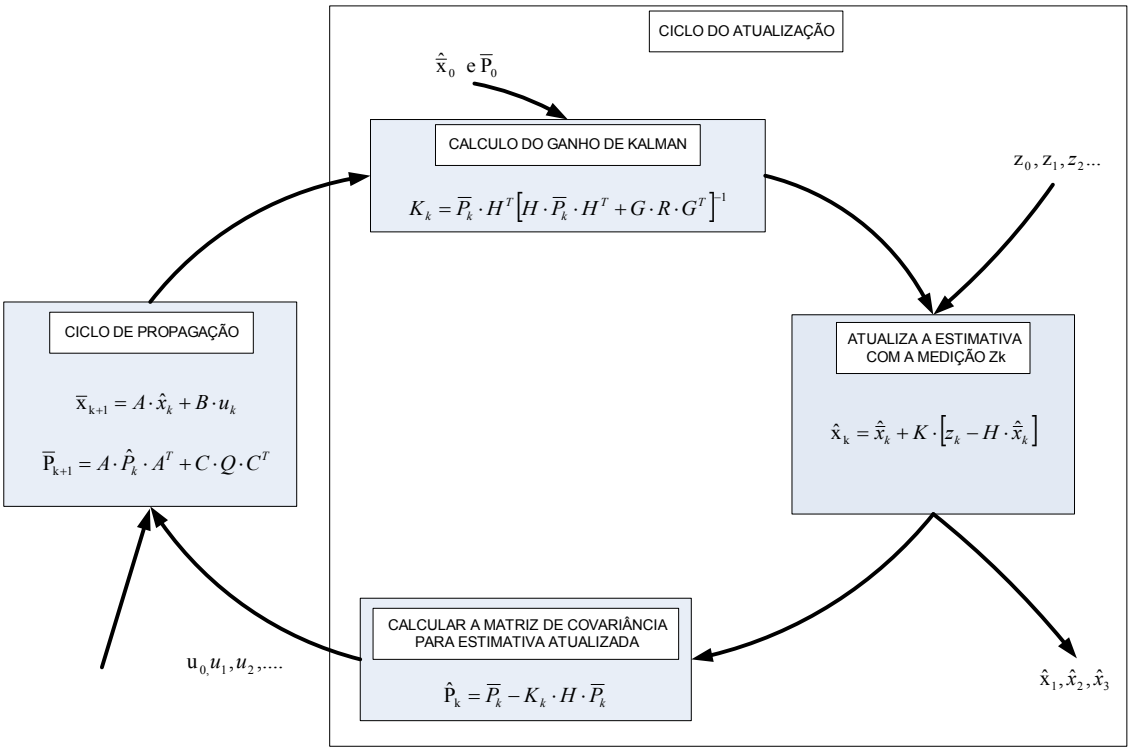

Figura 5.2 – Esquema simplificado do algoritmo de Kalman

A covariância do ruído de medição pode ser estimada a partir das especificações do GPS e Bússola. Já a covariância do ruído de processo não é bem conhecida, mas pode-se estimar a partir da covariância da medição da unidade inercial de medição (UMI). Já com relação à covariância dos estados, tem-se uma maior dificuldade em realizar a sua estimação, mas a partir do estado inicial que é dado pelo GPS e pela Bússola pode-se fazer esta estimação.

5.1.2.1. Modelo da Dinâmica de Processo de Estimação

O modelo da dinâmica de processo que será apresentado foi desenvolvido em Amianti (2007). Para desenvolver este modelo levou-se em consideração que observador de estados deve realizar a estimativa dos estados mesmo sem ter o conhecimento da dinâmica de vôo da aeronave, pois assim, consegue-se obter um observador de estados robusto para qualquer momento do vôo. Assim, utilizando os seguintes sensores, bússola, GPS, IMU, deve-se estimar a posição inercial, a velocidade e a atitude do veiculo.

O modelo proposto utiliza as leis do movimento de Newton que possibilita obter a velocidade e a posição a partir da Aceleração. Pelas leis de Newton, a aceleração pode ser expressa por:

$$\frac{\mathrm{dv}}{\mathrm{dt}} = a \tag{5.8}$$

Admita-se que a aceleração vem da IMU, integrando no tempo, tem-se a velocidade, que é dada por:

$$v = v_0 + at \tag{5.9}$$

A posição pode ser estressa por:

$$\frac{\mathrm{dp}}{\mathrm{dt}} = v = v_0 + at \tag{5.10}$$

$$p = p_0 + v_0 t + \frac{at^2}{2} \tag{5.11}$$

Discretizando em intervalos de tempo igual a T as equações (5.9) e (5.11) e representado vetorialmente, tem-se:

$$\{v_{k+1}\} = \{v_k\} + \{a_k\}T \tag{5.12}$$

$$\{p_{k+1}\} = \{p_k\} + \{v_k t\} + \frac{\{a_k\}T^2}{2} \tag{5.13}$$

Fazendo o mesmo procedimento para as taxas de Rotações adquiridas pela IMU, temos que a posição angular é dada por:

$$\frac{\mathrm{d}\theta}{\mathrm{dt}} = \omega \tag{5.14}$$

$$\theta = \theta_0 + \omega t \tag{5.15}$$

Discretizando em intervalos de tempo igual a T a equação (5.15) e representado vetorialmente, tem-se:

$$\{\theta_{k+1}\} = \{\theta_k\} + \{\omega_k\}T \tag{5.15}$$

Estas equações definem o modelo discreto da dinâmica do processo estimado e podem ser combinada para fornecer uma representação no espaço de estados discreto, do tipo:

$$\overline{\mathrm{x}}_{\mathrm{k+1}} = A \cdot \hat{x}_k + B \cdot u_k \tag{5.16}$$

Nas equações (5.12) e (5.13), o vetor $\{a_k\}$ contém as acelerações fornecidas pela IMU nas direções x, y, z, já transformadas para um sistema de

coordenada inercial e feia as correções determinísticas. Na equação (5.15), o vetor $\{\omega_k\}$ contém as taxa angulares nas direções x, y, z. Assim $\{a_k\}$ e $\{\omega_k\}$ são vetores de entrada. Deste modo, matricialmente tem-se:

$$\begin{Bmatrix} \overline{\mathrm{p}}_x \\ \overline{\mathrm{p}}_y \\ \overline{\mathrm{p}}_z \\ \overline{v}_x \\ \overline{v}_y \\ \overline{v}_z \\ \overline{\theta}_x \\ \overline{\theta}_y \\ \overline{\theta}_z \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & T & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & T & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{Bmatrix} \mathrm{p}_x \\ \mathrm{p}_y \\ \mathrm{p}_z \\ v_x \\ v_y \\ v_z \\ \theta_x \\ \theta_y \\ \theta_z \end{Bmatrix} + \begin{bmatrix} \frac{T^2}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{T^2}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{T^2}{2} & 0 & 0 & 0 \\ T & 0 & 0 & 0 & 0 & 0 \\ 0 & T & 0 & 0 & 0 & 0 \\ 0 & 0 & T & 0 & 0 & 0 \\ 0 & 0 & 0 & T & 0 & 0 \\ 0 & 0 & 0 & 0 & T & 0 \\ 0 & 0 & 0 & 0 & 0 & T \end{bmatrix} \cdot \begin{Bmatrix} a_x \\ a_y \\ a_z \\ \omega_x \\ \omega_y \\ \omega_z \end{Bmatrix} \tag{5.17}$$

Notar-se que este modelo não se aplica a sistemas com aceleração variável, entretanto para intervalos de amostragem T muito pequenos, pode-se admitir que a aceleração se mantém constante entre duas amostras consecutivas, sendo assim, as equações (5.12), (5.13) e (5.15) podem ser empregadas neste sistema.

5.1.3. Implementação do Filtro no MATLAB

Para realizar a implementação do filtro de Kalman no MATLAB, houve a necessidade de se estudar o seu toolbox, com a finalidade de se verificar a existência de algum conjunto de funções prontas que se pode implementar automaticamente o filtro, com isto poder-se-ia considerar a existência de uma solução já otimizada do filtro. Após este estudo observou-se que o MATLAB possui uma função que implementa o filtro de Kalman, mas, somente para uma variável, não fazendo desta maneira a fusão sensorial.

Assim, como se trabalha com diversos sensores e com diversos tempos de aquisição para cada sensor, não foi possível encontrar o conjunto de funções de maneira a realizar esta fusão sensorial. Conseqüentemente houve a necessidade de se programar o filtro utilizando a técnica apresentado no item 5.1.1. Observe abaixo, o algoritmo utilizado para implementar a fusão sensorial utilizando o filtro de Kalman.

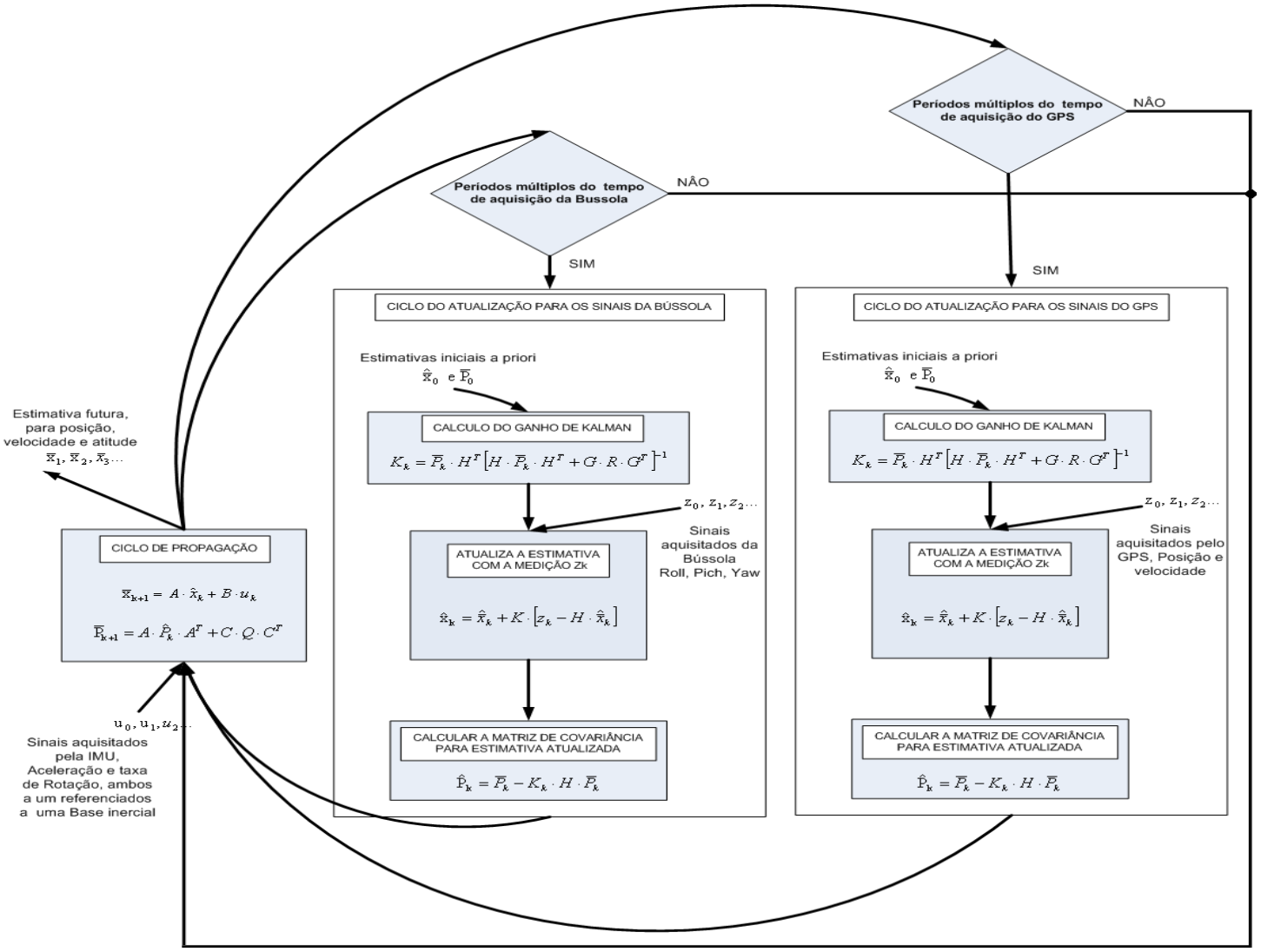

Figura 5.3 – Esquema que utilizado para implementar o filtro de Kalman com vários sensores sendo adquiridos em tempos diferentes

A princípio, estudou-se o código.m implementado por Amianti (2006), onde este pode ser observado no Anexo A. Mas, como a interação entre o MATLAB e o Rhapsody é dada pela criação de arquivos gerados pelo Real-Time Workshop do Simulink, houve a necessidade construir o Filtro de Kalman no Simulink.

Para isto, estudou-se também o toolbox do Simulink para verificar a existência de algum diagrama de bloco pronto que poderia implementar o Filtro de Kalman de um modo otimizado, utilizando como entrada diversos sensores que seriam aquisitados em tempos diferentes. Mas, chegou-se a mesma conclusão, quando se estudou o toolbox do MATLAB, não sendo possível encontrar um diagrama de bloco que se realiza esta fusão sensorial.

Outro ponto estudado no toolbox do SIMULINK, foi à verificação da existência de um diagrama de blocos onde poderia estar implementado o Filtro de Kalman Estendido, mas, o Simulink não tinha nenhum diagrama de blocos sobre o filtro estendido.

A principal vantagem de se utilizar o Simulink para a geração do programa em C++ pelo Real-Time Workshop é a possível utilização de funções já otimizadas como a de inversão e transposição de matrizes, ou seja, realizar operações matemáticas com matrizes. Mas, uma desvantagem que pode ser apontada em utilizar o Simulink é devido à impossibilidade de atualizar o código em C/C++, pois não se tem acesso a todas as bibliotecas que o programa inclui no código principal do programa criado pelo Real-Time Workshop (só consegue-se observar as chamadas de funções nos arquivos.h), deste modo praticamente dificulta ou até mesmo impossibilita alterações futuras diretamente no código em C/C++, caso seja necessário.

Quando se realizou este estudo, foi construída uma arvore que mostra as chamadas das bibliotecas incluídas no arquivo principal, criado pelo Real-Time Workshop, observe esta arvore na figura 5.4.

Apesar desta desvantagem apontada, optou-se por utilizar o Simulink, pela facilidade de implementação e também pela possibilidade de realizar simulações para que futuramente se estude os parâmetros do filtro, como por exemplo, a

matriz de covariância, de modo a otimizar o seu desempenho com relação a sua precisão.

Caso os requisitos de tempo real não forem atendidos devido a esta caixa preta gerada pelo Simulink, haverá a necessidade de se programar em C++ as funções que realizam as operações matemáticas com matrizes, como a inversa, multiplicação e transposição de matrizes.

O diagrama esquemático do filtro de Kalman implementado no simulador pode ser observado na figura 5.5. Neste simulador, o tempo de aquisição da bússola e do GPS são os mesmos, já para a IMU, a freqüência de aquisição é diferente destes outros sensores, mas ela sempre maior que da bússola e do GPS.

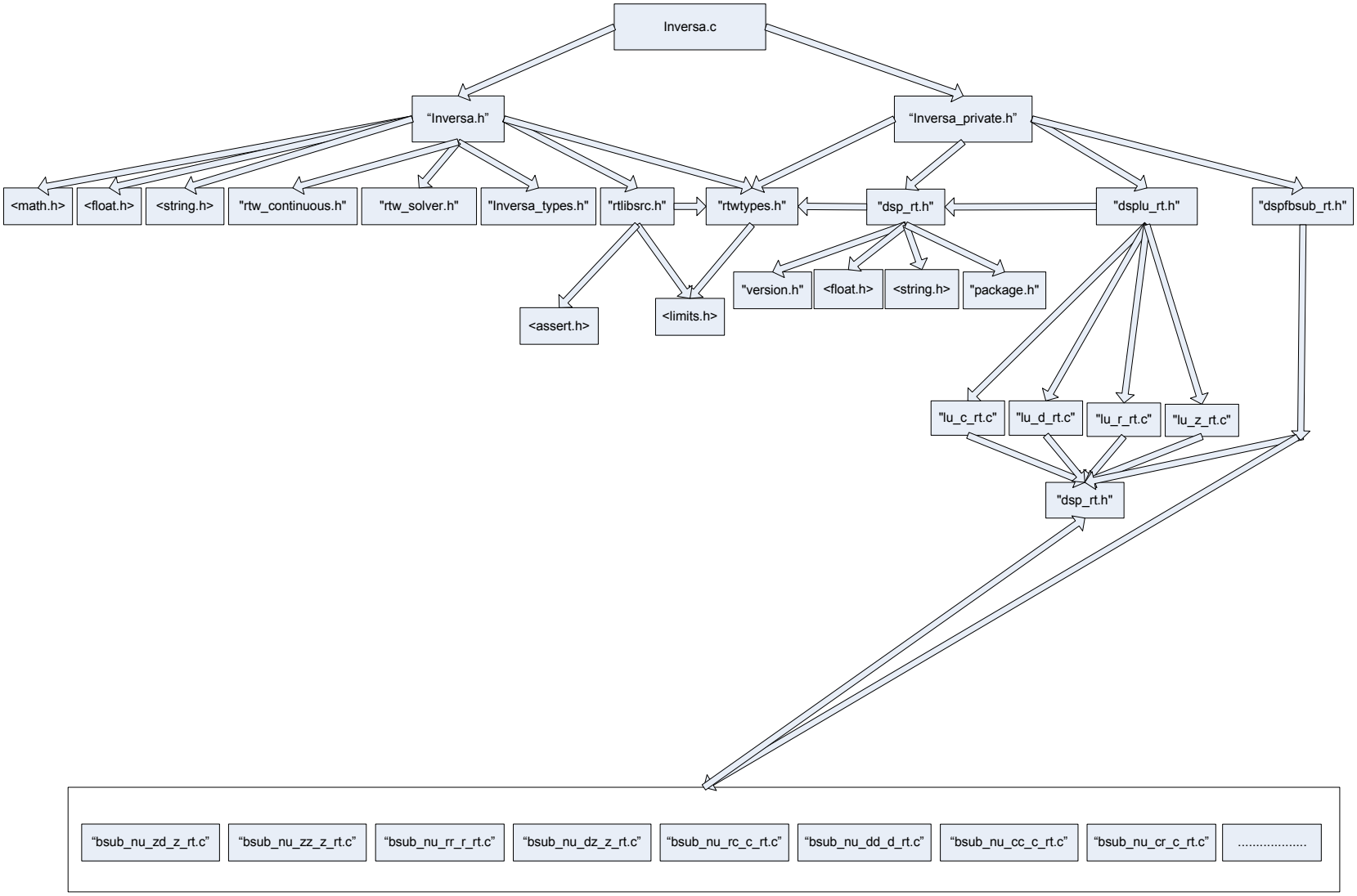

Figura 5.4 – Esquema de chamada dos Arquivos Criados pelo MATLAB

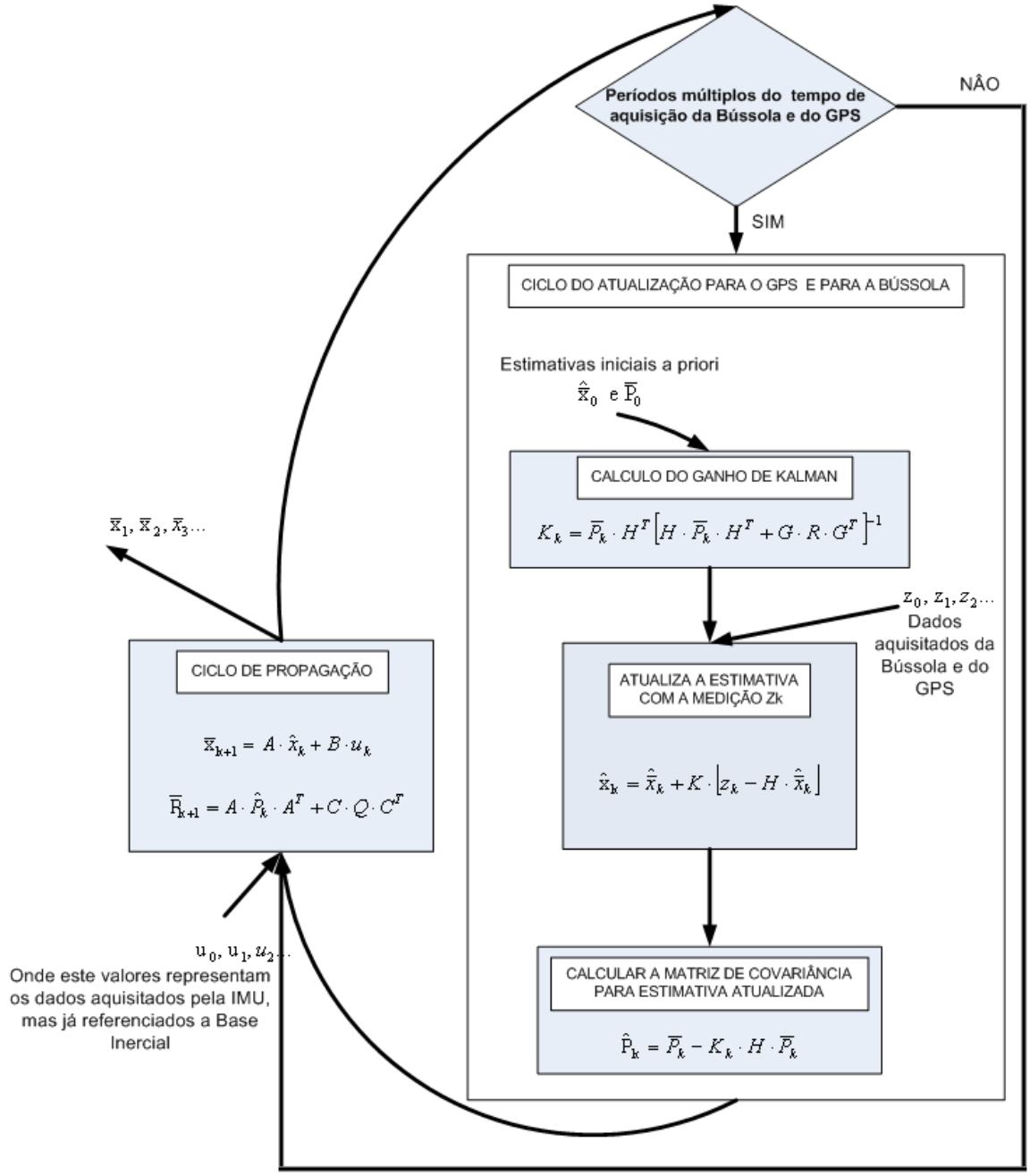

Figura 5.5 – Diagrama esquemático do Simulador Implementado

Para implementar o Filtro de Kalman no Simulink houve a necessidade de dividi-lo em subsistemas para facilitar a visualização e compreensão do seu funcionamento, além de facilitar a sua atualização futuramente por outra pessoa caso seja necessário. A sua implementação pode ser observada abaixo:

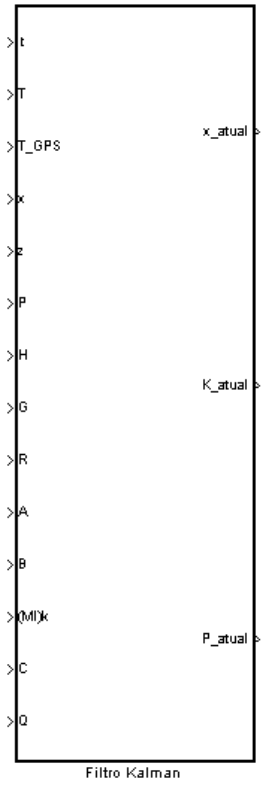

Figura 5.6 – Subsistema principal do Filtro de Kalman

Este Bloco se divide em mais dois sistemas os Ciclos de Atualização e o de Propagação. O Ciclo de Atualização pode ser visto abaixo:

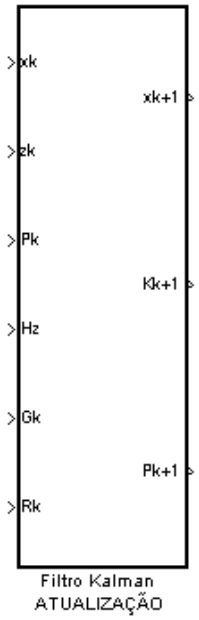

Figura 5.7 – Subsistema do Ciclo de Atualização do Filtro de Kalman

O Ciclo de Atualização do Filtro de Kalman é composto pelo calculo do ganho, da média e da covariância, para cada um destes cálculos foram construídos um subsistema que contem as equações já descritas anteriormente.

