MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-12977-PRE/8255

ALGORITMO COMPLETO PARA INTEGRAÇÃO EM SISTEMAS INERCIAIS SOLIDÁRIOS (STRAPDOWN)

Bruno Mohallem Paiva*

*Bolsista UNIFEI

Relatório Final de Projeto de Iniciação Científica (PIBIC/CNPq/INPE), orientado pelo Dr. Antônio Félix Martins Neto

INPE
São José dos Campos
2005

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

ALGORITMO COMPLETO PARA INTEGRAÇÃO EM SISTEMAS INERCIAIS SOLIDÁRIOS (STRAPDOWN)

RELATÓRIO FINAL DE PROJETO DE INICIAÇÃO CIENTÍFICA (PIBIC/CNPq/INPE)

Bruno Mohallem Paiva (UNIFEI, Itajubá, MG, Bolsista PIBIC/CNPq)
E-mail: bmohallem2@hotmail.com

Prof. Dr. Antônio Félix Martins Neto (DMC/INPE, São José dos Campos, SP, Orientador)

COLABORADOR
Eng. M.C. Henrique Mohallem Paiva (doutorando, ITA, São José dos Campos, SP)

Junho de 2005

AGRADECIMENTOS

Gostaria de agradecer ao meu orientador, Prof. Martins Neto, pelo grande suporte dado durante o desenvolvimento deste trabalho, pela disposição em sempre me ajudar e por ter aceitado mais um ano de desafio, para a continuação do desenvolvimento do nosso trabalho. Gostaria, também, de agradecer ao INPE, pelo suporte dado à pesquisa, e ao PIBIC/CNPq, pelo apoio financeiro durante esses dois anos de Iniciação Científica. Por fim, gostaria de agradecer ao meu irmão Henrique, pelo auxílio no entendimento da teoria geral dos sistemas de navegação inercial, pelo fornecimento de material extra para pesquisa e pelo constante incentivo.

SUMÁRIO

I. Introdução

A navegação inercial consiste essencialmente em utilizar, a bordo de um veículo, as sensações devidas ao movimento absoluto deste veículo, levando em conta a existência do campo de gravidade local [5]. Os sensores inerciais comumente utilizados são acelerômetros, que medem as acelerações específicas, e girômetros, que medem velocidade angular (girômetros de saída analógica) ou incrementos angulares (girômetros de saída digital) [6].

As medidas resultantes permitem estimar [5]:

a) a posição, a velocidade e a aceleração do veículo em um referencial conhecido;

b) a atitude, a velocidade angular e a aceleração angular do veículo em um referencial conhecido.

A determinação de atitude de um veículo, através de um procedimento inercial, pode ser efetuada de duas formas [5].:

a) estabilizando uma plataforma inercial na qual estão fixos giroscópios (ou girômetros) e medindo a atitude relativa entre o veículo e a plataforma;

b) fixando diretamente um bloco giroscópico (ou girométrico) sobre a estrutura do veículo, e processando os sinais fornecidos por essa aparelhagem. Os sistemas desse tipo são chamados de sistemas solidários ou "strapdown".

Sistemas "strapdown" apresentam a vantagem de uma maior simplicidade mecânica e oferecem certas facilidades para organizar a redundância do sistema de medida. Por outro lado, apresentam certas desvantagens, como a necessidade de cálculos mais complexos via software, realizados a uma freqüência mais elevada [2]. Contudo, o desenvolvimento computacional das últimas décadas viabilizou a utilização dessa alternativa, fazendo com que essa seja a opção mais adotada atualmente.

Em sistemas "strapdown", o conhecimento da atitude, que é determinada a partir das medidas girométricas, permite transformar medidas de força específica de acelerômetros fixos ao veículo para o sistema da horizontal local [6]. Nesta configuração, os algoritmos utilizados para determinação de atitude são responsáveis por grande parte da acurácia da navegação [7].