O Ciclo de Propagação do Filtro de Kalman é composto pelo calculo da média e da covariância, para cada um destes cálculos foram construídos um subsistema. Outra observação importante a ser referido para este ciclo é: sua linearização, para facilitar o seu calculo isto não prejudica muito a precisão, pois os períodos de amostragem dos sinais dos sensores são pequenos, mas caso queira aumentar a precisão do filtro pode-se equacionar um modelo não linear que se adeqüei ao projeto, isto pode ser utilizado no próximo capitulo.

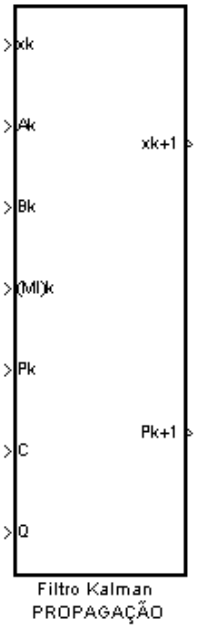

Figura 5.8 – Subsistema do Ciclo de Propagação do Filtro de Kalman

Outra observação importante que se pode fazer sobre este programa criado para o Filtro de Kalman é que ele pode ser facilmente convertido para o Filtro de Kalman estendido, sendo que para isto basta somente utilizar um modelo no qual as matrizes de transição de estados (A), matriz de entradas de controle (B) e a de leitura dos sensores (H) sejam dependentes do tempo, esta etapa de poderá ser empregada neste projeto no futuramente quando será realizado o estudo para aperfeiçoamento do filtro.

Para a verificação do pleno funcionamento do filtro foi construído um simulador completo onde se trata o sinal que vem da bússola e do GPS no Ciclo de Atualização e o sinal da IMU no Ciclo de Propagação. Os sinais da bússola, do GPS, da IMU e da velocidade do avião estão armazenados em uma matriz denominada de Apoena, fornecida pela empresa Xmobots que está desenvolvendo o projeto do UAV.

Para obter esta matriz Apena, esta empresa fez uma simulação de um vôo, armazenado assim, os dados simulados do GPS, da IMU e da velocidade do avião em nesta matriz. A partir desta matriz, se seleciona e trata cada sinal que seria inserido no filtro.

Para tratar o sinal da IMU utilizou-se o modelo matemático desenvolvido por Amianti (2006) onde se faz a mudança de coordenadas do corpo (o avião) para uma unidade inercial (a base de controle).

O programa completo desenvolvido neste trabalho de formatura pode ser observado abaixo:

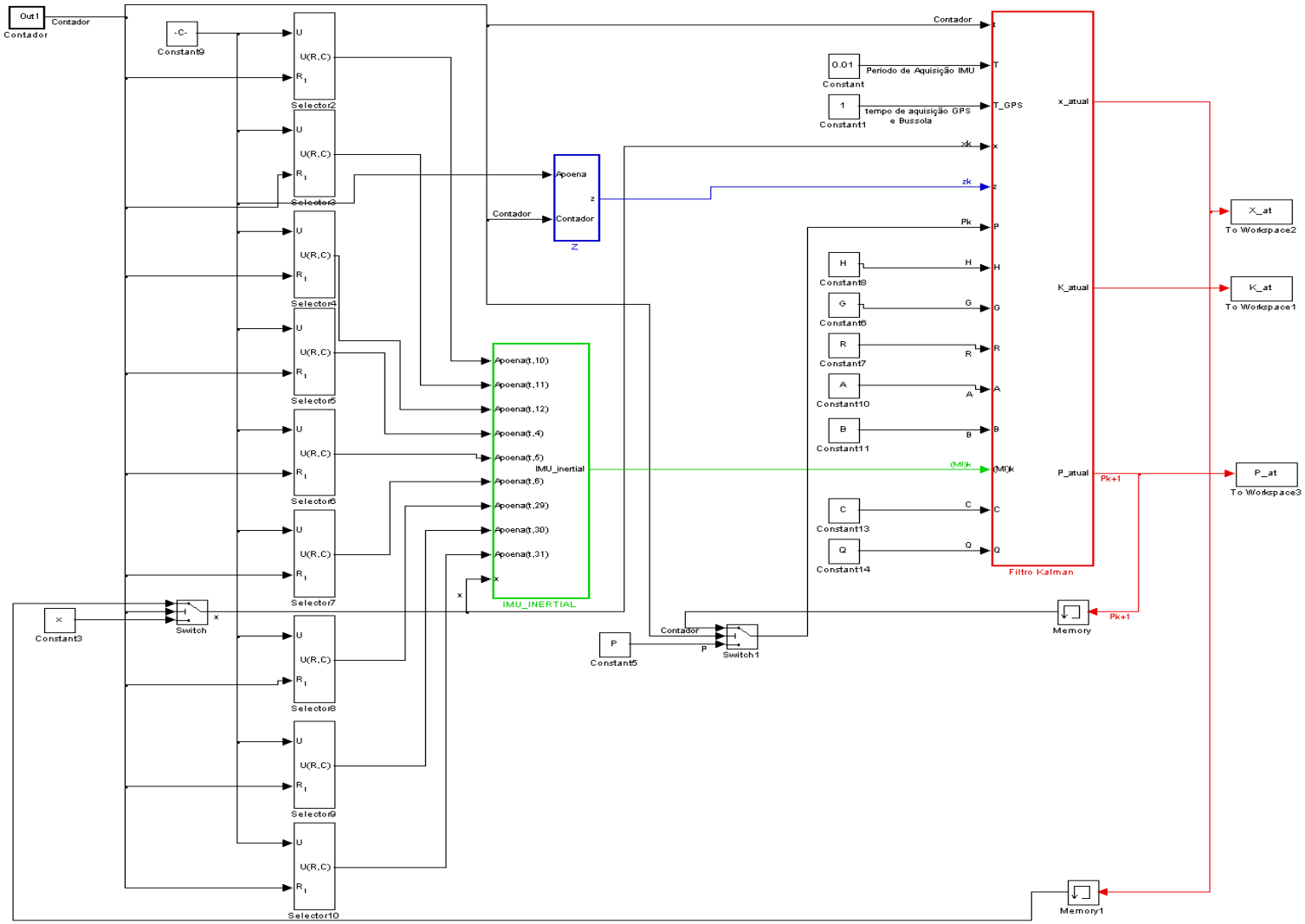

Figura 5.9 – Implementação completa para o Filtro de Kalman

5.1.4. Resultados das simulações Realizadas para o Filtro de Kalman Discreto

Para realizar a simulação utilizaram-se os dados apresentados no arquivo.m que se encontra no Anexo A, o diagrama de blocos da figura 5.9 e a matriz Apoena, que contem os dados da IMU, GPS e Bússola. Os resultados obtidos com esta simulação podem ser observados abaixo:

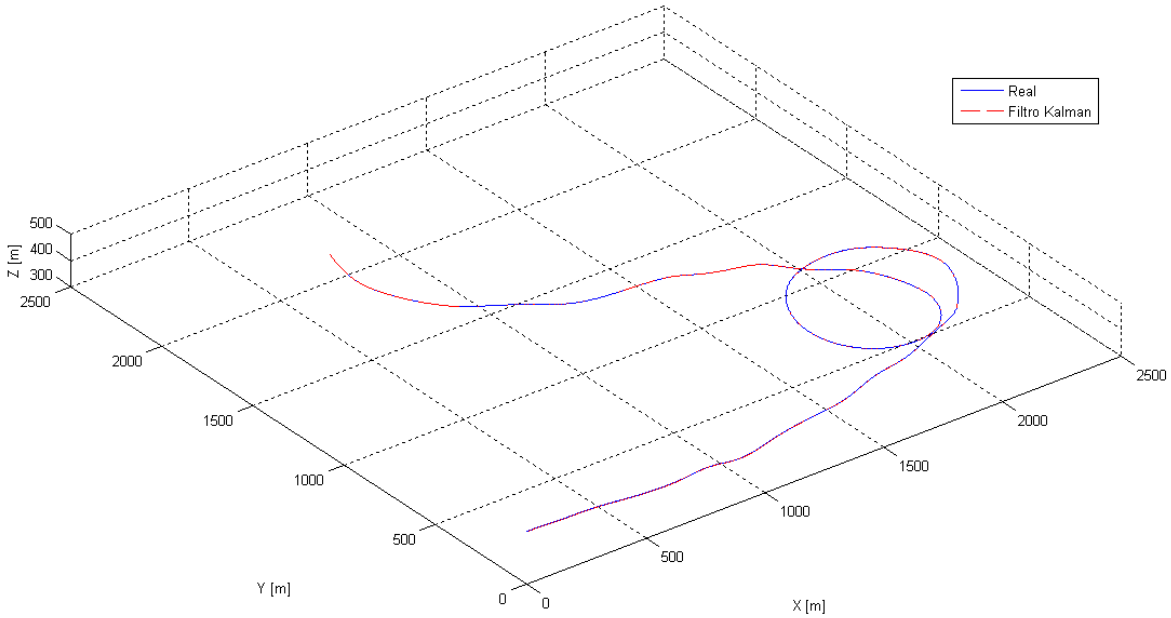

Figura 5.10 – Trajetória Percorrida e Simulada

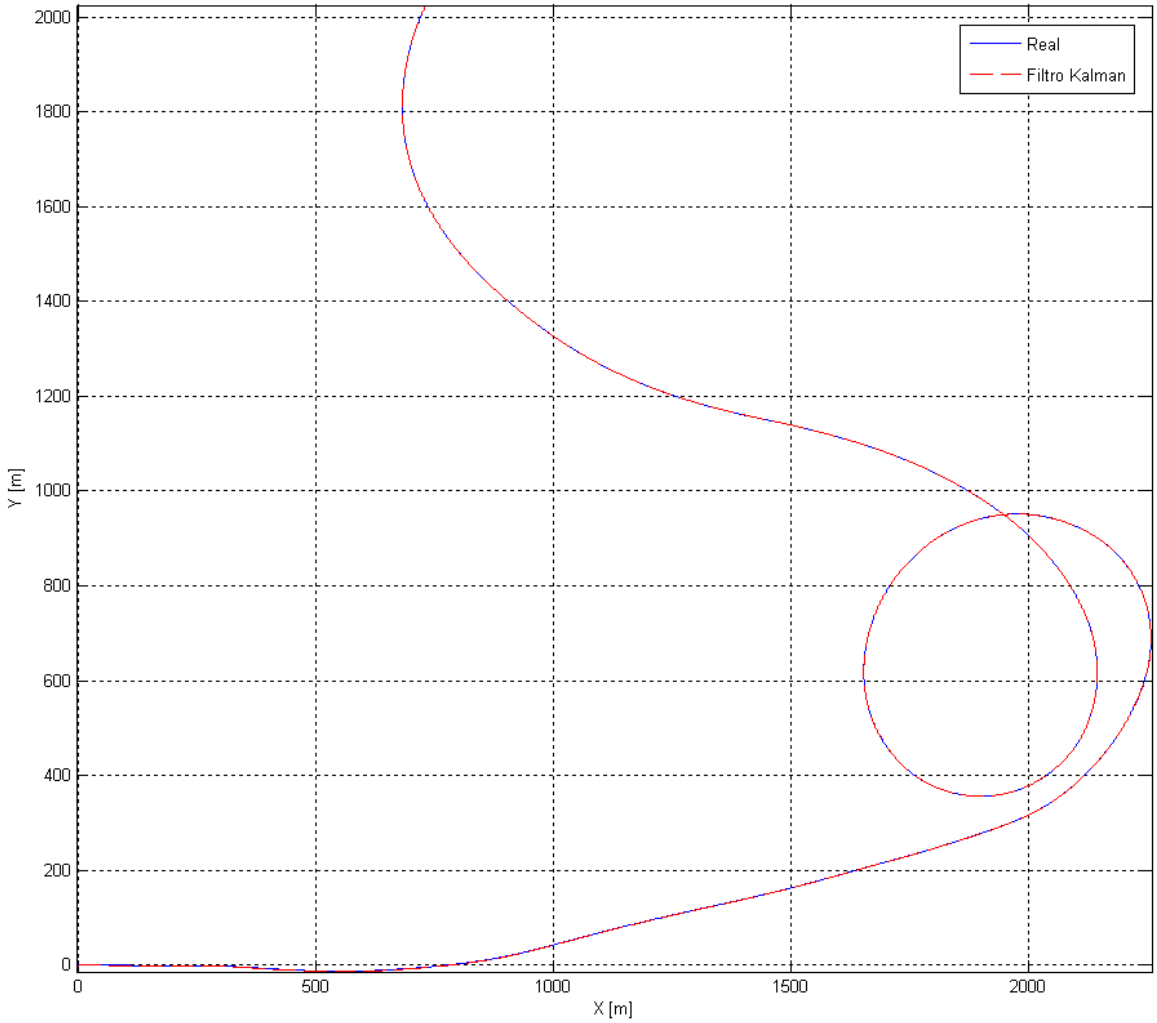

Figura 5.11 – Trajetória Percorrida e Simulada na direção x e y

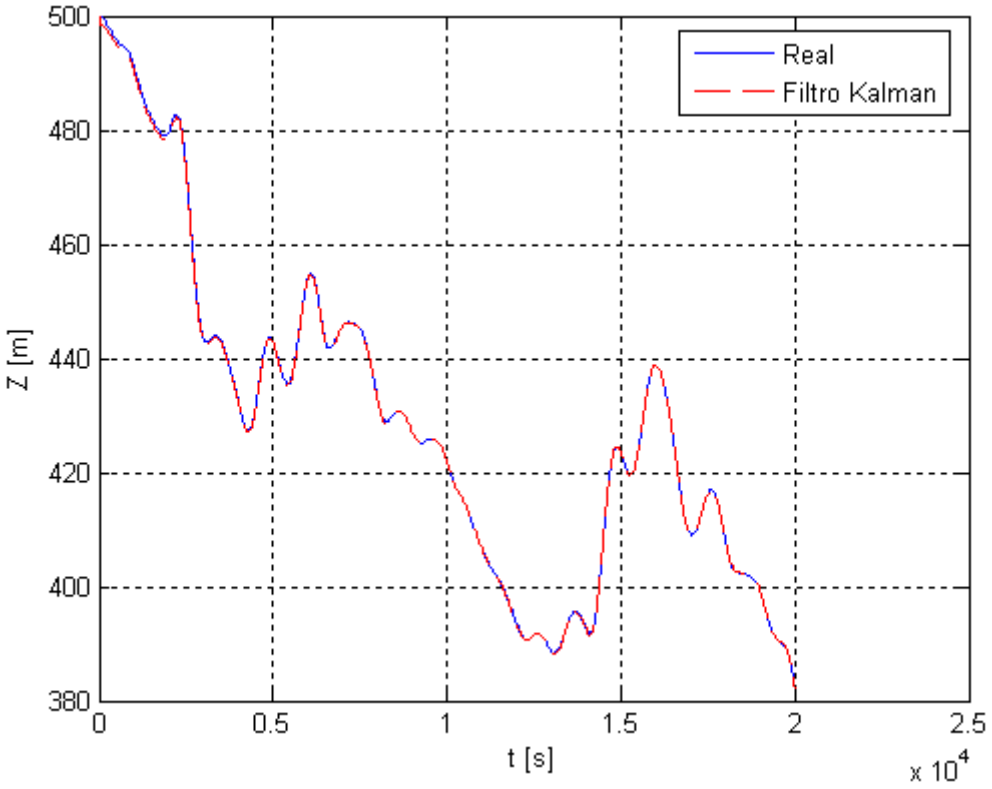

Figura 5.12 – Trajetória Percorrida e Simulada na direção z

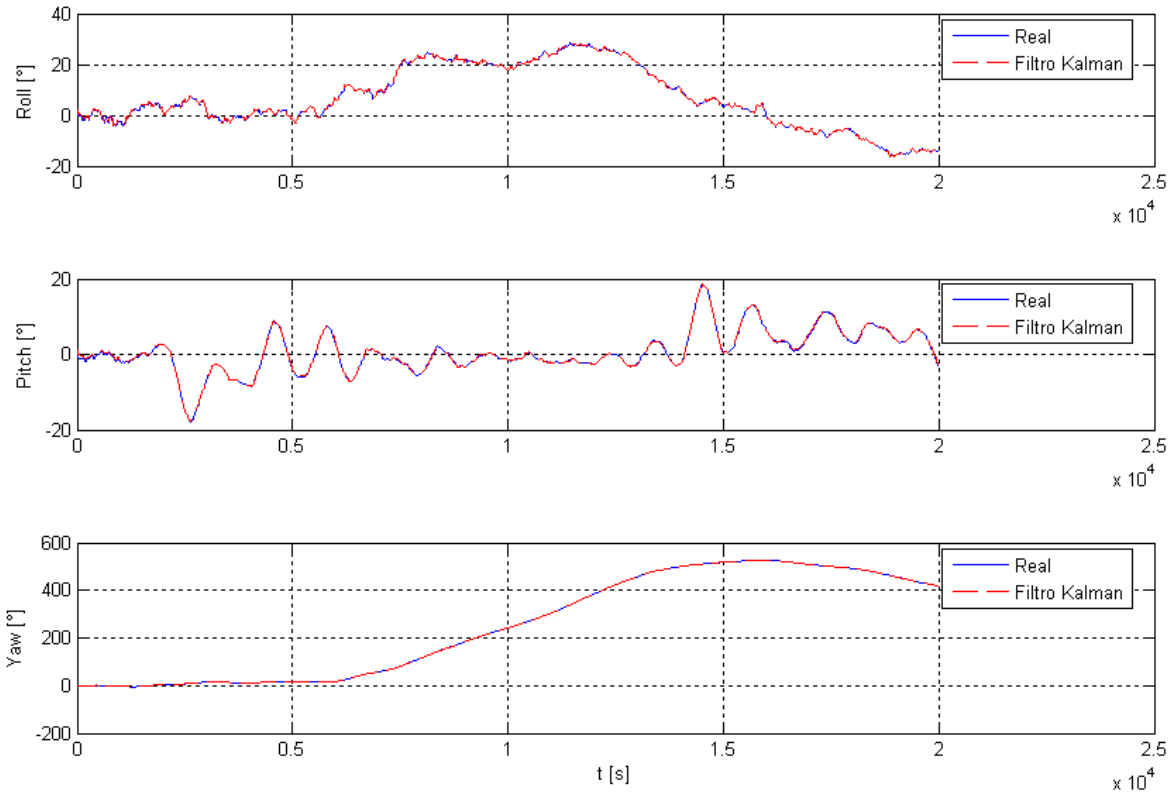

Figura 5.13 – Atitude

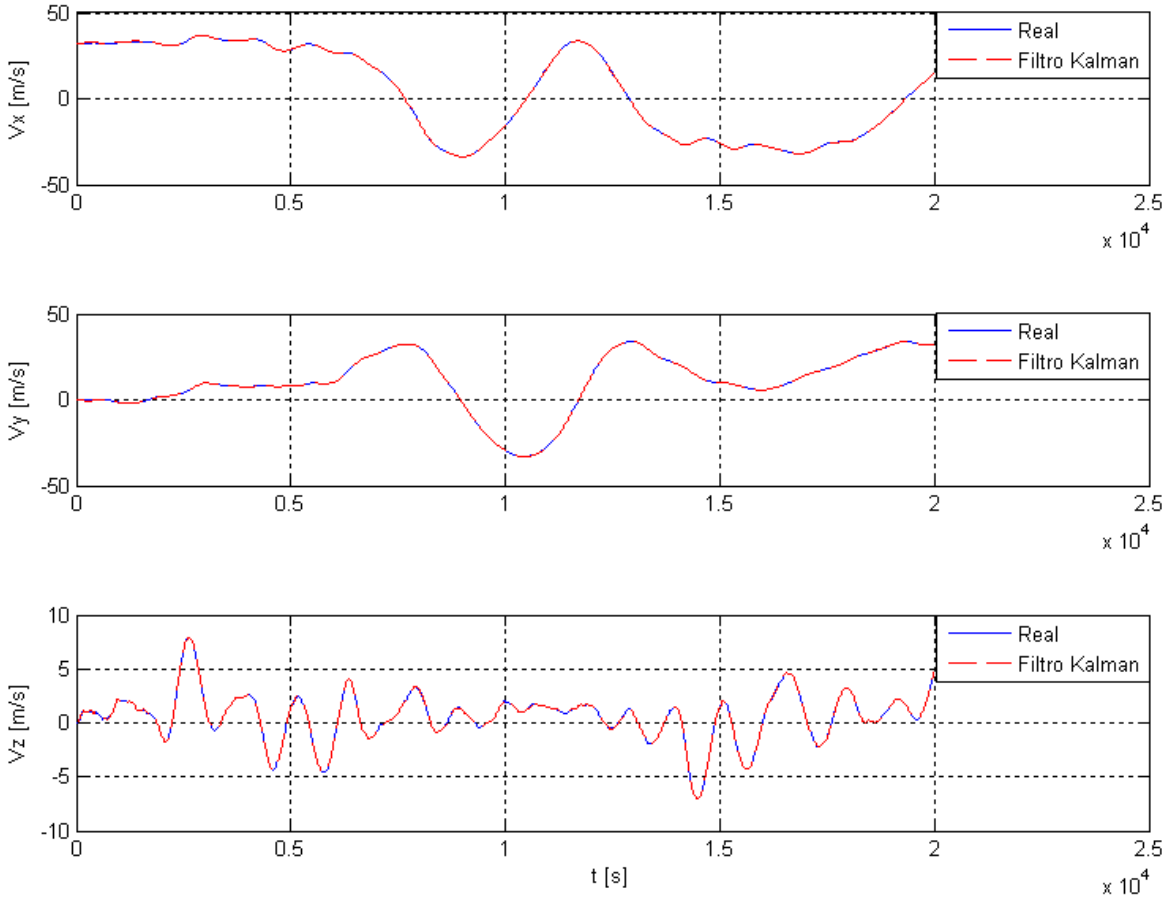

Figura 5.14 – Velocidade durante o Percurso

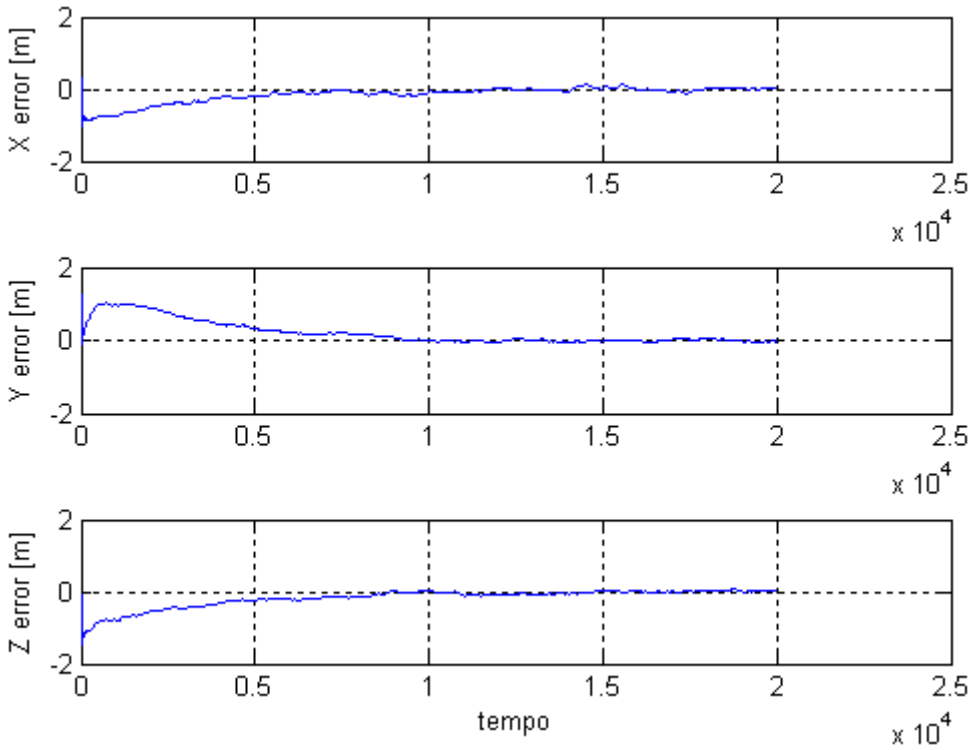

Figura 5.15 – Erro das Estimações para a Posição

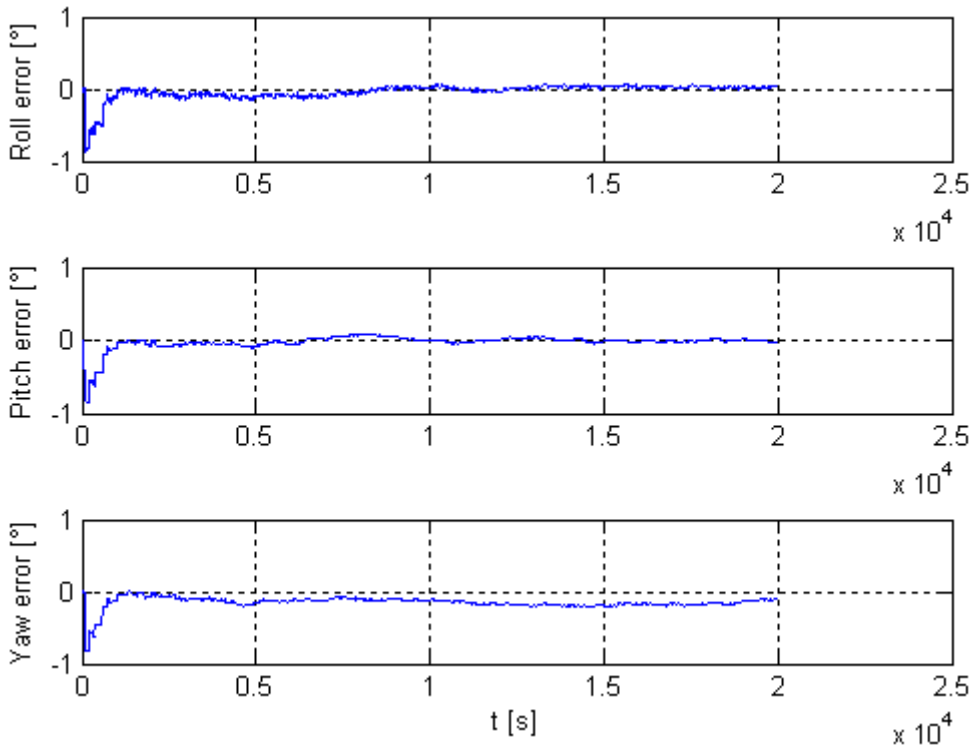

Figura 5.16 – Comparação do Erro das Estimações para a Atitude

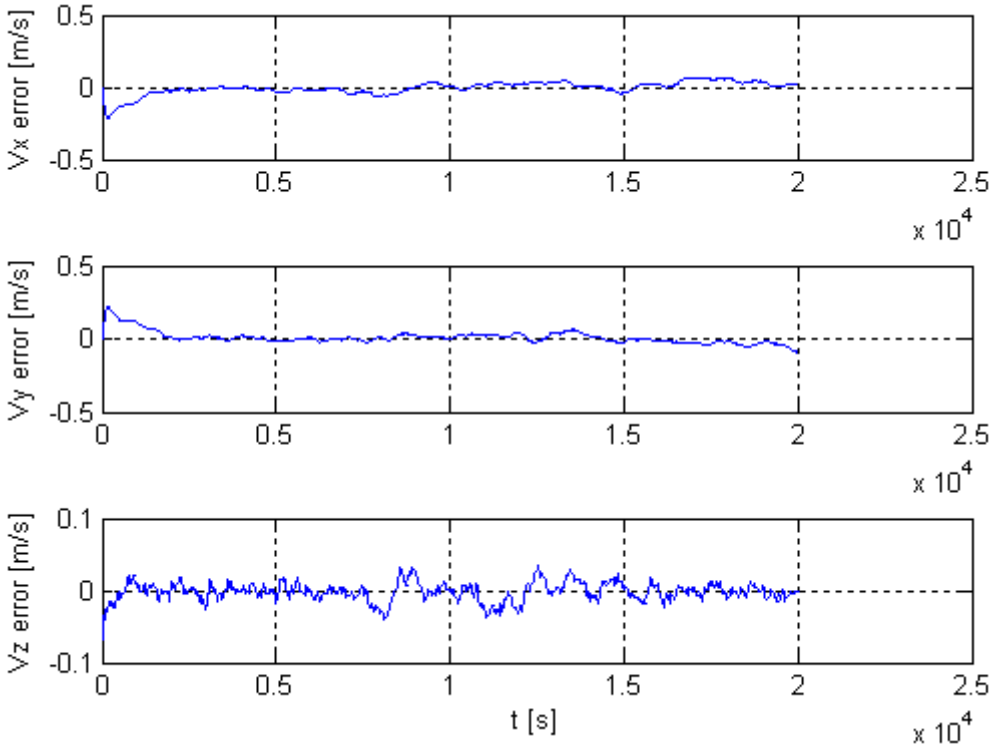

Figura 5.17 – Comparação do Erro das Estimações para a Velocidade

A partir destas simulações, pode-se concluir que o filtro possui uma boa estimativa, sendo que o maior erro para o posicionamento é de 1,47 m para a direção Z, no caso do erro de atitude o maior erro é menor que 1º para o roll, picht e yaw, para a velocidade o maior erro foi de 0,215 m/s na direção X e Y.

Ao fazer várias simulações variando a condição inicial da matriz covariância dos estados, pode se concluir que, quanto menor for os valores iniciais dos elementos desta matriz, mais difícil e demorada será a convergência dos estados para o valor esperado, próximo o do real, deste modo caso ela seja muito pequena, o a solução estimada pode até não convergir.

Outro parâmetro que se avaliou foi à variação dos períodos de amostragem da UMI e do GPS e Bússola. Ao se fazer a análise das respostas, observa-se que, quanto maior o período de amostragem dos sensores, pior será o valor estimado e também o período de amostragem que possui maior peso no filtro é o da UMI, pois ele é uma entrada de controle das estimativas futuras, assim, ao variarmos o seu período percebe-se uma grande variação na resposta do valor estimado.

5.2. Desenvolvimento de um Novo Modelo da Plataforma

Como se pode observar este modelo apresentado acima não corresponde corretamente ao modelo exigido para uma navegação aviônica, pois, ele não leva em considera a esfericidade da Terra, não tendo em consideração a aceleração centrípeta e nem a de Coriolis e além de considerar que a bússola fornece os ângulos roll, picth, yaw, corretamente o que não é verdade, devido ao inclino metro que ela possui ser influenciado pela a aceleração da plataforma, assim ele pode fornecer uma medida erronia da atitude. Para resolver este problema da bússola utilizaremos para determinar a atitude da plataforma o campo magnético fornecido por ela.

5.2.1. Determinação da posição da Plataforma

A aceleração da plataforma no sistema NED (NORTH, EAST, DOWN) pode ser representada pelas equações (1.1) e (1.2). Utilizam-se os índices: “n” para coordenadas de navegação, “e” para coordenadas da Terra, “i” para coordenadas inerciais e “p” para coordenadas da plataforma, e b^a para erro de bias nos sistema de coordenadas da plataforma.

$$\mathrm{a_e^n} = \mathrm{C_p^n}(\mathrm{a^p} - \mathrm{b^a}) + \mathrm{g} - [2\omega_{\mathrm{ie}}^{\mathrm{n}} + \omega_{\mathrm{en}}^{\mathrm{n}}] \otimes \mathrm{v_e^n} \tag{5.18}$$

$$\mathrm{a_e^n} = \mathrm{a^n} - \mathrm{b^n} + \mathrm{g} - 2\omega_{\mathrm{ie}}^{\mathrm{n}} \otimes \mathrm{v_e^n} - \omega_{\mathrm{en}}^{\mathrm{n}} \otimes \mathrm{v_e^n} \tag{5.19}$$

$$\mathrm{C_p^n} = \begin{bmatrix} (\mathrm{a^2}+\mathrm{b^2} - \mathrm{c^2} - \mathrm{d^2}) & 2(\mathrm{bc} - \mathrm{ad}) & 2(\mathrm{bd} + \mathrm{ac}) \\ 2(\mathrm{bc} + \mathrm{ad}) & (\mathrm{a^2}-\mathrm{b^2} + \mathrm{c^2} - \mathrm{d^2}) & 2(\mathrm{cd} - \mathrm{ab}) \\ 2(\mathrm{bd} - \mathrm{ac}) & 2(\mathrm{cd} + \mathrm{ab}) & (\mathrm{a^2} - \mathrm{b^2} - \mathrm{c^2} + \mathrm{d^2}) \end{bmatrix} \tag{5.20}$$

Onde a_e^n é a aceleração a ser utilizada para determinar a posição da plataforma no sistema NED, C_p^n é a matriz representada em quaternions para realizar a mudança de base das coordenada da plataforma para a coordenada no sistema NED, a_n é a aceleração medida pelo acelerômetro no sistema NED, b^n é o erro de bias do acelerômetro, g é a aceleração da gravidade, ω_{ie}^n o vetor rotação da Terra projetado no sistema NED, ω_{en}^n o vetor de velocidade angular do sistema NED em relação ao sistema ECEF, [a, b, c, d] são parâmetros do quaternions, o quarto termo da segunda equação é a aceleração de Coriolis, o quinto termo é a aceleração Centrípeta.

O motivo de se utilizar o quaternions para realizar a mudança de base é para simplificar o método computacional, pois ele não possui a necessidade de se trabalhar com equações transcendentais.

Deste modo pode-se perceber a necessidade de realizar correções nas acelerações medidas pelos acelerômetros, que se encontram na plataforma.

5.2.1.1. Determinação da aceleração de Coriolis

A aceleração de Coriolis representa a aceleração sentida pela plataforma ao se deslocar sobre um referencial girante. Seu equacionamento e dado por:

$$a^{coriolis} = -2\omega_{ie}^{n} \otimes v_e^n \tag{5.21}$$

A rotação da Terra no sistema de coordenada ECEF, é:

$$\omega_{ie}^{e} = \begin{bmatrix} 0 \\ 0 \\ \Omega \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0.00007292 \end{bmatrix} rad/s \tag{5.22}$$

Realizando a mudança de coordenadas para o sistema NED, tem-se

$$\omega_{ie}^{n} = C_e^n \omega_{ie}^{e} = \begin{bmatrix} -\sin(\lambda)\cos(\phi) & -\sin(\lambda)\sin(\phi) & \cos(\lambda) \\ -\sin(\phi) & \cos(\phi) & 0 \\ -\cos(\lambda)\cos(\phi) & -\cos(\lambda)\sin(\phi) & -\sin(\lambda) \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \Omega \end{bmatrix} = \begin{bmatrix} \Omega\cos(\lambda) \\ 0 \\ -\Omega\sin(\lambda) \end{bmatrix} \tag{5.23}$$

Onde λ é a Latitude e ϕ é a Longitude.

A velocidade da plataforma (v_e^p) pode ser projetada para o sistema NED, através de:

$$v_e^n = C_p^n v_e^p = C_p^n \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}_p = \begin{bmatrix} V_N \\ V_E \\ V_D \end{bmatrix}_n \tag{5.24}$$

Assim a aceleração de Coriolis pode ser expressa por:

$$a^{coriolis} = -2\omega_{ie}^{n} \otimes v_e^n = -2\begin{bmatrix} \Omega\cos(\lambda) \\ 0 \\ -\Omega\sin(\lambda) \end{bmatrix} \otimes \begin{bmatrix} V_N \\ V_E \\ V_D \end{bmatrix} = 2\begin{bmatrix} -V_d\Omega\sin(\lambda) \\ \Omega(V_N\sin(\lambda) + V_D\cos(\lambda)) \\ -V_E\Omega\cos(\lambda) \end{bmatrix} \tag{5.25}$$

Não se utiliza quaternions neste caso, devido à facilidade deste calculo e no filtro de Kalman implementado futuramente a latitude λ é um dos estados a ser estimado para determinar a trajetória da plataforma.

5.2.1.2. Determinação da aceleração Centrípeta

A aceleração centrípeta representa a aceleração sentida pela plataforma ao se deslocar sobre uma superfície curva. Seu equacionamento é dado por:

$$a_{centripeta} = -\omega_{en}^{n} \otimes v_{e}^{n} \tag{5.26}$$

Onde v_e^n e ω_{en}^n são a velocidade linear e a velocidade angular da plataforma no sistema de coordenadas NED, respectivamente.

A velocidade angular ω_{en}^n é calculada multiplicando a velocidade angular da plataforma no sistema ECEF (ω_{en}^e) pela matriz de rotação que realiza a mudança de base do sistema ECEF para o NED.

A velocidade angular da plataforma no sistema ECEF (ω_{en}^e) pode ser calculada ao se observar a figura 5.18.

$$\omega_{en}^{e} = \begin{bmatrix} \dot{\lambda}\cos(\phi) \\ -\dot{\lambda}\sin(\phi) \\ \dot{\phi} \end{bmatrix} \tag{5.27}$$

Realizando a mudança de coordenadas para o sistema NED, obtém:

$$\omega_{en}^{n} = C_{e}^{n}\omega_{en}^{e} = \begin{bmatrix} -\sin(\lambda)\cos(\phi) & -\sin(\lambda)\sin(\phi) & \cos(\lambda) \\ -\sin(\phi) & \cos(\phi) & 0 \\ -\cos(\lambda)\cos(\phi) & -\cos(\lambda)\sin(\phi) & -\sin(\lambda) \end{bmatrix} \begin{bmatrix} \dot{\lambda}\sin(\phi) \\ -\dot{\lambda}\cos(\phi) \\ \dot{\phi} \end{bmatrix} \tag{5.28}$$

$$\omega_{en}^{n} = \begin{bmatrix} \dot{\phi}\cos(\lambda) \\ \dot{\lambda} \\ -\dot{\phi}\sin(\lambda) \end{bmatrix} \tag{5.29}$$

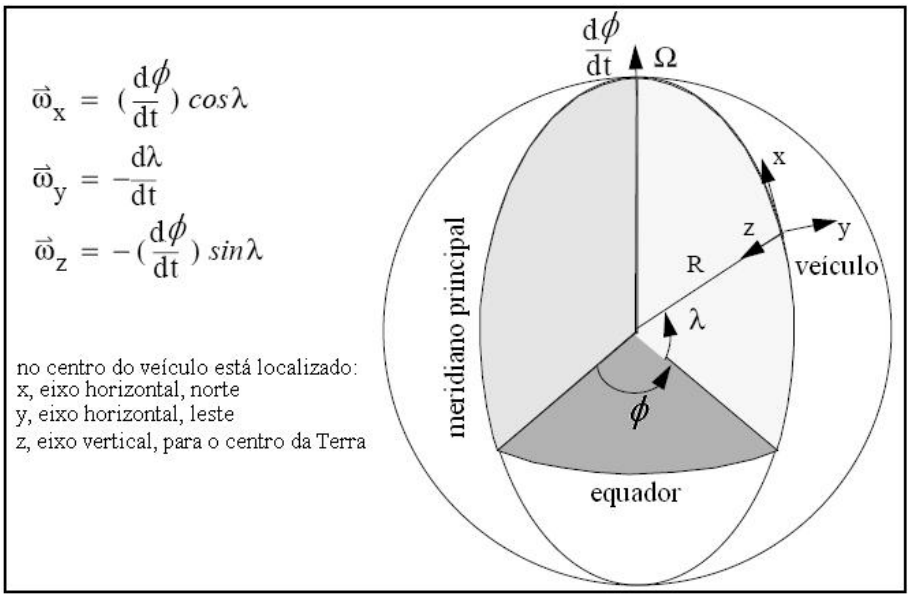

Figura 5.18 – Representação das coordenadas NED e ECEF

Substituindo a equação (5.34) na equação (5.29), obtém:

$$\omega_{en}^{n} = \left[\frac{V_E}{(R_\phi+h)} \quad \frac{-V_N}{(R_\lambda+h)} \quad \frac{-V_E Tan(\lambda)}{(R_\phi+h)}\right]^T \tag{5.30}$$

Deste modo, para corrigir a aceleração centrípeta faz-se:

$$a_{centripeta} = -\omega_{en}^{n} \otimes v_e^n = \begin{bmatrix} \frac{V_E}{(R_\phi+h)} \\ \frac{-V_N}{(R_\lambda+h)} \\ \frac{-V_E Tan(\lambda)}{(R_\phi+h)} \end{bmatrix} \otimes \begin{bmatrix} V_N \\ V_E \\ V_D \end{bmatrix} = \begin{bmatrix} \frac{-{V_E}^2 \sin(\lambda)}{(R_\phi+h)\cos(\lambda)} + \frac{V_N V_D}{(R_\lambda+h)} \\ \frac{V_E V_N \sin(\lambda)}{(R_\phi+h)\cos(\lambda)} + \frac{V_E V_D}{(R_\lambda+h)} \\ \frac{-{V_E}^2}{(R_\phi+h)} - \frac{{V_N}^2}{(R_\lambda+h)} \end{bmatrix} \tag{5.31}$$

5.2.1.3. Calculo da velocidade no sistema de coordenadas LLA

A velocidade da plataforma em torno da terra pode ser representada pela equação (5.32). Para demonstrá-la utilizarei a figura 5.19 e 5.20.

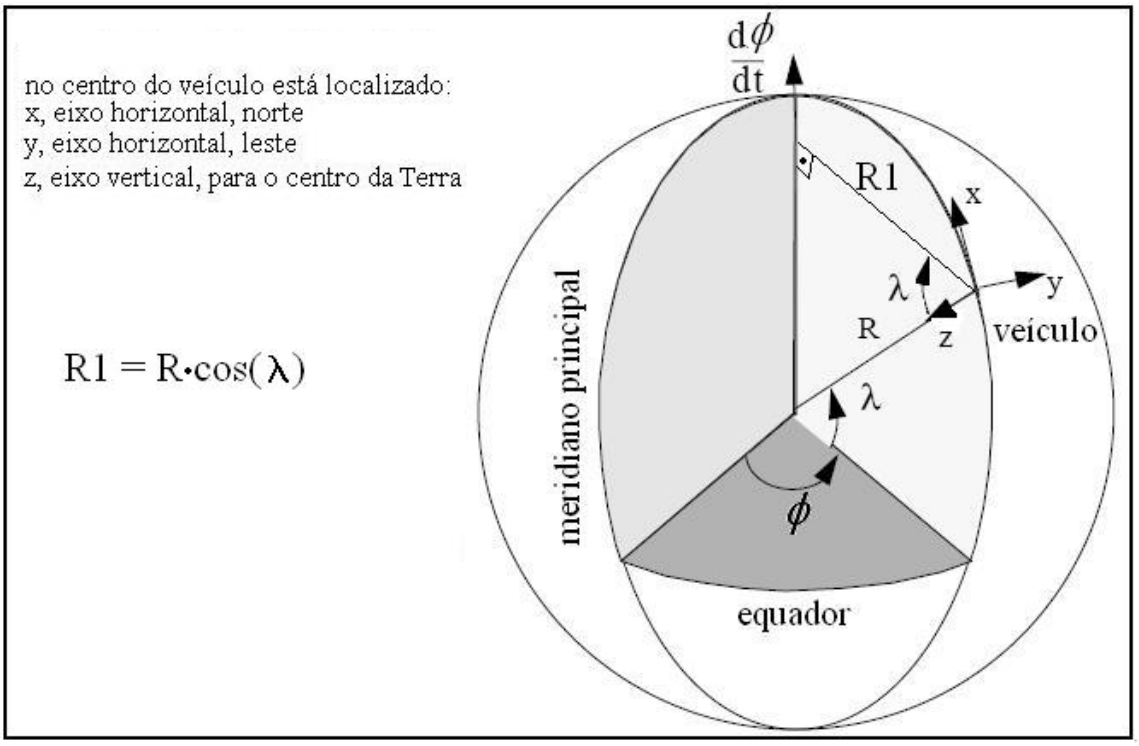

Figura 5.19 – Definição da Latitude, Longitude e do Raio a que elas estão sujeitas.

Para melhor observação da taxa da Longitude observe a figura abaixo, que é um corte paralelo ao equador e que passa pelo veículo. Deste modo, pode-se expressar esta taxa em função de R1 e V_E.

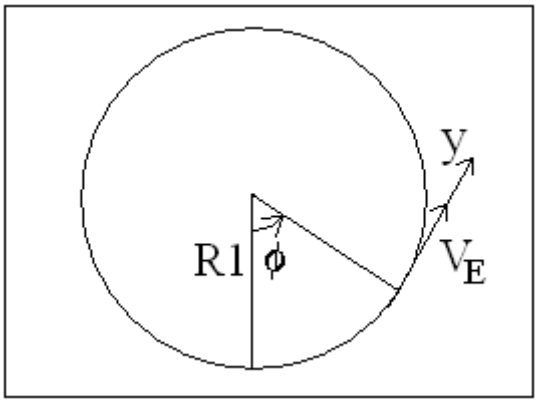

Figura 5.20 – Corte Paralelo ao Equador na Região do Veículo

$$\begin{bmatrix} \dot{\lambda} \\ \dot{\phi} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} \frac{V_N}{R} \\ \frac{V_E}{R1} \\ -V_D \end{bmatrix} = \begin{bmatrix} \frac{V_N}{R} \\ \frac{V_E}{Rcos(\lambda)} \\ -V_D \end{bmatrix} \tag{5.32}$$

Sendo que R é a soma da altura do veículo e o Raio da Terra, tem-se:

$$\begin{bmatrix} \dot{\lambda} \\ \dot{\phi} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} \frac{V_N}{(R_T+h)} \\ \frac{V_E}{(R_T+h)cos(\lambda)} \\ -V_D \end{bmatrix} \tag{5.33}$$

É notório que a Terra não é perfeitamente esférica, deste modo, para realizar uma modelagem mais consistente com a realidade e obtermos uma acuraria melhor durante a navegação, há a necessidade de modelarmos a Terra como se ela fosse um elipsóide, a fim de nos aproximarmos melhor da geometria verdadeira da Terra.