Para esse sistema de navegação, vários sistemas de coordenadas podem ser utilizados. Neste estudo serão utilizados os sistemas I, B, L, N e E, definidos a seguir:

1) O sistema I é um sistema inercial usado como referência para medidas de rotação angular. Neste estudo, ele estará fixo no centro da Terra;

2) O sistema E é o sistema de referência da Terra. Ele está fixo no centro Terra e acompanha a rotação desta. Seu eixo Z está na direção do eixo de rotação da Terra e seus os eixos X e Y estão no plano equatorial terreste.
3) O sistema N é o sistema de navegação, de nível local, fixo no centro de massa do veículo. Seu eixo Z é paralelo à direção vertical (em relação à Terra) e apontado para cima. Os eixos X e Y são posicionados convenientemente. Neste estudo utilizaremos o sistema ENU (East, North, Up), isto é, o eixo X apontando para a direção Leste, o eixo Y apontando para a direção Norte e o eixo Z apontando para cima.
4) O sistema L é também um sistema de nível local, fixo no centro de massa do veículo. Seu eixo Z é também paralelo à direção vertical, mas apontado para baixo, ao contrário do sistema N. Os eixos X e Y são posicionados convenientemente. Neste estudo utilizaremos o sistema NED (North, East, Down), isto é, o eixo X apontando para a direção Norte, o eixo Y apontando para a direção Leste e o eixo Z apontando para baixo. Note que há uma relação direta e constante no tempo entre os sistemas N e L.
5) O sistema B está fixo no centro de massa do veículo e acompanha suas rotações.

Os sistemas B, L, N e E variam no tempo em relação ao sistema I. Para explicitar a notação utilizada, $B_{I_{(m)}}$ e $L_{I_{(m)}}$ serão definidos como a orientação dos sistemas B e L em relação ao sistema inercial I no instante t_m de atualização computacional.

Os cálculos envolvidos na navegação inercial são um pouco complexos, aparecendo freqüentemente equações diferenciais não-lineares. Hughes [4] detalha os conceitos algébricos da cinemática vetorial. Neste estudo fazemos a conversão entre os sistemas de coordenadas através de matrizes de cossenos diretores. Existe, ainda, a opção de fazer tal conversão através de *quaternions*.

Vários algoritmos de aproximação para as equações cinemáticas são propostos na literatura, cada um com suas características e comportamentos próprios e erros inerentes.

Na primeira etapa deste trabalho [1] foram abordados os conceitos e os cálculos referentes à determinação da atitude do veículo. Na etapa atual, damos enfoque à continuação dos cálculos para a determinação da posição do veículo. Para efetuar esses cálculos utilizaremos as aproximações propostas por Savage [2] [3].

No primeiro trabalho foram abordado os conceitos teóricos sobre o efeito de *coning*, que recai sobre a atitude do móvel. O efeito de *sculling*, que recai sobre a velocidade, é análogo, e por isso, seus conceitos teóricos não serão explicitados.

II. Objetivos

Em continuação ao trabalho desenvolvido pelo autor no primeiro ano de Iniciação Científica, os objetivos do presente trabalho são explicitar de forma clara o algoritmo de navegação inercial proposto por Savage [2] [3], complementá-lo com as equações que não são explicitadas nos artigos, implementar o algoritmo completo e realizar testes com o algoritmo implementado. Como a codificação do algoritmo em C++ estava apresentando erros não identificados, optou-se por implementar o algoritmo em MatLab, que possui maior suporte à depuração de erros e visualização de resultados.

III. Nomenclatura

A, A_1, A_2, A_3	Sistemas de coordenadas arbitrários
$C_{A_2}^{A_1}$	Matriz de cossenos diretores que transforma um vetor expresso no sistema A_2 para o seu correspondente no sistema A_1.
I	Matriz identidade
V	Vetor sem nenhum sistema de coordenadas específico
V^A	Matriz coluna com as coordenas do vetor em relação ao sistema A
$V^A \times$	Produto vetorial de V^A, representado pela matriz quadrada a seguir, onde V_{XA}, V_{YA} e V_{ZA} são as componentes de V^A $\begin{bmatrix} 0 & -V_{ZA} & V_{YA} \\ V_{ZA} & 0 & -V_{XA} \\ -V_{YA} & V_{XA} & 0 \end{bmatrix}$ O produto matricial de ($V^A \times$) com outro vetor do sistema A é igual ao produto vetorial de V^A com esse outro vetor
$V_{(t)}$	Matriz, vetor ou escalar referente ao ciclo computacional *t*.
$V_{t_2}^{t_1}$	Matriz, vetor ou escalar avaliado entre os instantes t_1 e t_2, onde $t_2 > t_1$

IV. Algoritmo utilizado

IV.I. Introdução

Para a determinação da posição do móvel em relação ao referencial da Terra (latitude, longitude e altura), é utilizado um algoritmo composto por três velocidades de atualização. O algoritmo foi proposto por Savage [2] [3], e engloba duas etapas, que ocorrem de forma mesclada. Uma etapa faz os cálculos referentes às medidas girométricas, e a outra etapa faz basicamente os cálculos referentes às medidas acelerométricas, além de fazer uma pequena correção da matriz de atitudes.