Desta forma, modelando a Terra de acordo com um elipsóide, as taxas de latitude e da longitude podem ser expressas em função do raio meridiano de curvatura (R_λ) e do raio transversal de curvatura (R_ϕ). Observe a figura 5.21.

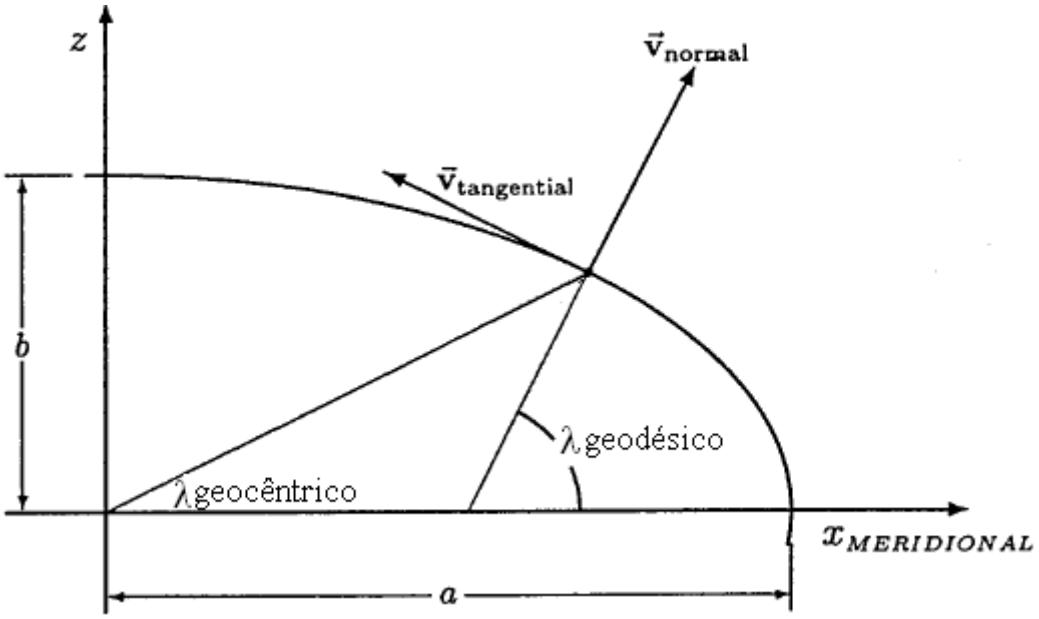

Figura 5.21 – Perfil elíptico aproximado da Terra

$$\begin{bmatrix} \dot{\lambda} \\ \dot{\phi} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} \frac{V_N}{(R_\lambda + h)} \\ \frac{V_E}{(R_\phi + h)cos(\lambda)} \\ -V_D \end{bmatrix} \tag{5.34}$$

Onde:

$$R_\lambda = \frac{a(1-e^2)}{(1-e^2(\sin(\lambda))^2)^{3/2}} \tag{5.35}$$

$$R_\phi = \frac{a}{(1-e^2(\sin(\lambda))^2)^{1/2}} \tag{5.36}$$

O raio de curvatura médio será $R_T = (R_\lambda R_\phi)^{1/2}$ e a maior excentricidade do elipsóide: $e = 0.08188191908426$. Nesta representação o $\lambda = \lambda_{geodésico}$.

5.2.2. Determinação da Atitude da Plataforma

Para resolver o problema da determinação da atitude da plataforma optou-se em realizar a propagação por quaternions utilizando os dados obtidos da taxa angular fornecido pela unidade de medição inercial (INS) já corrigido os erros de bias do giroscópio. A propagação é dada por:

$$\dot{q} = 0.5\begin{bmatrix} -b & -c & -d \\ a & -d & c \\ d & a & -b \\ -c & b & a \end{bmatrix}\begin{bmatrix} \omega_x - b_{g_x} \\ \omega_y - b_{g_y} \\ \omega_z - b_{g_z} \end{bmatrix} \quad (5.37)$$

Onde $[\omega_x,\ \omega_y,\ \omega_z]$ são velocidades angulares da plataforma no sistema de coordenadas dela e $[b_{gx},\ b_{gy},\ b_{gz}]$ são erros de bias do giroscópio no sistema de coordenadas da plataforma.

5.2.3. Conceituação do Filtro de Kalman Estendido

A essencial idéia do Filtro de Kalman Estendido (EKF – extended Kalman filter) foi proposta por Stanley F.Schmidt, onde o nome dado a ele era filtro “Kalman-Schmidt”.

A idéia principal do modelo do EKF é baseada na realização da linearização do modelo existente para que este seja utilizado no modelo de Kalman já descrito no capítulo anterior. Devido a esta linearização, tem-se a necessidade realizar a atualização das matrizes de transição de estado, de medição e de propagação dos erros, pois as derivadas parciais são avaliadas ao longo da trajetória, deste modo, recalculam-se estas matrizes após cada interação do algoritmo do filtro.

Há alguns cuidados que se tem que levar em consideração utilizar o EKF ao invés do filtro de Kalman linearizado, onde não se tem a necessidade de realizar as atualizações a cada interação do filtro. Estes cuidados são: deve-se conhecer bem as condições iniciais do sistema e também os erros não devem possuir grandes amplitudes.

De acordo com Brown,R.,2007, o processo de estimação pode ser representado por:

$$\dot{x}(t) = f(x,t) + w(t) \quad (5.38)$$

$$z = h(x,t) + v(t) \quad (5.39)$$

Onde f e v são funções conhecidas e w e v são erros brancos de média nula.

Assumindo que a trajetória aproximada $x^*(t)$ pode ser determinada a partir de algum meio, por exemplo, algum sensor. Esta será chamada de trajetória nominal ou de referencia e pode ser escrita por:

$$x(t) = x^*(t) + \Delta x(t) \tag{5.40}$$

Substituindo a (5.40) nas equações (5.38) e (5.39), tem-se:

$$x^*(t) + \Delta\dot{x}(t) = f(x^* + \Delta x, t) + w(t) \tag{5.41}$$

$$z = h(x^* + \Delta x, t) + v(t) \tag{5.42}$$

Assumindo que Δx é pequeno e fazendo a aproximação e expandindo por série de Taylor as funções f e h, obtém:

$$x^*(t) + \Delta\dot{x}(t) \approx f(x^*, t) + \left[\frac{\partial f}{\partial x}\right]_{x=x^*} \cdot \Delta x + w(t) \tag{5.43}$$

$$z = h(x^*, t) + \left[\frac{\partial h}{\partial x}\right]_{x=x^*} \cdot \Delta x + v(t) \tag{5.44}$$

Onde:

$$F = \frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \tag{5.45}$$

$$H = \frac{\partial h}{\partial x} = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_n}{\partial x_1} & \cdots & \frac{\partial h_n}{\partial x_n} \end{bmatrix} \tag{5.46}$$

Rearranjando a equação (5.44) tem-se:

$$[z - h(x^*, t)] = \left[\frac{\partial h}{\partial x}\right]_{x=x^*} \cdot \Delta x + v(t) \tag{5.47}$$

Ao se trabalhar no sistema discreto, o valor presente do filtro de Kalman é dado por $[z - h(x^*, t)]$ ao invés do valor de $[z]$, Brown,R.,2007. Logo, considerando a equação de atualização para o sistema discreto considerando a variação do tempo igual a t_k, tem-se:

$$\Delta\hat{x}_k = \Delta\hat{x}_k^- + K_k[z_k - h(x_k^*) - H_k\Delta\hat{x}_k^-] \tag{5.48}$$

Fazendo a associação dos termos $h(x_k^*)$ com $H_k\Delta\hat{x}_k^-$, pode-se calcular o erro da medição por:

$$Erro = z_k - \hat{z}_k^- \tag{5.49}$$

Agora substituindo (4.49) em (5.48) e somando x_k^* em ambos os lados da equação tem-se:

$$\underbrace{x_k^* + \Delta\hat{x}_k}_{\hat{x}_k} = \underbrace{x_k^* + \Delta\hat{x}_k^-}_{\hat{x}_k^-} + K_k[z_k - \hat{z}_k^-] \tag{5.50}$$

$$\hat{x}_k = \hat{x}_k^- + K_k[z_k - \hat{z}_k^-] \tag{5.51}$$

A atualização e a propagação da covariância do erro podem ser calculadas por:

$$P_k = (I - K_k H_k)P_k^- \tag{5.52}$$

$$P_{k+1}^- = A_k P_k A_k^T + Q_k \tag{5.53}$$

A atualização do ganho de Kalman pode ser calculada por:

$$\mathrm{K_k} = \mathrm{P_k^-}\mathrm{H_k^T}[\mathrm{H_k}\mathrm{P_k^-}\mathrm{H_k^T} + \mathrm{R_k}] \tag{5.54}$$

Um algoritmo esquemático do filtro pode ser observado abaixo:

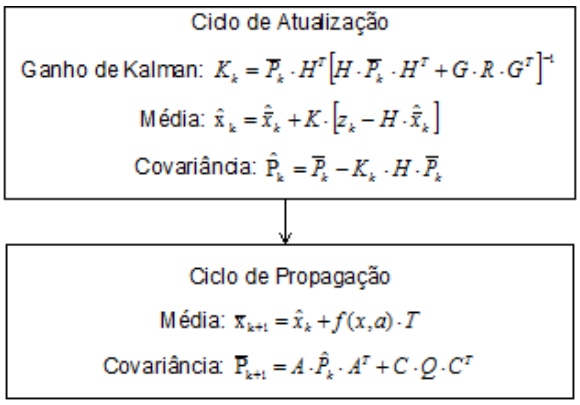

Figura 5.22 – Algoritmo esquemático do EKF

5.2.4. Modelo Proposto para Estimar os Estados da Plataforma

Para realizar a estimativa dos estados da plataforma, utiliza-se o Filtro de Kalman Estendido, pois, o modelo equacionado para os estados não é linear, deste modo, necessita-se linearizá-lo.

Como se deseja determinar a posição e a atitude da plataforma no espaço, os estados escolhidos são: posição do corpo nos três eixos no sistema de coordenadas LLA (Latitude, Longitude e Altura), as velocidades da plataforma no sistema de coordenadas NED, os erros de bias do acelerômetro e do giroscópio nos três eixos de coordenadas e a aceleração da gravidade no sistema NED.

A importância de se adotar os erros de bias do acelerômetro e do giroscópio e a aceleração da gravidade como estados, é poder computar os seus erros de medição na covariância do erro no EKF.

O vetor de estados adotado para este trabalho foi:

$$x = \begin{bmatrix} \lambda & \phi & h & V_N & V_E & V_D & a & b & c & d & g & b_{a_x} & b_{a_y} & b_{a_z} & b_{g_x} & b_{g_y} & b_{g_z} \end{bmatrix}^T \tag{5.55}$$

Onde: λ é a Latitude, ϕ é a Longitude, h é a altura, $[V_N \quad V_E \quad V_D]$ é a velocidade da plataforma no sistema de coordenadas NED, $\begin{bmatrix} b_{a_x} & b_{a_y} & b_{a_z} \end{bmatrix}$ são os erros de bias dos acelerômetros no sistema de coordenadas da plataforma, $[a \quad b \quad c \quad d]$ são os elementos do quartenions, $\begin{bmatrix} b_{g_x} & b_{g_y} & b_{g_z} \end{bmatrix}$ são os erros de bias dos giroscópios e g é a aceleração da gravidade no sistema de coordenada NED.

A função $f(x, a, \omega)$ adotada para este trabalho que rege a equação (5.38) é:

$$f(x,a,\omega)=\begin{bmatrix}\frac{V_N}{(R_\lambda+h)}\\ \frac{V_E}{(R_\phi+h)\cos(\lambda)}\\ -V_D\\ -\frac{{V_E}^2\sin(\lambda)}{(R_\phi+h)\cos(\lambda)}+\frac{V_N V_D}{(R_\lambda+h)}-2V_d\Omega\sin(\lambda)+a_N-b_{a_N}\\ \frac{V_e V_N\sin(\lambda)}{(R_\phi+h)\cos(\lambda)}+\frac{V_E V_D}{(R_\phi+h)}+2\Omega(V_N\sin(\lambda)+V_D\cos(\lambda))+a_E-b_{a_E}\\ \frac{-{V_E}^2}{(R_\phi+h)}-\frac{{V_N}^2}{(R_\lambda+h)}-2V_E\Omega\cos(\lambda)+g+a_D-b_{a_D}\\ \frac{1}{2}\begin{bmatrix}-b & -c & -d\\ a & -d & c\\ d & a & -b\\ -c & b & a\end{bmatrix}\begin{bmatrix}\omega_x-b_{g_x}\\ \omega_y-b_{g_y}\\ \omega_z-b_{g_z}\end{bmatrix}\\ 0\\0\\0\\0\\0\\0\\0\end{bmatrix} \quad (5.56)$$

Onde $[a_N \quad a_E \quad a_D]$ e $[b_N \quad b_E \quad b_D]$ podem ser calculados por:

$$\begin{bmatrix}a_N-b_{a_N}\\ a_E-b_{a_E}\\ a_D-b_{a_D}\end{bmatrix}=$$

$$=\begin{bmatrix}(a^2+b^2-c^2-d^2) & 2(bc-ad) & 2(bd+ac)\\ 2(bc+ad) & (a^2-b^2+c^2-d^2) & 2(cd-ab)\\ 2(bd-ac) & 2(cd+ab) & (a^2-b^2-c^2+d^2)\end{bmatrix}\begin{bmatrix}a_x-b_{a_x}\\ a_y-b_{a_y}\\ a_z-b_{a_z}\end{bmatrix} \quad (5.57)$$

Uma observação para esta implementação é que ela não será implementada utilizando o modelo elipsóide da Terra, para minimizar as contas, assim, $R_T=R_\lambda=R_\phi$.

Como no EKF trabalha-se no sistema discreto e também com o vetor $f(x,t)$ linearizado a necessidade de linearizá-lo:

$$A=\left[I+\frac{\partial f(x,t)}{\partial x}\Delta t\right] \quad (5.58)$$

Onde Δt é a variação do tempo e I uma matriz identidade.

A matriz “A” pode ser expressa por:

$$A = \left[\begin{array}{cccccccccc|cccccc} R_1 & R_2 & R_3 & R_4 & R_5 & R_6 & R_7 & R_8 & R_9 & R_{10} & R_{11} & R_{12} & R_{13} & R_{14} & 0^{6x1} & 0^{6x1} & 0^{6x1} \\ 0^{4x1} & 0^{4x1} & 0^{4x1} & 0^{4x1} & 0^{4x1} & 0^{4x1} & R_{15} & R_{16} & R_{17} & R_{18} & 0^{4x1} & 0^{4x1} & 0^{4x1} & 0^{4x1} & R_{19} & R_{20} & R_{21} \\ & & & & & 0^{7x10} & & & & & & & & I^{7x7} & & & \end{array}\right] \quad (5.59)$$

Onde:

0^{NxM} é uma matriz de zeros de N linhas e M colunas,

$$\mathrm{R}_1 = \begin{bmatrix} 1 \\ \frac{(\Delta T)\, V_E\, \sin(\lambda)}{\left(R_\phi + h\right)(\cos(\lambda))^2} \\ 0 \\ (\Delta T)\left(-\frac{{V_E}^2\left(1+(\tan(\lambda))^2\right)}{R_\phi + h} - 2\, V_D\, \Omega\, \cos(\lambda)\right) \\ (\Delta T)\left(\frac{V_E\, V_N\, \left(1+(\tan(\lambda))^2\right)}{R_\phi + h} + 2\,\Omega\,(V_N\, \cos(\lambda) - V_D\, \sin(\lambda))\right) \\ 2\,(\Delta T)\, V_E\, \Omega\, \sin(\lambda) \end{bmatrix} \quad (5.60)$$

$$\mathrm{R}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (5.61)$$

$$R_3 = \begin{bmatrix} -\frac{(\Delta T) V_N}{(R_\lambda + h)^2} \\ -\frac{(\Delta T) V_E}{(R_\phi + h)^2 \cos(\lambda)} \\ 1 \\ (\Delta T)\left(\frac{V_E{}^2 \tan(\lambda)}{(R_\phi + h)^2} - \frac{V_N V_D}{(R_\lambda + h)^2}\right) \\ (\Delta T)\left(-\frac{V_E V_N \tan(\lambda)}{(R_\phi + h)^2} - \frac{V_E V_D}{(R_\lambda + h)^2}\right) \\ (\Delta T)\left(\frac{V_E{}^2}{(R_\phi + h)^2} + \frac{V_N{}^2}{(R_\lambda + h)^2}\right) \end{bmatrix} \quad (5.62)$$

$$R_4 = \begin{bmatrix} \frac{(\Delta T)}{R_\lambda + h} \\ 0 \\ 0 \\ 1 + \frac{(\Delta T) V_D}{R_\lambda + h} \\ (\Delta T)\left(\frac{V_E \tan(\lambda)}{R_\phi + h} + 2\,\Omega \sin(\lambda)\right) \\ -2\,\frac{(\Delta T) V_N}{R_\lambda + h} \end{bmatrix} \quad (5.63)$$

$$R_5 = \begin{bmatrix} 0 \\ \frac{(\Delta T)}{(R_\phi + h)\cos(\lambda)} \\ 0 \\ -2\,\frac{(\Delta T) V_E \tan(\lambda)}{R_\phi + h} \\ 1 + (\Delta T)\left(\frac{V_N \tan(\lambda)}{R_\phi + h} + \frac{V_D}{R_\lambda + h}\right) \\ (\Delta T)\left(-2\,\frac{V_E}{R_\phi + h} - 2\,\Omega \cos(\lambda)\right) \end{bmatrix} \quad (5.64)$$

$$\mathrm{R}_6 = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ -(\Delta T) \\ (\Delta T)\left(\frac{V_N}{R_\lambda + h} - 2\,\Omega\,\sin(\lambda)\right) \\ (\Delta T)\left(\frac{V_E}{R_\lambda + h} + 2\,\Omega\,\cos(\lambda)\right) \\ 1 \end{bmatrix} \tag{5.65}$$

$$\mathrm{R}_7 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ (\Delta T)\left(2\,a\,(a_x - b_{a_x}) - 2\,d\left(a_y - b_{a_y}\right) + 2\,c\,(a_z - b_{a_z})\right) \\ (\Delta T)\left(2\,d\,(a_x - b_{a_x}) + 2\,a\left(a_y - b_{a_y}\right) - 2\,b\,(a_z - b_{a_z})\right) \\ (\Delta T)\left(-2\,c\,(a_x - b_{a_x}) + 2\,b\left(a_y - b_{a_y}\right) + 2\,a\,(a_z - b_{a_z})\right) \end{bmatrix} \tag{5.66}$$

$$\mathrm{R}_8 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ (\Delta T)\left(2\,b\,(a_x - b_{a_x}) + 2\,c\left(a_y - b_{a_y}\right) + 2\,d\,(a_z - b_{a_z})\right) \\ (\Delta T)\left(2\,c\,(a_x - b_{a_x}) - 2\,b\left(a_y - b_{a_y}\right) - 2\,a\,(a_z - b_{a_z})\right) \\ (\Delta T)\left(2\,d\,(a_x - b_{a_x}) + 2\,a\left(a_y - b_{a_y}\right) - 2\,b\,(a_z - b_{a_z})\right) \end{bmatrix} \tag{5.67}$$

$$\mathrm{R}_9 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ (\Delta T)\left(-2\,c\,(a_x - b_{a_x}) + 2\,b\left(a_y - b_{a_y}\right) + 2\,a\,(a_z - b_{a_z})\right) \\ (\Delta T)\left(2\,b\,(a_x - b_{a_x}) + 2\,c\left(a_y - b_{a_y}\right) + 2\,d\,(a_z - b_{a_z})\right) \\ (\Delta T)\left(-2\,a\,(a_x - b_{a_x}) + 2\,d\left(a_y - b_{a_y}\right) - 2\,c\,(a_z - b_{a_z})\right) \end{bmatrix} \tag{5.68}$$

$$\mathrm{R}_{10} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ (\Delta T)\left(-2\,d\,(a_x - b_{a_x}) - 2\,a\left(a_y - b_{a_y}\right) + 2\,b\,(a_z - b_{a_z})\right) \\ (\Delta T)\left(2\,a\,(a_x - b_{a_x}) - 2\,d\left(a_y - b_{a_y}\right) + 2\,c\,(a_z - b_{a_z})\right) \\ (\Delta T)\left(2\,b\,(a_x - b_{a_x}) + 2\,c\left(a_y - b_{a_y}\right) + 2\,d\,(a_z - b_{a_z})\right) \end{bmatrix} \tag{5.69}$$

$$\mathrm{R}_{11} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ (\Delta T) \end{bmatrix} \tag{5.70}$$

$$\mathrm{R}_{12} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ (\Delta T)\,(-a^2 - b^2 + c^2 + d^2) \\ (\Delta T)\,(-2\,bc - 2\,ad) \\ (\Delta T)\,(-2\,bd + 2\,ac) \end{bmatrix} \tag{5.71}$$

$$\mathrm{R}_{13} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ (\Delta T)\,(-2\,bc + 2\,ad) \\ (\Delta T)\,(-a^2 + b^2 - c^2 + d^2) \\ (\Delta T)\,(-2\,cd - 2\,ab) \end{bmatrix} \tag{5.72}$$

$$\mathrm{R}_{14} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ (\Delta T)\,(-2\,bd - 2\,ac) \\ (\Delta T)\,(-2\,cd + 2\,ab) \\ (\Delta T)\,(-a^2 + b^2 + c^2 - d^2) \end{bmatrix} \tag{5.73}$$

$$\mathrm{R}_{15} = \begin{bmatrix} \mathbf{1} \\ (\Delta T)\,(1/2\,\omega_x - 1/2\,b_{g_x}) \\ (\Delta T)\left(1/2\,\omega_y - 1/2\,b_{g_y}\right) \\ (\Delta T)\,(1/2\,\omega_z - 1/2\,b_{g_z}) \end{bmatrix} \tag{5.74}$$

$$R_{16} = \begin{bmatrix} (\Delta T)\,(-1/2\,\omega_x + 1/2\,b_{g_x}) \\ 1 \\ (\Delta T)\,(-1/2\,\omega_z + 1/2\,b_{g_z}) \\ (\Delta T)\left(1/2\,\omega_y - 1/2\,b_{g_y}\right) \end{bmatrix} \tag{5.75}$$

$$R_{17} = \begin{bmatrix} (\Delta T)\left(-1/2\,\omega_y + 1/2\,b_{g_y}\right) \\ (\Delta T)\,(1/2\,\omega_z - 1/2\,b_{g_z}) \\ 1 \\ (\Delta T)\,(-1/2\,\omega_x + 1/2\,b_{g_x}) \end{bmatrix} \tag{5.76}$$

$$R_{18} = \begin{bmatrix} (\Delta T)\,(-1/2\,\omega_z + 1/2\,b_{g_z}) \\ (\Delta T)\left(-1/2\,\omega_y + 1/2\,b_{g_y}\right) \\ (\Delta T)\,(1/2\,\omega_x - 1/2\,b_{g_x}) \\ 1 \end{bmatrix} \tag{5.77}$$

$$R_{19} = \begin{bmatrix} 1/2\,(\Delta T)b \\ -1/2\,(\Delta T)a \\ -1/2\,(\Delta T)d \\ 1/2\,(\Delta T)c \end{bmatrix} \tag{5.78}$$

$$R_{20} = \begin{bmatrix} 1/2\,(\Delta T)c \\ 1/2\,(\Delta T)d \\ -1/2\,(\Delta T)a \\ -1/2\,(\Delta T)b \end{bmatrix} \tag{5.79}$$

$$\mathrm{R}_{21} = \begin{bmatrix} 1/2\,(\Delta T)d \\ -1/2\,(\Delta T)c \\ 1/2\,(\Delta T)b \\ -1/2\,(\Delta T)a \end{bmatrix} \tag{5.80}$$

$$\mathrm{I}^{7x7} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.81}$$

Como estamos trabalhando com dois tipos de sensores (GPS e Bussola) para realizara o ciclo de atualização do EKF e os dados destes sensores podem ser aquisitados em tempos diferentes, há a necessidade de realizar dois ciclos de atualização um para cada sensor.