A uma velocidade mais rápida (ciclo computacional *l*, de período *Tl*), além de se contabilizarem as medidas girométricas e acelerométricas, são também contabilizados os efeitos de coning, para a atitude, e de sculling e de rotação, para velocidade.

Na etapa de velocidade média (ciclo computacional *m*, de período *Tm*), é calculada a matriz de atitude C_B^L, e também a velocidade do móvel em relação ao referencial de navegação (N) e feita a integração dela, para o cálculo da variação do vetor posição do móvel.

Na etapa lenta (ciclo computacional *n*, de período *Tn*), é feita uma correção na matriz C_B^L, levando-se em consideração a rotação do sistema L, e é calculada, finalmente, a posição do móvel em relação à superfície da Terra (latitude, longitude e altura).

O algoritmo referente aos cálculos com as medidas girométricas já foi discutido na primeira etapa desta pesquisa [1], realizada entre agosto de 2003 e julho de 2004. Contudo, em tal etapa, considerou-se desprezível a variação do sistema L em relação a ele mesmo durante os cálculos (considerou-se o sistema L sem rotação). Essa aproximação não será mais válida para este trabalho, e, assim, serão apresentados os cálculos referentes a essa rotação.

Na seção a seguir será discutido o equacionamento do algoritmo, e na seção posterior será mostrada sua implementação em MatLab.

IV.II. Equacionamento

Neste algoritmo as unidades são dadas no SI e utilizaremos as seguintes constantes [8] :

Raio maior da Terra: a = 6378137,0m

Raio menor da Terra: b = 6356752,3m

Excentricidade elíptica da Terra: e = 0,0818

Velocidade angular de rotação da Terra: $\omega_{IE}^{E} = 7{,}292115110.10^{-5}$rad/s

Utilizando as aproximações propostas por Savage [1] [2], as seguintes equações foram utilizadas:

→ **<u>Cálculos do efeito de coning e de sculling (etapa rápida, passo computacional *l*)</u>:**

Inicialmente, contabiliza-se o incremento $\Delta\alpha_l$ da velocidade angular integrada e também o incremento Δv_l da força específica integrada, ambos medidos no instante t_l,

$$\begin{aligned} &\alpha_{(l)} = \alpha_{(l-1)} + \Delta\alpha_{(l)} \\ &\alpha_{(l)} = 0 \ \text{ em } \ t = t_{m\text{-}1} \\ &\alpha_{(m)} = \alpha_{(l)}\left(t_l = t_m\right) \end{aligned} \qquad (1)$$

onde $\Delta\alpha_{(l)}$ é a medida do girômetro no instante t_l

.

$$\begin{aligned} &v_{(l)} = v_{(l-1)} + \Delta v_{(l)} \\ &v_{(l)} = 0 \ \text{ em } \ t = t_{m\text{-}1} \\ &v_{(m)} = v_{(l)}\left(t_l = t_m\right) \end{aligned} \qquad (2)$$

onde $\Delta v_{(l)}$ é a medida do acelerômetro no instante t_l

.

Com o resultado obtido, calcula-se o incremento de coning $\beta_{(m)}$, o incremento de rotação $\Delta v_{\mathrm{rot}(m)}$ e o incremento de sculling $\Delta v_{scul(m)}$:

$$\Delta\beta_{(l)} = \frac{1}{2}\left(\alpha_{(l-1)} + \frac{1}{6}\Delta\alpha_{(l-1)}\right)\times\Delta\alpha_{(l)}$$

$$\beta_{(l)} = \beta_{(l-1)} + \Delta\beta_{(l)}$$

$$\beta_{(l)} = 0 \text{ em } t = t_{m\text{-}1} \qquad (3)$$

$$\beta_{(m)} = \beta_{(l)}\left(t_l = t_m\right)$$

$$\delta v_{scul\,l} = \frac{1}{2}\left[\left(\alpha_{(l-1)} + \frac{1}{6}\Delta\alpha_{(l-1)}\right)\times\Delta v_{(l)} + \left(v_{(l-1)} + \frac{1}{6}\Delta v_{(l-1)}\right)\times\Delta\alpha_{(l)}\right]$$

$$\Delta v_{scul(l)} = \Delta v_{scul(l-1)} + \delta v_{scul(l)}$$

$$\Delta v_{scul(l)} = 0 \text{ em } t = t_{m\text{-}1} \qquad (4)$$

$$\Delta v_{scul(m)} = \Delta v_{scul(l)}\left(t_l = t_m\right)$$

→ **<u>Cálculos da matriz atitude (etapa de velocidade normal, passo computacional *m*)</u>:**