Para o ciclo de atualização que utiliza os dados do GPS, o vetor e a matriz de medição é dada por:

$$h_{GPS} = \lfloor I^{6x6} | 0^{6x11} \rceil \{x\} = \begin{bmatrix} \lambda \\ \phi \\ h \\ V_N \\ V_E \\ V_D \end{bmatrix} \tag{5.82}$$

$$H_{GPS_k} = \frac{\partial h_{GPS}}{\partial x}\Big|_k = \lfloor I^{6x6} | 0^{6x11} \rceil \tag{5.83}$$

Para o ciclo de atualização que utiliza os dados da bussola, pode-se determinar vetor e a matriz de medição do seguinte modo:

Primeiramente adota-se que o vetor campo magnético da Terra apontará sempre para uma direção e ainda terá uma componente na direção Down do

sistema de coordenadas NED. Deste modo, o vetor campo magnético da Terra no sistema NED pode ser dado por:

$$B = \begin{bmatrix} B_N \\ 0 \\ Bd \end{bmatrix} \tag{5.84}$$

Onde B_N é a componente do campo magnético na direção Norte (North) e B_d é a componente do campo magnético em direção perpendicular ao solo passando pelo centro da Terra (Down).

Após isto, pode-se determinar o vetor campo magnético da Terra realizando uma mudança de base do vetor campo magnético terrestre aquisitado pela bussola no qual se encontra no sistema de coordenadas da plataforma.

$$B = C_p^n \begin{bmatrix} B_x \\ B_y \\ B_z \end{bmatrix} \tag{5.85}$$

Como para o ciclo de atualização do EKF necessita-se de mudar o sistema de coordenadas do sistema NED para o da plataforma, tem-se:

$$\begin{bmatrix} B_x \\ B_y \\ B_z \end{bmatrix} = C_p^{n-1} \begin{bmatrix} B_N \\ 0 \\ Bd \end{bmatrix} \tag{5.86}$$

Deste modo o vetor de medição pode ser dado por:

$$h_{bussola} = C_p^{n-1} \begin{bmatrix} B_N \\ 0 \\ Bd \end{bmatrix} \tag{5.87}$$

A definição da matriz de medição é dada pelo jacobiano do vetor $h_{bussola}$.

$$H_{bussola_k} = \frac{\partial h_{bussola}}{\partial x}\Big|_k = [0^{3x6}|H_1|H_2|H_3|H_4|0^{3x7}] \tag{5.88}$$

Onde:

$$\mathrm{H}_1 = \begin{bmatrix} 2\frac{aB_n}{(a^2+b^2+c^2+d^2)^2} - 4\frac{\left(a^2+b^2-c^2-d^2\right)B_n\,a}{(a^2+b^2+c^2+d^2)^3} - 2\frac{cB_d}{(a^2+b^2+c^2+d^2)^2} - 4\frac{(2\,bd-2\,ac)B_d\,a}{(a^2+b^2+c^2+d^2)^3} \\ -2\frac{dB_n}{(a^2+b^2+c^2+d^2)^2} - 4\frac{(2\,bc-2\,ad)B_n\,a}{(a^2+b^2+c^2+d^2)^3} + 2\frac{bB_d}{(a^2+b^2+c^2+d^2)^2} - 4\frac{(2\,cd+2\,ab)B_d\,a}{(a^2+b^2+c^2+d^2)^3} \\ 2\frac{cB_n}{(a^2+b^2+c^2+d^2)^2} - 4\frac{(2\,bd+2\,ac)B_n\,a}{(a^2+b^2+c^2+d^2)^3} + 2\frac{aB_d}{(a^2+b^2+c^2+d^2)^2} - 4\frac{\left(a^2-b^2-c^2+d^2\right)B_d\,a}{(a^2+b^2+c^2+d^2)^3} \end{bmatrix} \quad (5.89)$$

$$\mathrm{H}_2 = \begin{bmatrix} 2\frac{bB_n}{(a^2+b^2+c^2+d^2)^2} - 4\frac{\left(a^2+b^2-c^2-d^2\right)B_n\,b}{(a^2+b^2-c^2+d^2)^3} + 2\frac{dB_d}{(a^2+b^2+c^2+d^2)^2} - 4\frac{(2\,bd\ \ 2\,ac)B_d\,b}{(a^2+b^2+c^2+d^2)^3} \\ 2\frac{cB_n}{(a^2-b^2+c^2+d^2)^2} - 4\frac{(2\,bc-2\,ad)B_n\,b}{(a^2+b^2-c^2+d^2)^3} + 2\frac{aB_d}{(a^2+b^2+c^2+d^2)^2} - 4\frac{(2\,cd+2\,ab)B_d\,b}{(a^2+b^2+c^2+d^2)^3} \\ 2\frac{dB_n}{(a^2+b^2+c^2+d^2)^2} - 4\frac{(2\,bd+2\,ac)B_n\,b}{(a^2+b^2+c^2+d^2)^3} - 2\frac{bB_d}{(a^2+b^2+c^2+d^2)^2} - 4\frac{\left(a^2-b^2-c^2+d^2\right)B_d\,b}{(a^2+b^2+c^2+d^2)^3} \end{bmatrix} \quad (5.90)$$

$$\mathrm{H}_3 = \begin{bmatrix} -2\frac{cB_n}{(a^2+b^2+c^2+d^2)^2} - 4\frac{\left(a^2+b^2-c^2-d^2\right)B_n\,c}{(a^2+b^2+c^2+d^2)^3} - 2\frac{aB_d}{(a^2+b^2+c^2+d^2)^2} - 4\frac{(2\,bd-2\,ac)B_d\,c}{(a^2+b^2+c^2+d^2)^3} \\ 2\frac{bB_n}{(a^2+b^2+c^2+d^2)^2} - 4\frac{(2\,bc-2\,ad)B_n\,c}{(a^2+b^2+c^2+d^2)^3} + 2\frac{dB_d}{(a^2+b^2+c^2+d^2)^2} - 4\frac{(2\,cd+2\,ab)B_d\,c}{(a^2+b^2+c^2+d^2)^3} \\ 2\frac{aB_n}{(a^2+b^2+c^2+d^2)^2} - 4\frac{(2\,bd+2\,ac)B_n\,c}{(a^2+b^2+c^2+d^2)^3} - 2\frac{cB_d}{(a^2+b^2+c^2+d^2)^2} - 4\frac{\left(a^2-b^2-c^2+d^2\right)B_d\,c}{(a^2+b^2+c^2+d^2)^3} \end{bmatrix} \quad (5.91)$$

$$\mathrm{H}_4 = \begin{bmatrix} -2\frac{dB_n}{(a^2+b^2+c^2+d^2)^2} - 4\frac{\left(a^2+b^2-c^2-d^2\right)B_n\,d}{(a^2+b^2+c^2+d^2)^3} + 2\frac{bB_d}{(a^2+b^2+c^2+d^2)^2} - 4\frac{(2\,bd-2\,ac)B_d\,d}{(a^2+b^2+c^2+d^2)^3} \\ -2\frac{aB_n}{(a^2+b^2+c^2+d^2)^2} - 4\frac{(2\,bc-2\,ad)B_n\,d}{(a^2+b^2+c^2+d^2)^3} + 2\frac{cB_d}{(a^2+b^2+c^2+d^2)^2} - 4\frac{(2\,cd+2\,ab)B_d\,d}{(a^2+b^2+c^2+d^2)^3} \\ 2\frac{bB_n}{(a^2+b^2+c^2+d^2)^2} - 4\frac{(2\,bd+2\,ac)B_n\,d}{(a^2+b^2+c^2+d^2)^3} + 2\frac{dB_d}{(a^2+b^2+c^2+d^2)^2} - 4\frac{\left(a^2-b^2-c^2+d^2\right)B_d\,d}{(a^2+b^2+c^2+d^2)^3} \end{bmatrix} \quad (5.92)$$

A matriz covariância do ruído de medição depende essencialmente dos ruídos sensores que estão sendo utilizados para realizar o ciclo de atualização. Deste modo, ela será uma matriz diagonal, onde o elemento a_{MxM} corresponde o valor do ruído do sensor.

A matriz covariância do ruído para o GPS é dada por:

$$R_{GPS} = \begin{bmatrix} \delta_{\lambda}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \delta_{\emptyset}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \delta_h^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & {\delta_{V_N}}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \delta_{V_E}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \delta_{V_D}^2 \end{bmatrix} \quad (5.93)$$

A matriz covariância do ruído para a bussola é dada por:

$$R_{bussola} = \begin{bmatrix} \delta_{B_x}^2 & 0 & 0 \\ 0 & \delta_{B_y}^2 & 0 \\ 0 & 0 & \delta_{B_z}^2 \end{bmatrix} \quad (5.94)$$

Em geral, o ruído de processo Q é de difícil caracterização, demandando um grande esforço experimental e de modelo. Por este motivo, existem várias técnicas para se chegar a uma estimativa inicial passível ainda de ajustes por tentativa e erro.

Neste trabalho emprega-se um modelo parecido com uma técnica numérica extraída de (Brown, 1997) para estimação de Q. Deste modo, apresentarei a primeiramente a técnica numérica extraída de (Brown, 1997)

Primeiramente forma-se uma matriz bloco, tal que:

$$M = \begin{bmatrix} -F^{(17x17)}(\Delta T) & (C_i W_i {C_i}^T(\Delta T))^{(17x17)} \\ 0^{(17x17)} & F^T(\Delta T)^{(17x17)} \end{bmatrix} \quad (5.95)$$

Onde F é o jacobiano de $f(x, a, \omega)$ em relação ao vetor de estado x equação (5.45), W_i é a densidade espectral de potência do ruído associados ao vetor u, no caso as leituras dos acelerômetros e giroscópio da IMU, C_i é o jacobiano de $f(x, a, \omega)$ em relação ao vetor $u = [a, \omega]$, sendo $a = [a_x \ a_y \ a_z]$ (vetor aceleração no sistema de coordenadas da plataforma) e $\omega = [\omega_x \ \omega_y \ \omega_z]$ (vetor taxa de rotação no sistema de coordenadas da plataforma).

Admite-se inicialmente que este ruído seja normalmente distribuído com média zero e covariância igual a dos respectivos sensores.

$$W = \begin{bmatrix} \delta_{a_x}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \delta_{a_y}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \delta_{a_z}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \delta_{\omega_x}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \delta_{\omega_y}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \delta_{\omega_z}^2 \end{bmatrix} \quad (5.96)$$

A operação e^M produz uma nova matriz bloco, cujos elementos são:

$$e^M = \begin{bmatrix} \ldots^{(17x17)} & (A^{-1}Q)^{(17x17)} \\ 0^{(17x17)} & A^{T(17x17)} \end{bmatrix} \quad (5.97)$$

Portanto, Q é obtido transponde-se A^Tpara obter A e em seguida multiplica-se este resultado por $A^{-1}Q$. Os resultados utilizando estão apresentados abaixo:

Tabela 1 – Comparação dos erros variando a amplitude de C_i

Estados	C_i	$10C_i$	$25C_i$	$50C_i$	$75C_i$	$100C_i$
	Erro médio	Erro médio	Erro médio	Erro médio	Erro médio	Erro médio
$\lambda * R$ (m)	0.7850	0.7010	0.7429	0.6968	0.6855	0.6928
$\emptyset * R$ (m)	4.5290	4.5257	4.4390	4.3936	4.3115	4.1839
h (m)	0.1795	0.1491	0.1991	0.1911	0.1319	0.1329
V_N (m/s)	0.0124	0.0146	0.0135	0.0152	0.0156	0.00181
V_E (m/s)	0.0110	0.0146	0.0181	0.0205	0.0229	0.0231
V_D (m/s)	0.0696	0.0146	0.0143	0.118	0.0135	0.0171
a	3.31e-4	0.0012	0.0018	0.0019	0.0021	0.0023
b	9.98e-4	0.0016	0.0022	0.0027	0.0034	0.0038
c	9.99e-4	0.0016	0.0029	0.0040	0.0053	0.0057
d	0.0032	0.0037	0.0022	0.0038	0.0036	0.0032
Roll º	0.1249	0.1725	0.2838	0.3702	0.4593	0.4923
Pitchº	0.0938	0.1837	0.2970	0.4108	0.5254	0.5779
Yawº	0.3989	0.6135	0.218	0.2457	0.6052	0.2867

Para realizar esta simulação acima foram desconsiderados o erro de bias da IMU e a aceleração de Coriolis.

Agora apresentarei o modelo do ruído de processo *Q* que está sendo empregado para este trabalho. O modelo que propus basicamente se deferência na geração do bloco M, sendo o restante do procedimento idêntico ao extraído de (Brown, 1997). O bloco M proposto é:

$$M = \begin{bmatrix} -F^{(17x17)}(\Delta T) & (CWC^T)^{(17x17)} \\ 0^{(17x17)} & F^T(\Delta T)^{(17x17)} \end{bmatrix} \tag{5.98}$$

Onde F é o jacobiano de $f(x, a, \omega)$ em relação ao vetor de estado x, W é a densidade espectral de potência do ruído associado ao vetor $j = [x|u]$, sendo x o vetor dos estados e $u = [a, \omega]$ o vetor de leituras da IMU, o vetor de C é o jacobiano de $X = x + (\Delta T) f(x, a, \omega)$ em relação ao vetor j. Foi escolhida esta função X, pois ela é a função de predição dos estados do EKF, sendo assim quero estimar o erro do ruído de processo gerado por ela.

Como e^M é aproximado por:

$$e^M = \left[I + \frac{\partial X(x,t)}{\partial j} \right] \tag{5.99}$$

Após está operação, canto superior direito do Bloco de e^M é:

$$(A^{-1}Q)^{(17x17)} = (CWC^T)^{(17x17)} \tag{5.100}$$

Sendo assim, ruído de processo *Q é:*

$$(Q)^{(17x17)} = (ACWC^T)^{(17x17)} \tag{5.101}$$

Como para as várias simulações realizadas, a matriz *A* era muito próxima da matriz identidade, resolveu-se não realizar a multiplicação dela com a matriz $(CWC^T)^{(17x17)}$, para minimizar os cálculos e serem realizado e diminuir o tempo de processo do EKF. Deste modo, pode-se calcular o ruído de processo *Q* por:

$$(Q)^{(17x17)} = (CWC^T)^{(17x17)} \tag{5.102}$$

Onde as matrizes *C* e *W* são:

$$W=\begin{bmatrix}
\delta_\lambda^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \delta_\varphi^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \delta_h^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \delta_{V_x}^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \delta_{V_y}^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \delta_{V_z}^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \delta_a^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta_b^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta_c^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta_d^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta_g^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta_{b_{ox}}^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta_{b_{oy}}^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta_{b_{oz}}^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta_{b_{gx}}^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta_{b_{gy}}^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta_{b_{gz}}^2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta_{a_x}^2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta_{a_y}^2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta_{a_z}^2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta_{W_x}^2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta_{W_y}^2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta_{W_z}^2
\end{bmatrix} \tag{5.103}$$

(5.104)

Onde e são:

$$C_1 = \begin{bmatrix}
(\Delta T)\,(a^2+b^2-c^2-d^2) & (\Delta T)\,(2\,bc-2\,ad) & (\Delta T)\,(2\,bd+2\,ac) \\
(\Delta T)\,(2\,bc+2\,ad) & (\Delta T)\,(a^2-b^2+c^2-d^2) & (\Delta T)\,(2\,cd-2\,ab) \\
(\Delta T)\,(2\,bd-2\,ac) & (\Delta T)\,(2\,cd+2\,ab) & (\Delta T)\,(a^2-b^2-c^2+d^2)
\end{bmatrix} \tag{5.105}$$

$$C_2 = \begin{bmatrix}
-1/2\,(\Delta T)b & -1/2\,(\Delta T)d & -1/2\,(\Delta T)c \\
1/2\,(\Delta T)a & 1/2\,(\Delta T)c & -1/2\,(\Delta T)d \\
1/2\,(\Delta T)d & -1/2\,(\Delta T)b & 1/2\,(\Delta T)a \\
-1/2\,(\Delta T)c & 1/2\,(\Delta T)a & 1/2\,(\Delta T)b
\end{bmatrix} \tag{5.106}$$

5.2.5. Resultados das simulações Realizadas para o EKF

Para realizar a simulação utilizou-se o novo modelo para o EKF, um arquivo.m gerado no Matlab e a matriz Apoena, que contém os dados da IMU, GPS e Bússola, ambos simulados, além disto, foram desconsiderados o erro de bias da IMU e a aceleração de Coriolis . Os resultados obtidos com esta simulação podem ser observados abaixo:

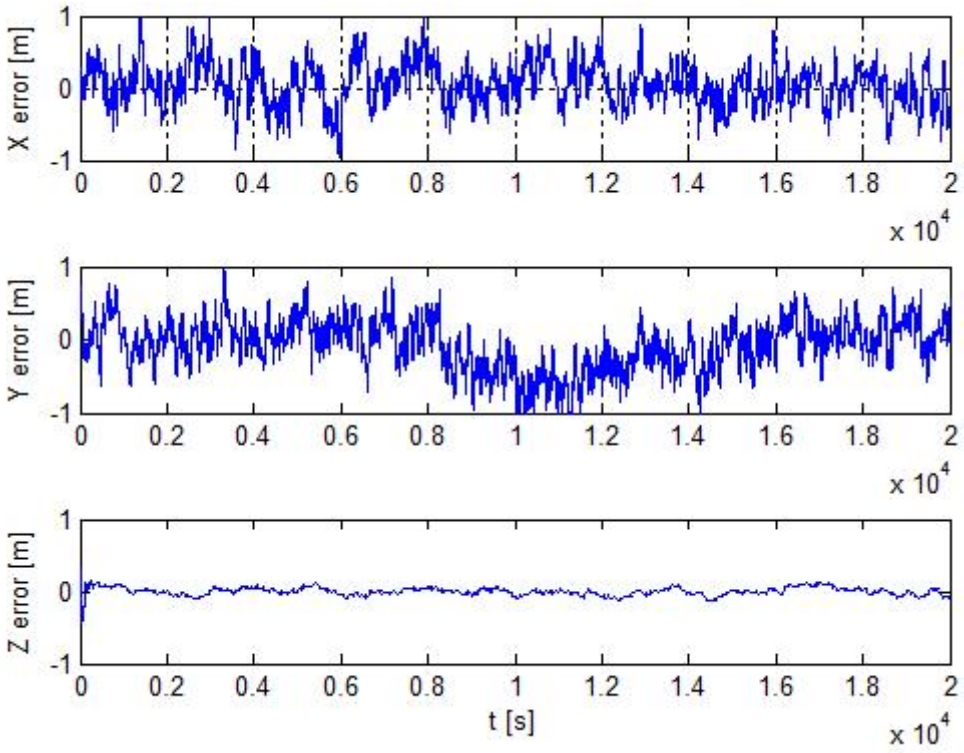

Figura 5.23 – Erro de posição

Pode-se perceber que: o erro de posição está em um patamar aceitável, sendo o erro máximo inferior a um metrô, o erro médio na direção $e_x = 0.2171m$, o erro médio na direção $e_y = 0.2102m$ e o erro médio na direção $e_z = 0.0517m$.

Para obter o erro de posição em metros, foi realizada a conversão dos parâmetros de latitude e longitude para os seus respectivos comprimento de arcos.

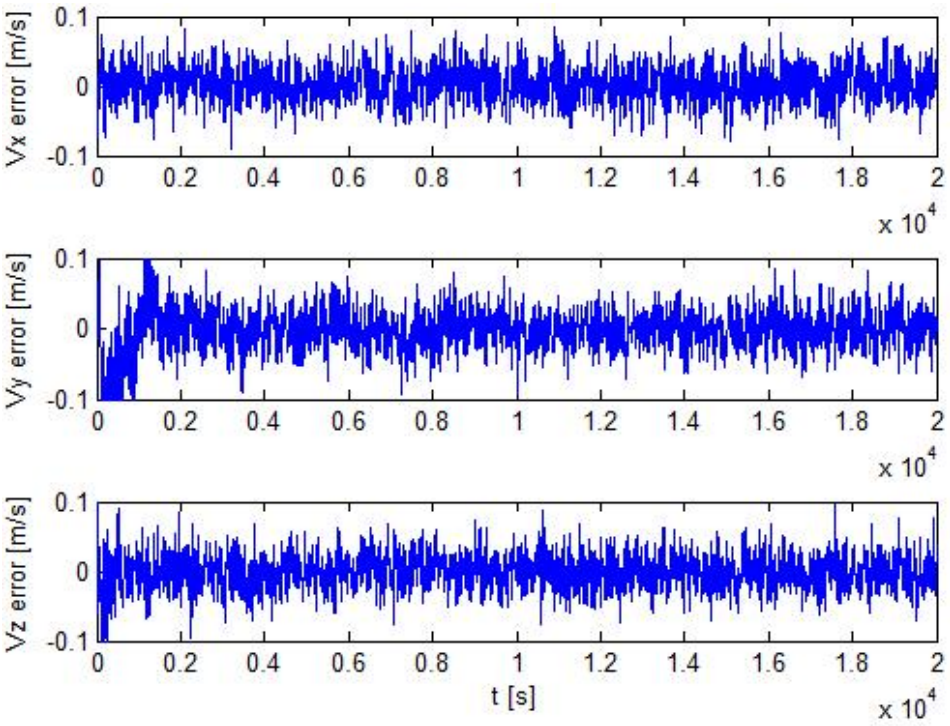

Figura 5.24 – Erro de Velocidade

Pode-se perceber que: o erro de velocidade está em um patamar aceitável, sendo o erro máximo inferior que 0.1m/s, o erro médio na direção $e_{Vx} = 0.0213m/s$, o erro médio na direção $e_{Vy} = 0.0201m/s$ e o erro médio na direção $e_{Vz} = 0.0199m/s$.