Calcula-se, inicialmente, o vetor Φ_m de rotação do sistema B:

$$\Phi_{(m)} = \alpha_{(m)} + \beta_{(m)} \qquad (5)$$

Então se calcula a matriz $C_{B_{I(m)}}^{B_{I(m-1)}}$ de rotação do sistema B em relação a ele próprio (para a atualização da matriz de atitude):

$$C_{B_{I(m)}}^{B_{I(m-1)}} = I + \frac{\text{sen}(|\Phi_{(m)}|)}{|\Phi_{(m)}|}(\Phi_{(m)}\times) + \frac{1-\cos(|\Phi_{(m)}|)}{|\Phi_{(m)}|^2}(\Phi_{(m)}\times)(\Phi_{(m)}\times) \qquad (6)$$

Como nesse ponto a rotação do sistema L é desprezível, ela não é contabilizada, e a atualização da matriz de atitude C_B^L é dada por:

$$C_{B_{I(m)}}^{L_{I(n-1)}} = C_{B_{I(m-1)}}^{L_{I(n-1)}} \cdot C_{B_{I(m)}}^{B_{I(m-1)}} \qquad (7)$$

→ **<u>Cálculos da variação da posição (velocidade normal, passo computacional *m*)</u>:**

Calculamos, primeiramente, o incremento de velocidade devida ao efeito de rotação:

$$\Delta v_{\mathrm{rot}(m)} = \tfrac{1}{2}\left(\alpha_{(m)} \times v_{(m)}\right) \tag{8}$$

Em seguida, calculamos o incremento de força específica, expresso no sistema B:

$$\Delta v^{B_{I(m-1)}}_{SF_{(m)}} = v_{(m)} + \Delta v_{\mathrm{rot}(m)} + \Delta v_{\mathrm{scul}(m)} \tag{9}$$

Então expressamos o incremento de força específica no sistema L:

$$\Delta v^{L_{(n-1)}}_{SF_{(m)}} = C^{L_{I(n-1)}}_{B_{I(m-1)}} \cdot \Delta v^{B_{I(m-1)}}_{SF_{(m)}} \tag{10}$$

Considerando a rotação de L pequena o suficiente, temos, então, que o incremento de força específica referente ao ciclo *m* é aproximadamente igual ao incremento de força calculado pela equação 10:

$$\Delta v^{L}_{SF_{(m)}} \cong \Delta v^{L_{(n-1)}}_{SF_{(m)}} \tag{11}$$

Calculamos, em seguida, a correção da velocidade devido à gravidade e ao efeito de Coriolis através de:

$$\Delta v^{N}_{G/COR_{(m)}} = \left\{ g^{N}_{p\,(m-\frac{1}{2})} - \left[2\omega^{N}_{IE_{(m-\frac{1}{2})}} + \rho_{ZN_{(m-\frac{1}{2})}} \cdot u^{N}_{ZN} + Fc_{(m-\frac{1}{2})} \cdot \left(u^{N}_{ZN} \times v^{N}_{(m-\frac{1}{2})} \right) \right] \times v^{N}_{(m-\frac{1}{2})} \right\} \cdot Tm \tag{12}$$

Conforme proposto por Savage, uma grandeza X pode ser estimada no meio do intervalo *m* de computação matemática através de seus valores anteriores. Definamos, para essa estimação, a representação $X_{(m-1/2)}$. Para a velocidade, Savage propõe a seguinte aproximação:

$$\mathbf{v}^{\mathrm{N}}_{\left(m-\frac{1}{2}\right)} = \tfrac{3}{2}\,\mathbf{v}^{\mathrm{N}}_{(m-1)} - \tfrac{1}{2}\,\mathbf{v}^{\mathrm{N}}_{(m-2)} \qquad (13)$$

Para as demais grandezas, Savage propõe a seguinte aproximação:

$$X_{\left(m-\frac{1}{2}\right)} = X_{(n-1)} + \frac{\left(r-\frac{1}{2}\right)}{j}\left(X_{(n-1)} - X_{(n-2)}\right) \tag{14}$$

onde: j é o número de ciclos m em cada ciclo n

r é o número de ciclos m desde o último ciclo n

A partir da velocidade angular ω_{IE}^{E} da Terra expressa em seu próprio referencial, calculamos a velocidade angular da Terra expressa no sistema N através de:

$$\omega_{IE}^{N} = \left(C_{N}^{E}\right)^{T} \cdot \omega_{IE}^{E} \tag{15}$$

O modelo de gravidade *plumb-bob* da Terra utilizado [9], em função da latitude e da altura, é dado por:

$$g_{p}^{N} = \begin{bmatrix} 0 \\ 0 \\ -\gamma(\lambda, h) \end{bmatrix}$$

$$\gamma(\lambda, h) = \gamma(\lambda) - \left[3{,}0877.10^{-6} - 0{,}0044.10^{-6}\,\mathrm{sen}^{2}(\lambda)\right]h + 0{,}072.10^{-12}.h^{2} \tag{16}$$

$$\gamma(\lambda) = 9{,}780327.\left[1 + 0{,}0053024.\mathrm{sen}^{2}(\lambda) - 0{,}0000058.\mathrm{sen}^{2}(2\lambda)\right]$$

O modelo geodético da Terra considerado foi o modelo elíptico [8], cujos valores dos parâmetros estão definidos no começo desta seção. Para esse modelo, a matriz Fc de curvatura da Terra é dada por:

$$Fc = \begin{bmatrix} \frac{1}{R_{\lambda} + h} & 0 & 0 \\ 0 & \frac{1}{R_{\varphi} + h} & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad (17)$$

Os raios de latitude R_λ e de longitude R_φ são dados por [10]:

$$R_\lambda = \frac{a\left(1-e^2\right)}{\left(1-e^2 sen^2\gamma\right)^{3/2}}$$
$$R_\varphi = \frac{a}{\left(1-e^2 sen^2\gamma\right)^{1/2}} \qquad (18)$$

O vetor vertical u_{ZN}^N do sistema N, expresso nesse mesmo sistema de coordenadas é dado por:

$$u_{ZN}^N = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad (19)$$

A componente vertical ρ_{ZN} da velocidade de transporte é dada por [10]:

$$\rho_{ZN} = \frac{v_e^N}{\left(R_\varphi + h\right)} \cdot tg\lambda \qquad (20)$$

onde v_e^N é a componente leste (primeira componente) de v^N

A matriz de transformação entre os sistemas L e N é constante e dada por:

$$C_L^N = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{21}$$

Achamos, a seguir, o valor da velocidade do móvel, expressa no referencial N:

$$v_{(m)}^N = v_{(m\text{-}1)}^N + C_L^N \Delta v_{SF_{(m)}}^L + \Delta v_{G/COR_{(m)}}^N \tag{22}$$

Integrando a velocidade (por integração trapezoidal), achamos a variação do vetor posição $\Delta R_{(m)}^N$:

$$\Delta R_{(m)}^N = \tfrac{1}{2}\left(v_{(m)}^N + v_{(m\text{-}1)}^N\right) \cdot Tm \tag{23}$$

→ **Correção da matriz de atitude (etapa lenta, passo computacional *n*):**

Além do cálculo da posição do móvel, na etapa computacional mais lenta também se faz uma correção na matriz de atitude, conforme descrito a seguir.

Inicialmente, calcula-se o vetor de rotação do sistema L:

$$\zeta_{(n)} = \left(C_{L}^{N}\right)^{T}\left[\omega_{IE_{\left(n-\frac{1}{2}\right)}}^{N}\cdot Tn \;+\; \rho_{ZN_{\left(n-\frac{1}{2}\right)}}\cdot u_{ZN}^{N}\cdot Tn \;+\; Fc_{\left(n-\frac{1}{2}\right)}\left(u_{ZN}^{N}\times\sum_{m=1}^{j}\Delta R_{(m)}^{N}\right)\right] \quad (24)$$

Através dele é, então, calculada matriz de cossenos diretores de atualização do sistema L:

$$C_{L_I(n-1)}^{L_I(n)} = I - \frac{\operatorname{sen}\left|\zeta_{(n)}\right|}{\left|\zeta_{(n)}\right|}\left(\zeta_{(n)}\times\right) + \frac{1-\cos\left|\zeta_{(n)}\right|}{\left|\zeta_{(n)}\right|^{2}}\left(\zeta_{(n)}\times\right)\left(\zeta_{(n)}\times\right) \quad (25)$$