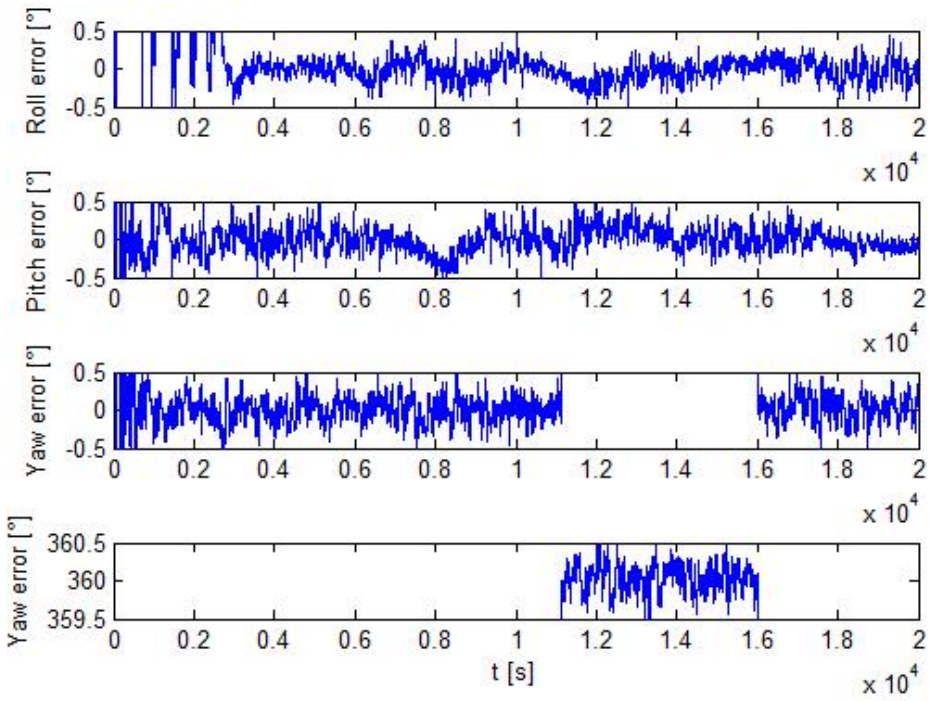

Figura 5.25 – Erro de Atitude

Pode-se perceber que: o erro de atitude está em um patamar aceitável, sendo o erro máximo inferior que 0.5º, o erro médio de roll $e_{roll} = 0.1253º$, o erro médio de pitch $e_{pitch} = 0.1054º$ e o erro médio de yaw $e_{yaw} = 0.1459º$.

Para obter o erro de atitude em graus, foi realizada a conversão dos parâmetros do quaternions para os seus respectivos ângulos de Euler e como esta conversão utiliza arcoseno e arcotangente, pode-se explicar o motivo do ângulo de yaw ter alterado bruscamente de 0º para 360º.

5.3. Desenvolvimento do Programa em Tempo Real

Para desenvolver o programa em tempo real está sendo utilizado à linguagem C++, sendo que, o programa de desenvolvimento do software é o Rhapsody, pois já existe um projeto em andamento do UAV, que está sendo desenvolvido pela empresa XMobots, onde a base de desenvolvimento do software do UAV é o Rhapsody.

Deste modo, para que o desenvolvimento do software do Filtro de Kalman em Tempo Real seja compatível com este projeto, adotou-se o Rhapsody. Além disto, este tem a vantagem de interagir com o Matlab onde foi desenvolvido o simulador do filtro, e ainda de interagir com o Momentics QNX no qual embarca e monitora o funcionamento do software durante o processo.

Devido à dificuldade de adquirir a licença do Rhapsody para que este interaja com o Simulink e também a dificuldade de realizar a “Linkagem” dos arquivos gerados em C++ pelo Real-Time Workshop, optou-se por gerar manualmente o código em C++ que implementa do EKF, sendo a única forma utilizada para automatizar a geração do código em C++ foi com referência as matrizes e vetores que definem os parâmetros do filtro.

Para gerar as matrizes que definem os parâmetros para o EKF foi utilizado à manipulação simbólica do Matlab, deste modo, é possível converter estas matrizes em código C utilizando a função “ccode(nome_da_matriz)”, facilitando e diminuindo a porcentagem de erros ao gerá-las. Como foi observado acima, estas matrizes são muito extensa e se fosse construída manualmente possivelmente seria uma fonte de erros para implementação do EKF.

Assim, para implementar o EKF em Tempo Real, bastou-se desenvolver algoritmos de manipulação matemática de matrizes e vetores.

Os algoritmos para manipulação de matrizes foram: Soma entre duas matrizes, Subtração entre duas matrizes, Multiplicação entre duas matrizes, Transposição de uma matriz, Multiplicação de uma matriz por um ganho, Multiplicação de uma matriz por um vetor e Inversa de uma matriz.

O algoritmo mais complicado de se tratar foi à inversão de matrizes, pois dependendo de como este foi implementado, pode-se gerar grandes distúrbios no EKF, devido a erros de precisão, truncamento e inversão de matrizes com determinantes nulos. O algoritmo implementado para realizar a inversão neste trabalho foi o Gauss Jordan, devido sua facilidade de implementação (com relação aos outros métodos) e também da possibilidade de realizar a inversão de matrizes de qualquer tamanho com uma boa precisão.

Os algoritmos implementados para manipulação de vetores foram: soma entre dois vetores e subtração entre dois vetores.

Por fim, implementou-se o algoritmo do EKF. Para gerá-lo em C++ utilizou-se das funções e métodos descritos acima, além de gerar mais dois métodos um para a predição do filtro e outro para atualização do filtro.

Como o meu trabalho consistia em implemetar o Filtro de Kalman em Tempo Real, houve a necessidade de realizar a aquisição dos sensores (Bússola, GPS, IMU).

Para isto, utilizou um software desenvolvido pelo aluno de mestrado Gionvani Aminati, onde este foi estruturado e gerado no programa Rhapsody. Este software tem a função de fornecer ao filtro os seguintes parâmetros: valor da variável de entrada do filtro (ex: acelerações, taxa de rotação, campo magnético, e a posição fornecida pelo GPS) e uma referência desta variável para indicar se ela foi atualizada ou não, para não entrar com valores errados no filtro, o que poderia ser um grande problema para a precisão e desempenho do filtro.

Como houve a necessidade de realizar a integração entre o software desenvolvido por mim e pelo Amianti surgiram várias complicações antes que esta integração funciona-se.

Esta fase de integração foi a mais complexa e a mais trabalhosa, pois surgiram vários erros de compatibilidade e também de estruturação dos

diagramas em UML, no Rhapsody, e devido a isto, muitas vezes o sistema operacional abortava o programa que foi embarcado.

Este problema foi solucionado ao se descobrir que o software estava tentando alocar valores nas matrizes antes inicializá-las. Este efeito ocorria devido ao modelo estruturado em UML. Para resolver este problema criou-se uma porta para passar os parâmetros aquisitados pelos sensores.

A partir deste ponto, o sistema começou a funcionar sem que o QNX aborta-se o software que era embarcado nele.

Mas perceberam-se outros problemas, pois, o Filtro de Kalman não estava se estabilizado em uma posição e sim ele ficava oscilando em torno de um ponto. Para solucionar esta oscilação, bastou-se fazer a sintonização do EKF alterando o módulo dos erros estimados para os estados, que se encontra na matriz Q e na matriz inicial de covariância dos Estados (matriz P inicial).

Após realizar a sintonização do filtro, ele se estabilizou e passou funcionar corretamente, apesar do surgimento de um off-set no picht. Este off-set surgiu devido a modelagem inicial adotada para o campo magnético terrestre. A principio havia adotado que o campo magnético apontava sempre para o Norte sem componentes na direção perpendicular a crosta terrestre, o que ao fazer os testes com a bussola se descobriu que este procedimento não é verídico, sendo que o campo magnético da Terra para este ano e para a latitude e longitude da cidade universitária da USP é inclinado aproximadamente de 30 graus.

Para corrigir este problema foi refeito o vetor de medição hcompass e a matriz de medição Hcompass, para poder inserirem neles o campo magnético perpendicular a crosta terrestre.

A partir deste ponto o filtro passou a funcionar regularmente e se estabilizou próximo a uma posição prevista, ou seja, sem nenhum off-set de grandes proporções.

6. Testes Experimentais em Tempo Real

6.1. Ensaio em Tempo Real para o Sistema Fixo em uma Posição

Este ensaio foi realizado para validar o Filtro de Kalman implementado em Tempo Real, além de verificar o seu desempenho com relação ao funcionamento em Tempo Real e o nível de processamento requerido pelo filtro. A bancada de testes e os instrumentos utilizados para realiza este ensaio podem ser observados na figura abaixo.

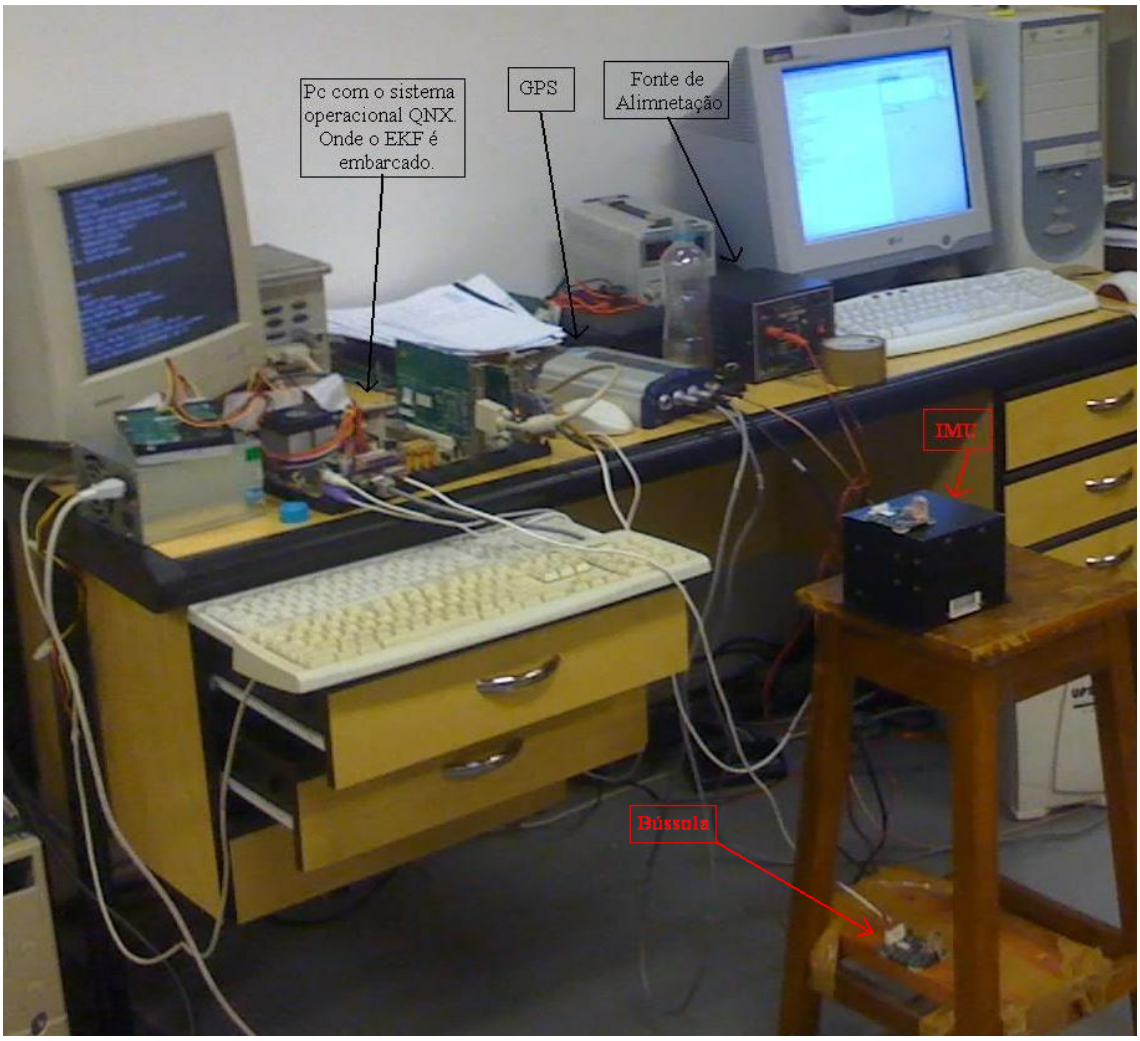

Figura 5.26 – Bancada do Ensaio

O primeiro ensaio realizado foi para a validação do filtro com o sistema estático. Neste ensaio deixaram-se os sensores em uma determinada posição fixa no espaço e se verificou a estabilidade do sistema utilizando uma interface gráfica parecida com o cockpit de um avião, esta interface foi desenvolvida pela

empresa Xmobots, no qual foi cedida para os experimentos, ela pode ser visualizada na figura 5.27. Os resultados deste ensaio foram satisfatórios, pois o filtro se estabilizou rapidamente em torno da posição espera, tendo uma oscilação menor que 2 graus do roll, pitch e yaw e uma oscilação menor que um metro na latitude e longitude e um erro médio menor que 1.7 metros para a altitude, enquanto que o GPS fornecia um erro de aproximadamente 2.3 metros para a latitude e longitude e um erro de 3.74 metros para a altitude.

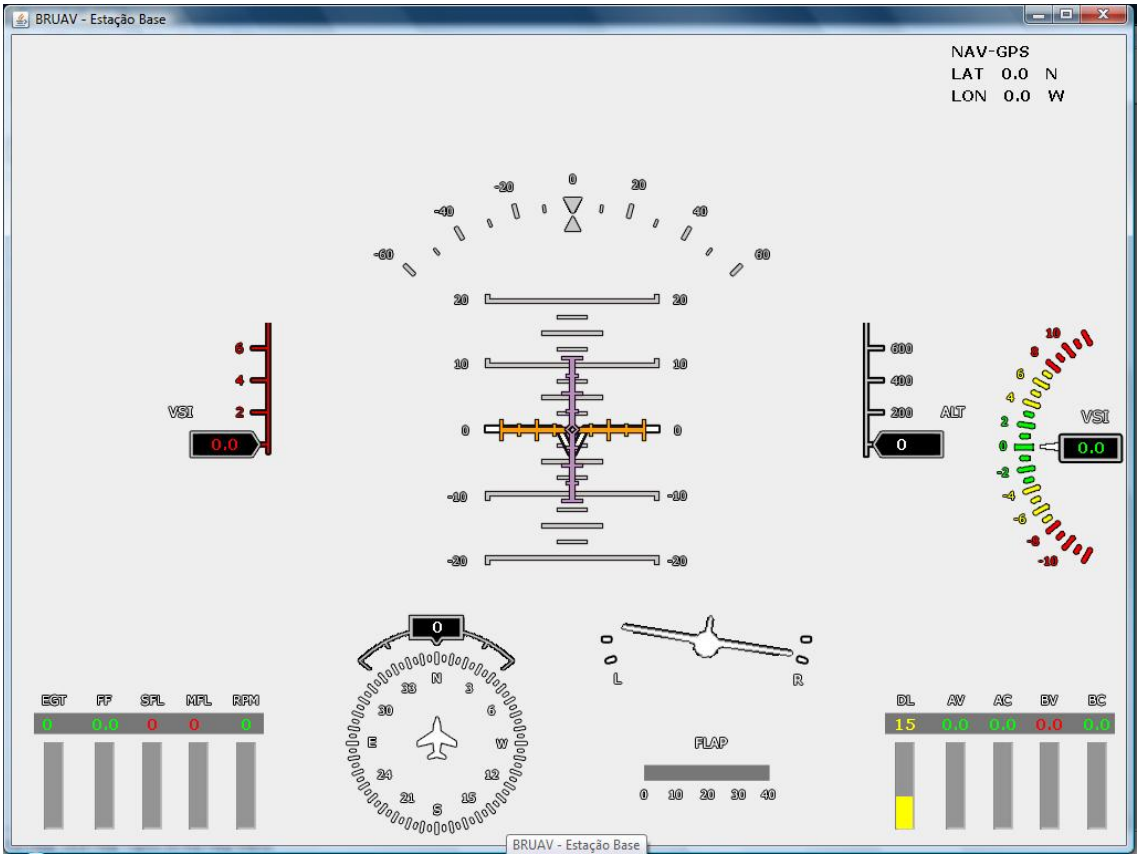

Figura 5.27 – Cockpit fornecido pela Xmobots

Os resultados para o caso totalmente estático pode ser observado abaixo:

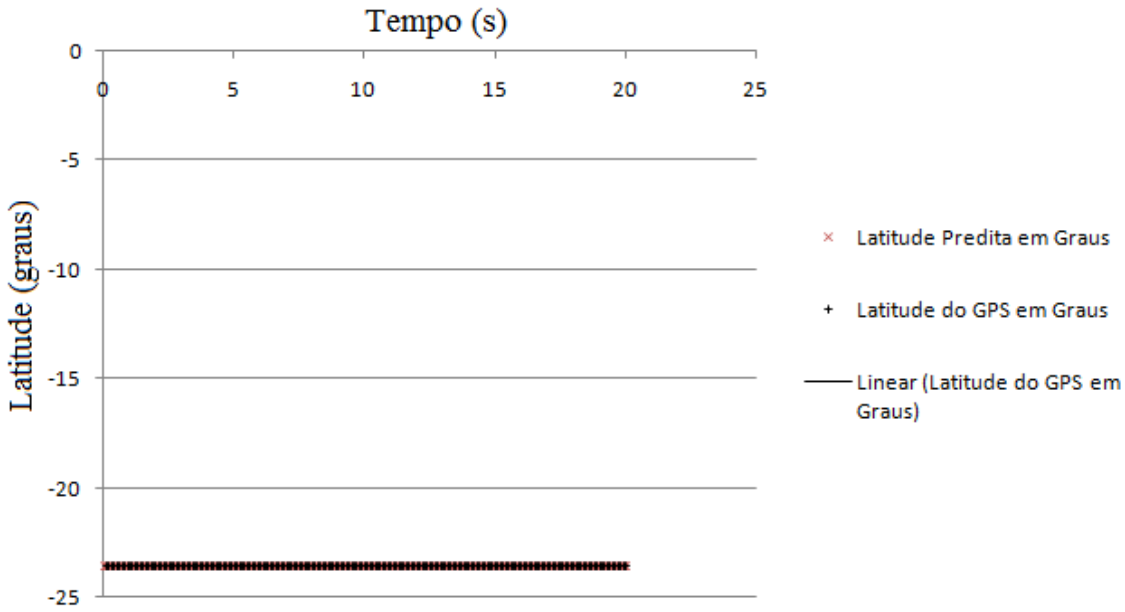

Figura 5.28 – Resultado da Latitude para a execução do EKF em “Tempo Real”

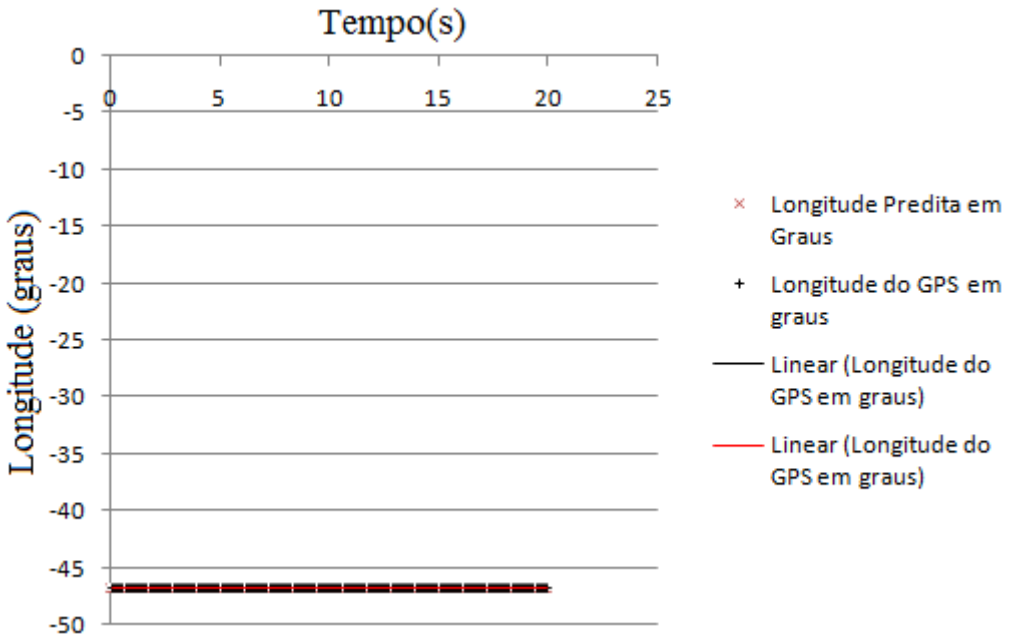

Figura 5.29 – Resultado da Longitude para a execução do EKF em “Tempo Real”

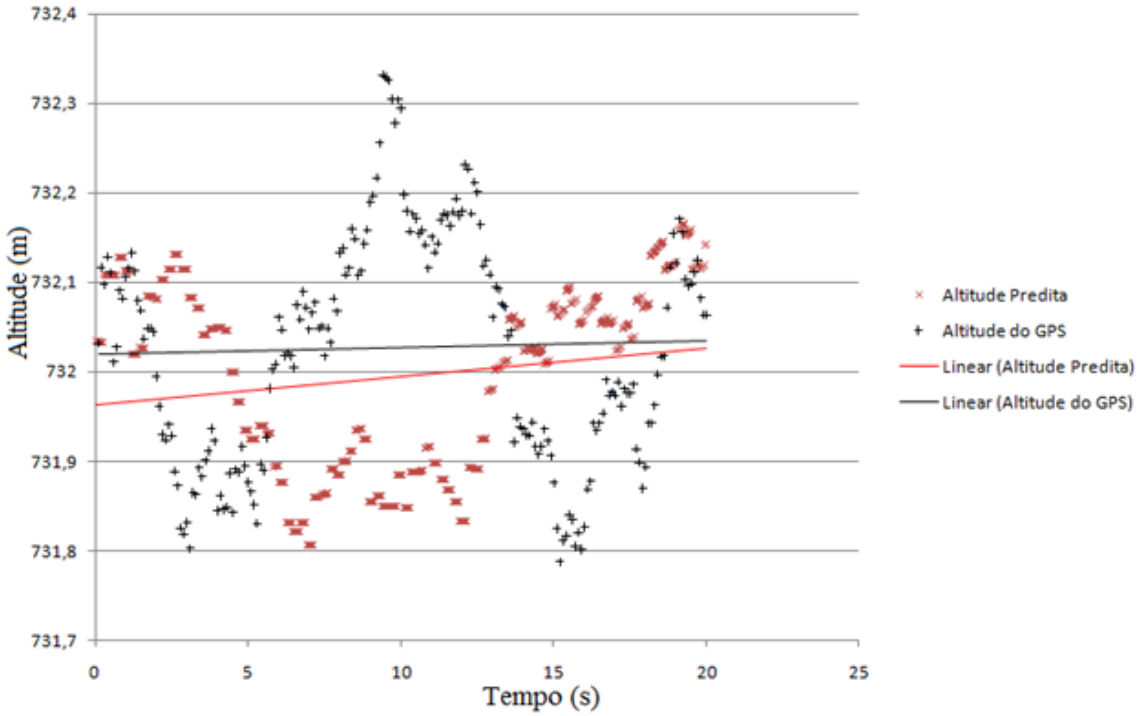

Figura 5.30 – Resultado da Altitude para a execução do EKF em “Tempo Real”