Finalmente, faz-se a correção da matriz de atitude, de conversão entre os sistemas B e L:

$$C_{B_I(m)}^{L_I(n)} = C_{B_I(m)}^{L_I(n-1)}\cdot C_{L_I(n-1)}^{L_I(n)} \quad (26)$$

→ **Cálculo da posição do móvel (etapa lenta, passo computacional *n*):**

Inicialmente, calcula-se a variação da altura:

$$\Delta h_{(n)} = u_{ZN}^{N}\sum_{m=1}^{j}\Delta R_{(m)}^{N} \quad (27)$$

Então a altura é atualizada:

$$h_{(n)} = h_{(n-1)} + \Delta h_{(n)} \tag{28}$$

O vetor de rotação da posição é calculado pela seguinte expressão:

$$\xi_{(n)} = \rho_{ZN_{\left(n-\frac{1}{2}\right)}} \cdot u_{ZN}^{N} \cdot Tn \quad + Fc_{\left(n-\frac{1}{2}\right)} \left(u_{ZN}^{N} \times \sum_{m=1}^{j} \Delta R_{(m)}^{N} \right) \tag{29}$$

Analogamente à aproximação feita nos ciclos computacionais *m*, a seguinte aproximação é proposta para a estimativa do valor das grandezas X no meio do intervalo n:

$$X_{\left(n-\frac{1}{2}\right)} = \tfrac{3}{2} X_{(n-1)} - \tfrac{1}{2} X_{(n-2)} \tag{30}$$

Através do vetor de rotação, calcula-se a atualização da matriz de posição:

$$C_{N_{E(n)}}^{N_{E(n\text{-}1)}} = I + \frac{\operatorname{sen}\left(\left|\xi_{(n)}\right|\right)}{\left|\xi_{(n)}\right|} \left(\xi_{(n)} \times\right) + \frac{1-\cos\left(\left|\xi_{(n)}\right|\right)}{\left|\xi_{(n)}\right|^{2}} \left(\xi_{(n)} \times\right)\left(\xi_{(n)} \times\right) \tag{31}$$

Faz-se, então, a atualização da matriz de posição:

$$C_{N_{E(n)}}^{E} = C_{N_{E(n-1)}}^{E} \cdot C_{N_{E(n)}}^{N_{E(n\text{-}1)}} \tag{32}$$

Em seguida, obtém-se a matriz de transformação do sistema E para o sistema L:

$$C_E^L = \left(C_L^N \cdot C_N^E\right)^T \tag{33}$$

Sabe-se que a matriz C_E^L tem a seguinte forma [10]:

$$C_E^L = \begin{bmatrix} -\text{sen}(\lambda)\cos(\varphi) & -\text{sen}(\lambda)\text{sen}(\varphi) & \cos(\lambda) \\ -\text{sen}(\varphi) & \cos(\varphi) & 0 \\ -\cos(\lambda)\cos(\varphi) & -\cos(\lambda)\text{sen}(\varphi) & -\text{sen}(\lambda) \end{bmatrix} \tag{34}$$

Assim, dos termos a_{13} e a_{22} dessa matriz, obtemos diretamente os valores da latitude λ e da longitude φ (ressalta-se que tanto latitude quanto longitude variam de -90º a +90º):

$$\begin{aligned} \lambda_{(n)} &= \text{arcsen}(-a_{33}) \\ \varphi_{(n)} &= \text{arcsen}(-a_{21}) \end{aligned} \tag{35}$$

Assim, têm-se calculadas todas as coordenadas da posição do móvel: λ, φ e h.

IV.III. Implementação

No primeiro plano de trabalho para o segundo ano de Iniciação Científica foi decido que se daria continuidade à implementação do algoritmo em C++, implementado no primeiro ano. Contudo, o código desenvolvido em C++ não apresentou os resultados esperados para casos de teste simples, e então se optou pela implementação em MatLab do algoritmo, utilizando como ponto de partida o código em MatLab apresentado na primeira etapa da pesquisa. A linguagem de MatLab apresenta duas grandes vantagens neste caso: ela é de maior nível de abstração (evita-se eventuais erros devidos a problemas com alocação de memória e vetores não-inicializados) e apresenta maior facilidade de depuração de código.