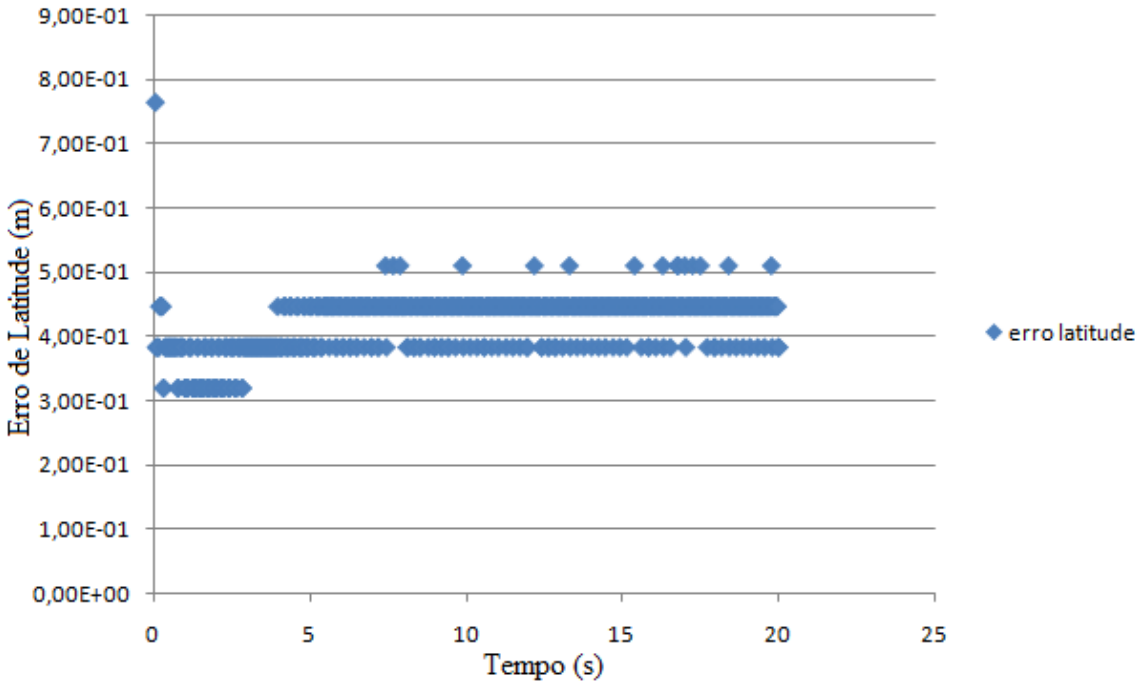

Figura 5.31 – Erro de Latitude para a execução do EKF em “Tempo Real”

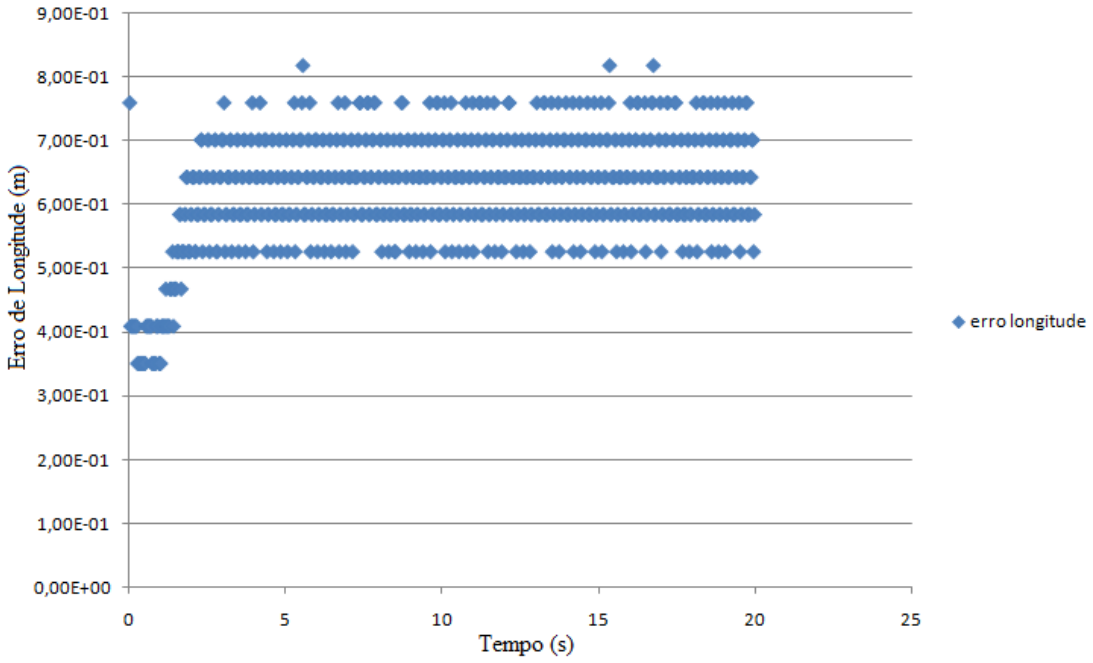

Figura 5.32 – Erro deLongitude para a execução do EKF em “Tempo Real”

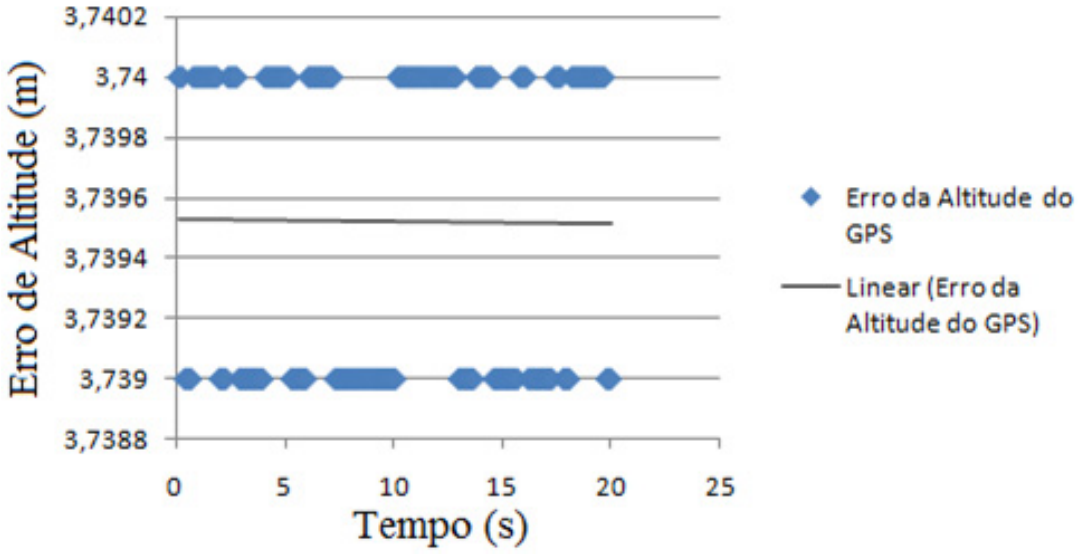

Figura 5.33 – Erro de Altitude do GPS

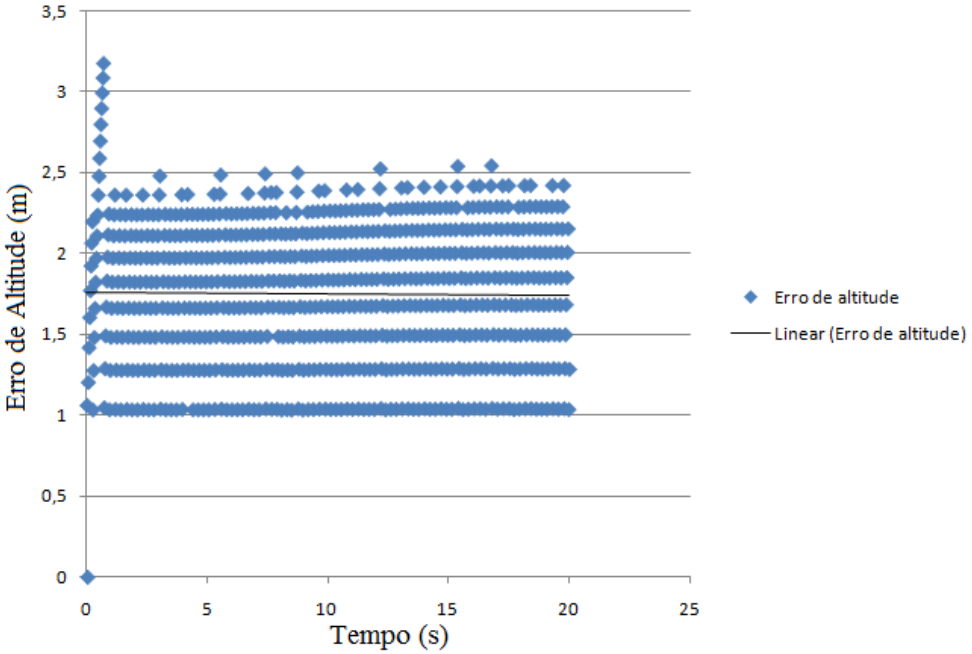

Figura 5.34 – Erro de Altitude para a execução do EKF em “Tempo Real”

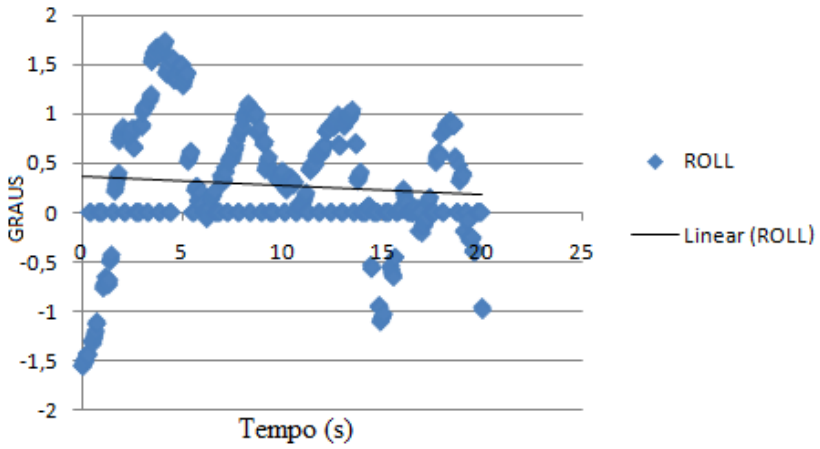

Figura 5.35 – Resultado da Atitude Roll para a execução do EKF em “Tempo Real”

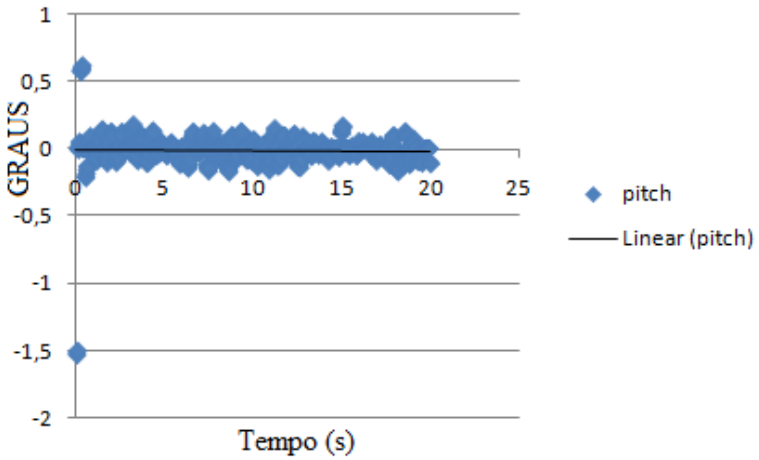

Figura 5.36– Resultado da Atitude Pitch para a execução do EKF em “Tempo Real”

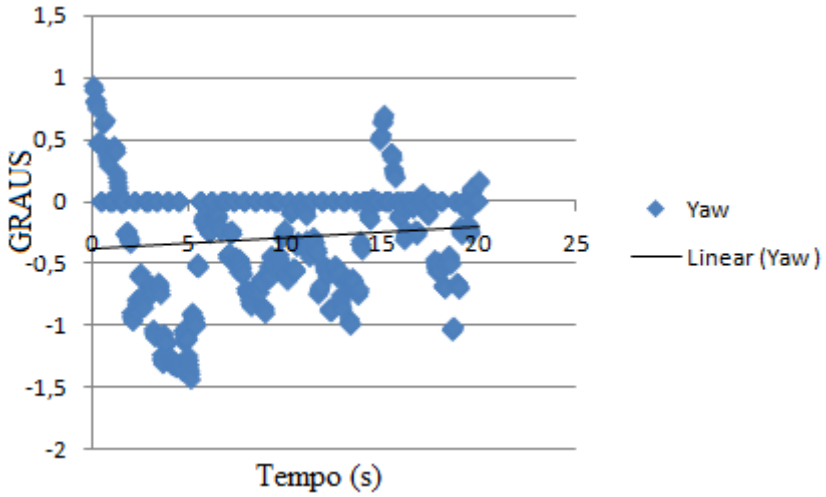

Figura 5.37 – Resultado da Atitude Yaw para a execução do EKF em “Tempo Real”

O segundo ensaio realizado foi para verificar se o sistema estava sendo executado em Tempo Real e verificar o nível de processamento utilizado. Neste ponto de vista o filtro pode-se validar o seu funcionamento em Tempo Real além de constatar que o nível de processamento requerido para ele foi muito abaixo do que era esperado, sendo este um ponto positivo do filtro, pois, para executá-lo não há necessidade de um computador com alto desempenho de processamento. O nível de processamento utilizado pelo filtro foi de aproximadamente 10% de um processado de 1.2GHz, ou seja, o filtro utiliza cerca de 120Mhz do processado. Para comprovar seu funcionamento On-line e em “Tempo Real” pode-se observar as figuras abaixo:

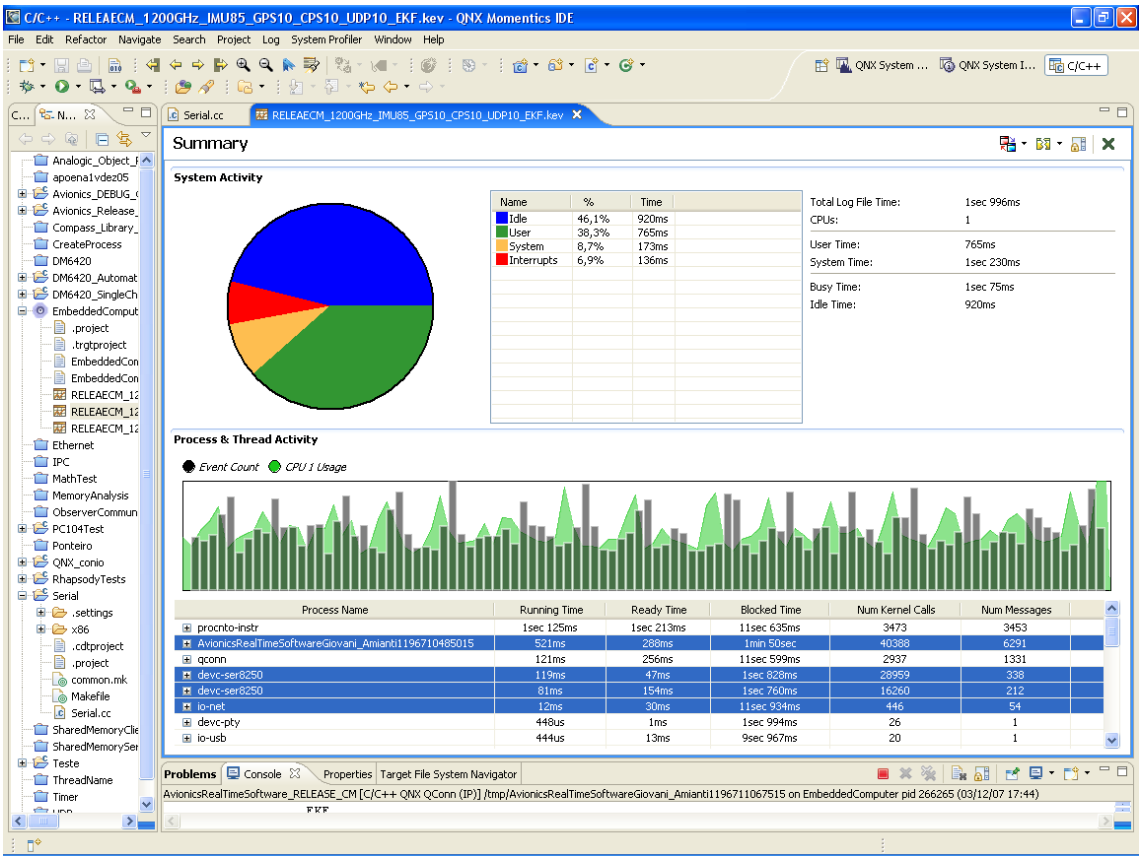

Figura 5.38 – Verificação do Nível de processamento utilizando no EKF durante a sua execução em “Tempo Real”

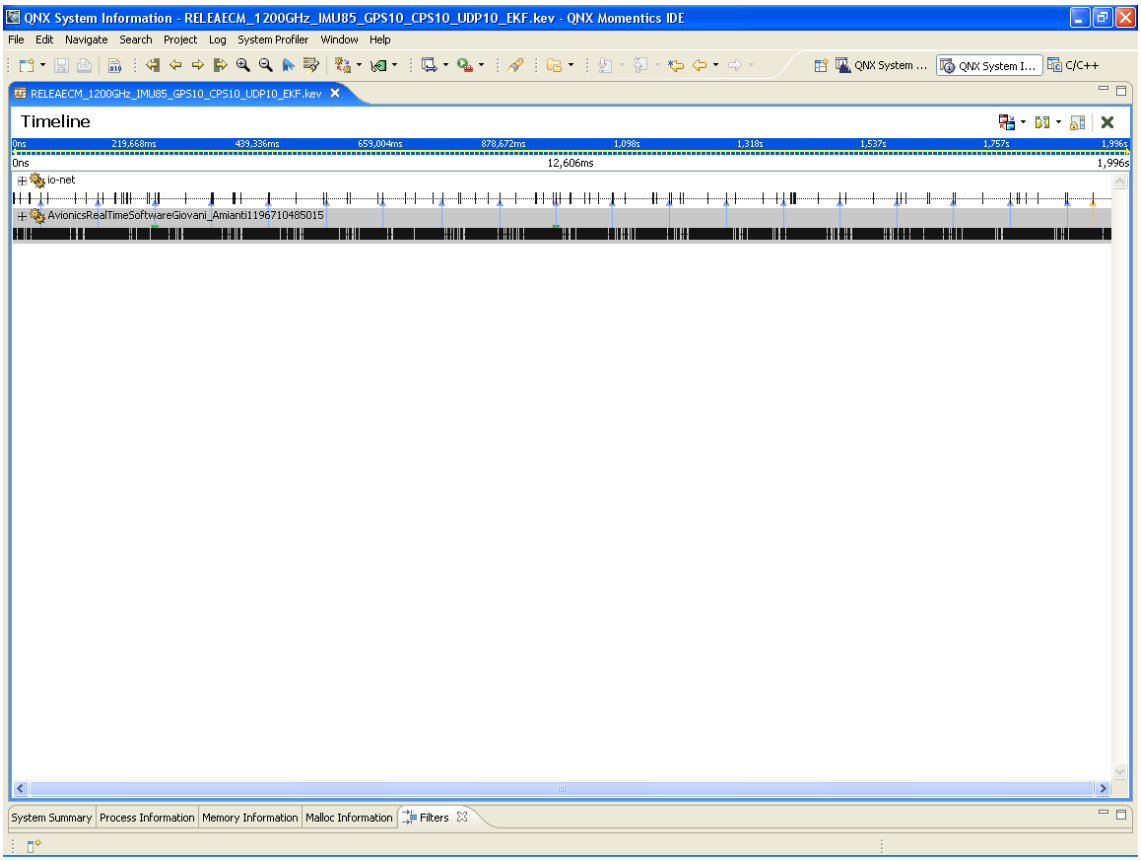

Figura 5.39 – Verificação das chamadas das Threads e verificação do tempo utilizado para sua execução em “Tempo Real”

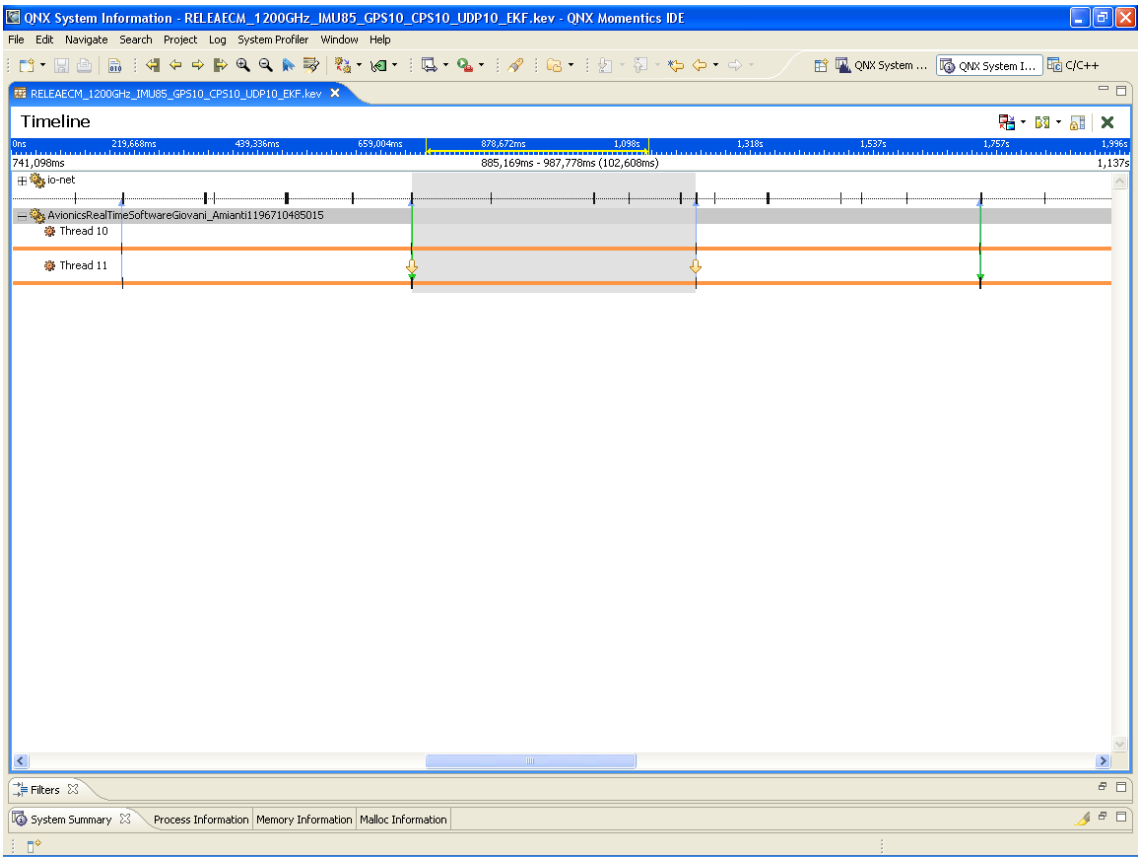

Figura 5.40 – Verificação do tempo utilizando para a execução do EKF em “Tempo Real”

O terceiro ensaio realizado foi para verificar o desempenho do filtro caso a bussola pare de funcionar. Neste experimento o filtro teve um resultado regular, pois consegui manter estável, mas com uma oscilação um pouco maior do que quando a bússola estava ligada.

O quarto ensaio realizado foi para o caso em que o GPS pare de funcionar. Neste experimento o filtro também teve um resultado regular apesar da posição começou a divergir lentamente da posição esperada.

O ultimo ensaio realizado nesta bancada foi considerar a posição fixa e variar somente a atitude do sistema. Para este ensaio o filtro teve um bom desempenho. Fornecendo os valores de atitude esperados, sem grandes oscilações da posição, e da atitude.

6.2. Ensaio em Tempo Real para um Sistema Dinâmico

O ensaio do EKF sendo executado em um sistema dinâmico foi realizado para validá-lo dinamicamente. Este ensaio foi realizado embarcando os equipamentos em um carro e andando com o carro pela Cidade Universitária da USP. Os equipamentos embarcados no carro podem ser observados na figura abaixo.

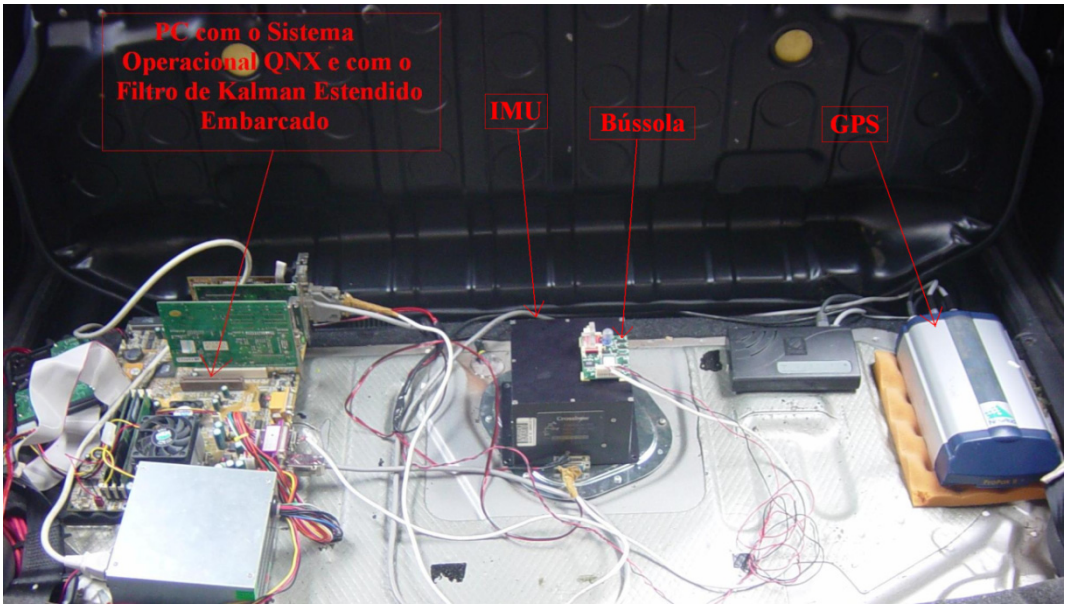

Figura 5.41 – Equipamento Embarcado no Carro

Figura 5.42 – Fixação da Antena do GPS no Carro

Para começar este ensaio alinhava-se o carro com o norte magnético da terra, pois, é a condição de inicialização do filtro para poder obter os valores do campo magnético na direção norte e na direção perpendicular a corta terrestre podendo assim zerar a componente magnética na direção Y da Bussola (nesta situação, esta direção apontava para o leste terrestre, onde não há nenhum campo magnético apontando para esta direção), após isto se aciona o filtro.

Quando o filtro se estabilizava colocava-se o carro em movimento. No percurso escolhido para o ensaio havia um local com uma reta de comprimento de aproximadamente de 4 metros e após isto uma curva acentuada para a esquerda e a partir deste momento o carro circulava em torno de uma trajetória circular.

Os resultados do ensaio dinâmico não foram satisfatórios, pois, a partir da curva acentuada para a esquerda a atitude fornecida pelo filtro entrou em colapso não conseguindo mais se estabilizar, mas na trajetória quase que retilínea no começo teve um resultado relativamente satisfatório.

Um dos possíveis motivos para este colapso do filtro pode estar correlacionado com a sintonia delo, pois, foi realizada a sua sintonia para o caso estático, se pensado que seria o mesmo para o caso dinâmico, mas isto não pode ser observado.

Deste modo, para que o filtro funcione corretamente a necessidade de se realizar mais ensaios dinâmicos para tentar realizar uma estimativa melhor para os valores estimados dos erros dos estados que se encontram na matriz Q.

7. Conclusão

Neste trabalho foram desenvolvidos dois modelos de navegação para realizar a fusão sensorial dos dados aquisitados pelo GPS, Bussola e IMU. Sendo que o primeiro é um modelo bem simplificado e distante da realidade, onde foi utilizado o Filtro de Kalman Discreto.

Já o segundo é um modelo mais próximo da realidade e utiliza o Filtro de Kalman Estendido para realizar a fusão e também para melhorar a estimativa da posição, velocidade e atitude da plataforma, quando este estiver funcionando com os sensores reais e não simulados.

Visto que na navegação inercial, tem-se que realizar uma série de correções e transformações de coordenadas antes de estimarmos a posição do veículo, a modelagem adequada do problema no espaço de estados pode levar ao sucesso da aplicação do filtro, como pode ser observado na simulação realizado no Matlab do filtro proposto neste trabalho.

Ao observar os resultados simulados do Filtro de Kalman, para ambos os modelos, pode-se concluir que ele realiza uma ótima estimativa, pois ele conseguiu minimizar significativamente o erro entre o valor estimado e o real.

Mas ao utilizá-lo tem-se que tomar cuidado, pois se devem levar em consideração os parâmetros da matriz covariância do estado inicial, além do período de aquisição dos sensores, pois se eles forem muito elevados, principalmente a IMU, o filtro pode não convergir.

Quanto ao sistema de tempo real, a principal idéia que se deve levar em conta é: o sistema de processamento de informação que tem que responder aos estímulos externos assíncronos e síncronos, sendo o tempo requerido para isto tem que ser finito e estar dentro do prazo definido anteriormente para o sistema.

Como já foi descrito no relatório ao implementar o EKF em C++ para poder embarcá-lo em um sistema que opera em Tempo Real ocorreu vários problemas de integração entre o software do EKF e do software de aquisição de dados. Para tentar solucionar isto futuramente pode-se tentar conseguir a licença do Rhapsody para que se possa utilizar o bloco que realiza a integração entre ele o Matllab/Simulink podendo assim utilizar o modelo desenvolvido no Simulink.

Outra importante observação que se pode fazer com relação ao filtro implementado em Tempo Real pode-se dizer que ao se trabalhar com ele fixo em uma posição (na bancada) ele tem um desempenho satisfatório sendo o seu erro de atitude menor que 2º e erro de posição menor que um metro para a latitude e Longitude e para o erro de altitude um erro médio menor que 1.7 metros.

Mas ao embarcado em um sistema dinâmico ele não atendeu os requisitos de navegação, pois, os resultados do filtro não condiziam mais com os resultados esperados, sendo um dos motivos suspeitos para esta causa pode ser a má sintonização do filtro, deste modo, para que se possa utilizando-lo em um veiculo móvel há a necessidade de realizar mais ensaio pra que se possam determinar os erros estimados dos estados que pertence a matriz de ruído de processo Q.

Os principais ganhos deste trabalho para o Sistema de Navegação de veículos são: o método de desenvolvimento da matriz de covariância do processo Q e também ao grande avanço ao implementá-lo em “Tempo Real”, pois até então, na Escola Politécnica da USP ninguém o havia implementado em “Tempo Real”.

A partir das analises de sua execução em “Tempo Real" pode-se concluir que ele possui um desempenho muito melhor do que era esperado com relação ao nível de utilização do processamento, sendo que este exige um processado menor que 120Mhz.

8. Bibliografia

[1] Amianti,G., "**MEMORIAL DO PROJETO BR-UAV AERONAVE APOENA I**". 131p. – Artigo Interno, Escola Politécnica, Universidade de São Paulo. São Paulo, SP, 2007

[2] Douglass, B. e Evangelist, C., "**Capturing Requirements for Real-Time and Embedded System**". Disponível em < http://www.ilogix.com >, Data de acesso: 15/04/2007

[3] Santana, Douglas, S,." **Estimação de trajetórias terrestres utilizando unidade de medição inercial de baixo custo e fusão sensorial**", Dissertação de Mestrado, Escola Politécnica, Universidade de São Paulo. São Paulo, SP. 2005

[4] Brown, R., "**Introduction to random signals and applied Kalman filtering: with MATLAB exercises and solutions**", John Wiley & Sons, 1997.

[5] Amianti,G., "**Relatório Parcial de Projeto Aviônica – Hardware/software - Sistema de Observação**". 63p. – Artigo Interno, Escola Politécnica, Universidade de São Paulo. São Paulo, SP, 2006

[6] Jang, J. S., LICCARDO, D., "**Automation of Small UAVS Using a Low Cost Mems Sensor and Embedded Computing Platform**". Disponível em : < http://www.xbow.com>, Data de Acesso: 20/08/2007

[7] Titterton, D., Weston, J., "**Strapdown Inertial Navigation Technology**". Editora Institution of Engineering and Technology, 2nd ED, 2004.

Anexo

A. Implementação do Filtro de Kalman no MATLAB

```
%%%%Observador de estados pelo filtro de Kalman (independente da
dinâmica
%%%%do veículo)
clc
clear all
load Apoena1_6DOFsim %matriz de dados adquiridos

T=0.01;       %Período da propagação (período da IMU)
T_GPS=1;      %Período de amostragem do GPS

%Matriz de transiçao de estados
A=[ 1 0 0 T 0 0 0 0 0       %Xe
    0 1 0 0 T 0 0 0 0       %Ye
    0 0 1 0 0 T 0 0 0       %Ze
    0 0 0 1 0 0 0 0 0       %Vxe
    0 0 0 0 1 0 0 0 0       %Vye
    0 0 0 0 0 1 0 0 0       %Vze
    0 0 0 0 0 0 1 0 0       %roll
    0 0 0 0 0 0 0 1 0       %pitch
    0 0 0 0 0 0 0 0 1];     %yaw

%Matriz de entradas de controle (mediçao da IMU)
B=[ 0.5*T^2     0           0           0   0   0
    0           0.5*T^2     0           0   0   0
    0           0           0.5*T^2     0   0   0
    T           0           0           0   0   0
    0           T           0           0   0   0
    0           0           T           0   0   0
    0           0           0           T   0   0
    0           0           0           0   T   0
    0           0           0           0   0   T];

%Matriz de ruídos da dinâmica
C=10*[  0.0017*0.5*T^2  0               0               0           0
0
        0               0.0017*0.5*T^2  0               0           0
0
        0               0               0.0017*0.5*T^2  0           0
0
        0.0017*T        0               0               0           0
0
        0               0.0017*T        0               0           0
0
        0               0               0.0017*T        0           0
0
        0               0               0               0.0001163*T 0
0
        0               0               0               0
0.0001163*T 0
0       0               0               0               0
0.000163*T];

%Covariancia do ruído de dinâmica
Q=eye(6);
```

```
%Matriz de leitura dos sensores (GPS e Bussola)
H=[ 1   0   0   0   0   0   0   0   0   %Xe
    0   1   0   0   0   0   0   0   0   %Ye
    0   0   1   0   0   0   0   0   0   %Ze
    0   0   0   1   0   0   0   0   0   %Vxe
    0   0   0   0   1   0   0   0   0   %Vye
    0   0   0   0   0   1   0   0   0   %Vze
    0   0   0   0   0   0   1   0   0   %roll
    0   0   0   0   0   0   0   1   0   %pitch
    0   0   0   0   0   0   0   0   1]; %yaw

%Matriz de ruídos de medição
G=[ 0.75    0       0       0       0       0       0       0       0
    0       0.75    0       0       0       0       0       0       0
    0       0       0.75    0       0       0       0       0       0
    0       0       0       0.03    0       0       0       0       0
    0       0       0       0       0.03    0       0       0       0
    0       0       0       0       0       0.03    0       0       0
    0       0       0       0       0       0       0.0175  0       0
    0       0       0       0       0       0       0       0.0175  0
    0       0       0       0       0       0       0       0
0.0175];

%Covariancia do ruído de mediçao
R=eye(9);


%Condições iniciais (Dadas as condiçoes de variância)
x(1,1)=Apoena(1,19);
x(2,1)=Apoena(1,20);
x(3,1)=Apoena(1,21);
x(4,1)=Apoena(1,16);
x(5,1)=Apoena(1,17);
x(6,1)=Apoena(1,18);
x(7,1)=Apoena(1,7);
x(8,1)=Apoena(1,8);
x(9,1)=Apoena(1,9);
x(:,1)=x(:,1)+[ 0.75    0       0       0       0       0       0
0       0
                0       0.75    0       0       0       0       0
0       0
                0       0       0.75    0       0       0       0
0       0
                0       0       0       0.03    0       0       0
0       0
                0       0       0       0       0.03    0       0
0       0
                0       0       0       0       0       0.03    0
0       0
                0       0       0       0       0       0
0.0175^2 0       0
                0       0       0       0       0       0       0
0.0175^2 0
                0       0       0       0       0       0       0
0       0.0175^2]*randn(9,1);
```

```
%Matriz de covariancia do estado inicial
P(:,:,1)=[  5^2     5^2     5^2     5^2     5^2     5^2     5^2
5^2         5^2
            5^2     5^2     5^2     5^2     5^2     5^2     5^2
5^2         5^2
            5^2     5^2     5^2     5^2     5^2     5^2     5^2
5^2         5^2
            5^2     5^2     5^2     1^2     1^2     1^2     1^2
1^2         1^2
            5^2     5^2     5^2     1^2     1^2     1^2     1^2
1^2         1^2
            5^2     5^2     5^2     1^2     1^2     1^2     1^2
1^2         1^2
            5^2     5^2     5^2     1^2     1^2     1^2     0.0175^2
0.0175^2    0.0175^2
            5^2     5^2     5^2     1^2     1^2     1^2     0.0175^2
0.0175^2    0.0175^2
            5^2     5^2     5^2     1^2     1^2     1^2     0.0175^2
0.0175^2    0.0175^2];

for t=1:1:(length(Apoena(:,1))-1)
    %Aceleraçao medida pela IMU (Inertial Mesurement Unit)
    %IMU=[Aceleração de corpo e Taxa de rotação] + [Gravidade e
Rotação da
    %terra(desprezível)] + [ruído de medição de aceleraçao e taxa de
    %rotação]
    IMU_body_g=[Apoena(t,10) Apoena(t,11) Apoena(t,12) Apoena(t,4)
Apoena(t,5) Apoena(t,6)]+[Apoena(t,29) Apoena(t,30) Apoena(t,31) 0  0
0]+[randn(1,3)*0.0017 randn(1,3)*0.0001163];
    IMU_body_g=IMU_body_g';
    %Variáveis da iteraçao anterior
    phi=x(7,t);    %Roll
    the=x(8,t);    %Pitch
    psi=x(9,t);    %Yaw

    %Subtraçao da aceleraçao da gravidade da aceleraçao apresentada
pela
    %IMU
    g=sqrt(Apoena(t,29)^2+Apoena(t,30)^2+Apoena(t,31)^2);
    DCM=inv([cos(the)*cos(psi)        (sin(phi)*sin(the)*cos(psi)-
cos(phi)*sin(psi))      (cos(phi)*sin(the)*cos(psi)+sin(phi)*sin(psi))
        cos(the)*sin(psi)
(sin(phi)*sin(the)*sin(psi)+cos(phi)*cos(psi))
(cos(phi)*sin(the)*sin(psi)-sin(phi)*cos(psi))
        -sin(the)               sin(phi)*cos(the)
cos(phi)*cos(the)                                ]);
    IMU_body=IMU_body_g-[DCM*[0 0 g]'; 0 ;0 ;0];
```

```
%Mudança para o sistema inercial:aceleração e taxa de rotação
    IMU_inertial=[  cos(the)*cos(psi)
(sin(phi)*sin(the)*cos(psi)-cos(phi)*sin(psi))
(cos(phi)*sin(the)*cos(psi)+sin(phi)*sin(psi))      0       0
0
                    cos(the)*sin(psi)
(sin(phi)*sin(the)*sin(psi)+cos(phi)*cos(psi))
(cos(phi)*sin(the)*sin(psi)-sin(phi)*cos(psi))      0       0
0
                    -sin(the)               sin(phi)*cos(the)
cos(phi)*cos(the)                                   0       0
0
                    0                       0
0                                                   1
sin(phi)*tan(the)   cos(phi)*tan(the)
                    0                       0
0                                                   0       cos(phi)
-sin(phi)
                    0                       0
0                                                   0
sin(phi)*sec(the)   cos(phi)*sec(the)]*IMU_body;

    %Ciclo de atualização
    if(rem((t*T),T_GPS)==0)
        t*T
        %Leituras de GPS
        GPS=[Apoena(t,19) Apoena(t,20) Apoena(t,21) Apoena(t,16)
Apoena(t,17) Apoena(t,18)]+[randn(1,3)*0.75 randn(1,3)*0.03];

        %Leitura da Bússula
        Bus=[Apoena(t,7) Apoena(t,8) Apoena(t,9)]+randn(1,3)*0.0175;
        %Filtro de Kalman propriamente dito
        %Observações
        z(:,t)=[GPS Bus]';

        %Ganho de Kalman
        k=P(:,:,t)*H'*inv(H*P(:,:,t)*H'+G*R*G');

        %Média
        x(:,t)=x(:,t)+k*(z(:,t)-H*x(:,t));

        %Covariancia
        P(:,:,t)=P(:,:,t)-k*H*P(:,:,t);

    end
    %Ciclo de propagação
        %Média
        x(:,t+1)=A*x(:,t)+B*IMU_inertial;
        %Covariância
        P(:,:,t+1)=A*P(:,:,t)*A'+C*Q*C';
end

subplot(4,1,1)
plot(Apoena(:,32))
ylabel('Rotation [RPM]')
subplot(4,1,2)
plot(Apoena(:,33))
ylabel('Elevator Deflection [°]')
subplot(4,1,3)
plot(Apoena(:,34))
```

```
ylabel('Aileron Deflection [°]')
subplot(4,1,4)
plot(Apoena(:,35))
ylabel('Rudder Deflection [°]')
xlabel('t [s]')

subplot(2,1,1)
plot(rad2deg(Apoena(:,22)))
ylabel('Attack Angle [°]')
subplot(2,1,2)
plot(rad2deg(Apoena(:,23)))
ylabel('SideSlip Angle [°]')
xlabel('t [s]')

plot3(Apoena(:,19),Apoena(:,20),-Apoena(:,21))
xlabel('X [m]')
ylabel('Y [m]')
zlabel('Z [m]')
axis equal
hold on
plot3(x(1,:),x(2,:),-x(3,:),'--r')
legend('Real','Filtro Kalman')

plot(Apoena(:,19),Apoena(:,20))
xlabel('X [m]')
ylabel('Y [m]')
hold on
axis equal
plot(x(1,:),x(2,:),'--r')
legend('Real','Filtro Kalman')

plot(-Apoena(:,21))
xlabel('t [s]')
ylabel('Z [m]')
hold on
plot(-x(3,:),'--r')
legend('Real','Filtro Kalman')

subplot(3,1,1)
plot(Apoena(:,19)-x(1,:)')
ylabel('X error [m]')
subplot(3,1,2)
plot(Apoena(:,20)-x(2,:)')
ylabel('Y error [m]')
subplot(3,1,3)
plot(Apoena(:,21)-x(3,:)')
ylabel('Z error [m]')
xlabel('t [s]')

subplot(3,1,1)
plot(Apoena(:,16))
hold on
plot(x(4,:),'--r')
ylabel('Vx [m/s]')
legend('Real','Filtro Kalman')
subplot(3,1,2)
plot(Apoena(:,17))
hold on
plot(x(5,:),'--r')
ylabel('Vy [m/s]')
```

```
legend('Real','Filtro Kalman')
subplot(3,1,3)
plot(Apoena(:,18))
hold on
plot(x(6,:),'--r')
ylabel('Vz [m/s]')
legend('Real','Filtro Kalman')
xlabel('t [s]')

subplot(3,1,1)
plot(Apoena(:,16)-x(4,:)')
ylabel('Vx error [m/s]')
subplot(3,1,2)
plot(Apoena(:,17)-x(5,:)')
ylabel('Vy error [m/s]')
subplot(3,1,3)
plot(Apoena(:,18)-x(6,:)')
ylabel('Vz error [m/s]')
xlabel('t [s]')

subplot(3,1,1)
plot(rad2deg(Apoena(:,7)))
hold on
plot(rad2deg(x(7,:)),'--r')
ylabel('Roll [°]')
legend('Real','Filtro Kalman')
subplot(3,1,2)
plot(rad2deg(Apoena(:,8)))
hold on
plot(rad2deg(x(8,:)),'--r')
ylabel('Pitch [°]')
legend('Real','Filtro Kalman')
subplot(3,1,3)
plot(rad2deg(Apoena(:,9)))
hold on
plot(rad2deg(x(9,:)),'--r')
ylabel('Yaw [°]')
legend('Real','Filtro Kalman')
xlabel('t [s]')

subplot(3,1,1)
plot(rad2deg(Apoena(:,7)-x(7,:)'))
ylabel('Roll error [°]')
subplot(3,1,2)
plot(rad2deg(Apoena(:,8)-x(8,:)'))
ylabel('Pitch error [°]')
subplot(3,1,3)
plot(rad2deg(Apoena(:,9)-x(9,:)'))
ylabel('Yaw error [°]')
xlabel('t [s]')
```

B. Características dos Sensores Utilizados

- **IMU**

	Unidade de Medição Inercial XBow VG700AA
Taxa de aquisição:	100Hz
Taxa de Rotação	
• Bias (Roll, Pitch, Yaw):	0,03°/s
• Random walk:	0.4 °/sqrt(h)

$$ARW\left(\frac{^\circ}{\sqrt{h}}\right)=\frac{1}{60}\sigma\left(\frac{^\circ}{h}\right)\cdot\frac{1}{\sqrt{BW(Hz)}}\Rightarrow\sigma\left(\frac{^\circ}{h}\right)=ARW\left(\frac{^\circ}{\sqrt{h}}\right)60.\sqrt{BW(Hz)}$$

$$\therefore\sigma\left(\frac{rad}{s}\right)=\frac{\frac{\pi}{180}}{3600}ARW\left(\frac{^\circ}{\sqrt{h}}\right)60.\sqrt{BW(Hz)}=\frac{\pi}{10800}.ARW\left(\frac{^\circ}{\sqrt{h}}\right).\sqrt{BW(Hz)}$$

• Acurácia equivalente (sr):	0,0012 (rad/s)
Aceleração	
• Bias (Roll, Pitch, Yaw):	0,0085g
• Random walk:	1.0 m/s/sqrt(h)

$$ARW\left(\frac{m/s}{\sqrt{h}}\right)=\frac{1}{60}\sigma\left(\frac{m/s}{h}\right)\cdot\frac{1}{\sqrt{BW(Hz)}}\Rightarrow\sigma\left(\frac{m/s}{h}\right)=ARW\left(\frac{m/s}{\sqrt{h}}\right).60.\sqrt{BW(Hz)}$$

$$\therefore\sigma\left(\frac{m}{s^2}\right)=\frac{\frac{\pi}{180}}{3600}ARW\left(\frac{m/s}{\sqrt{h}}\right).60.\sqrt{BW(Hz)}=\frac{\pi}{10800}.ARW\left(\frac{m/s}{\sqrt{h}}\right).\sqrt{BW(Hz)}$$

• Acurácia equivalente (sa):	0,000920 (m/s²)

- **Bússola**

	Bússola PNI corp TCM2-20
Taxa de aquisição:	1Hz
Acurácia da medição de atitude (st):	1,5° RMS
Acurácia da medição do campo magnetico:	0,2μT

- **GPS**

	GPS Novatel Propak II RTK
Taxa de aquisição:	1Hz
Acurácia da medição de posição:	0,75m RMS
Acurácia da medição de velocidade (sp):	0,03m/s RMS