Apesar de a migração do código em C++ para MatLab ser relativamente fácil e direta, o código em MatLab foi desenvolvido unicamente a partir das equações apresentadas neste trabalho, para evitar eventuais erros cometidos durante a implementação do código em C++.

A implementação em MatLab foi feita em uma função chamada `algortimo4()`, presente no arquivo `algoritmo4.m`. Foram criadas algumas funções auxiliares, internas ao arquivo, para realizar cálculos de várias grandezas (uma função por grandeza ou por grandezas correlatas), dando, assim, maior clareza ao código. O código desenvolvido é mostrado a seguir:

```
function algoritmo4()

    evalin('base', 'clear all');
    clc;
    format long e;

    RAD2DEG = 180/pi; %constante de conversao radianos -> graus
    DEG2RAD = pi/180; %constante de conversao graus -> radianos

    % Constantes de simulaçao
    MAX_TEMPO = 2500;
    DELTA_TL = 0.01;
    DELTA_TM = 0.05;
    DELTA_TN = 0.2;

    lmax= round(DELTA_TM/DELTA_TL);
    mmax= round(DELTA_TN/DELTA_TM);
    nmax= round(MAX_TEMPO/DELTA_TN);

    % Constante - Eixo Z -> (0,0,1)
    uZN_N=[0.0; 0.0; 1.0];


    % Constante - Matriz de transformação entre o sistema L e o sistema N
    Cln = [0.0,   1.0,    0.0;
           1.0,   0.0,    0.0;
           0.0,   0.0,   -1.0];
```

Figura 1 – Arquivo algoritmo4.m – Parte 1/8

```
% Inicializaçao das condicoes iniciais
latitude_old = 50*DEG2RAD;
longitude_old = 60*DEG2RAD;
h_old = 0;

Cbl_old = [1 0 0; 0 1 0; 0 0 1];
Cel_old = matrizCel(latitude_old, longitude_old);
Cen_old = Cel_old*Cln;
Cne_old = Cen_old';

v_N_oldM1 = [0;0;0];
v_N_oldM2 = v_N_oldM1;
v_east_old = v_N_oldM1(1);  %Sistema N => ENU (East, North, Down)

gp_N_oldN1 = vetorGravidade(latitude_old, h_old);
gp_N_oldN2 = gp_N_oldN1;
wIE_N_oldN1 = velocidadeAngularTerra(Cne_old);
wIE_N_oldN2 = wIE_N_oldN1;
Fc_oldN1 = matrizCurvatura(latitude_old, h_old);
Fc_oldN2 = Fc_oldN1;
rhoZN_oldN1 = craftRate(v_east_old, latitude_old, h_old);
rhoZN_oldN2 = rhoZN_oldN1;

% Inicializacao do vetor de armazenamento da posicao
posicao = zeros(3,1,nmax);

% DEBUG - Inicializacao dos vetores de armazenamento das variaveis
alphaM_array = zeros(3,1,nmax*mmax);
betaM_array = zeros(3,1,nmax*mmax);
phiM_array = zeros(3,1,nmax*mmax);
Cbb_array = zeros(3,3,nmax*mmax);
Cbl_array = zeros(3,3,nmax*mmax);
vM_array = zeros(3,1,nmax*mmax);
deltaVRotM_array = zeros(3,1,nmax*mmax);
deltaVSculM_array = zeros(3,1,nmax*mmax);
deltaVSF_B_M_array = zeros(3,1,nmax*mmax);
deltaVSF_L_M_array = zeros(3,1,nmax*mmax);
v_N_M_array = zeros(3,1,nmax*mmax);
deltaR_N_M_array = zeros(3,1,nmax*mmax);
deltaVCor_N_M_array = zeros(3,1,nmax*mmax);
gp_N_halfM_array = zeros(3,1,nmax*mmax);
wIE_N_halfM_array = zeros(3,1,nmax*mmax);
rhoZN_halfM_array = zeros(1,nmax*mmax);
Fc_halfM_array = zeros(3,3,nmax*mmax);
v_N_halfM_array = zeros(3,1,nmax*mmax);
zeta_array = zeros(3,1,nmax);
epslon_array = zeros(3,1,nmax);
wIE_N_halfN_array = zeros(3,1,nmax);
rhoZN_halfN_array = zeros(1, nmax);
Fc_halfN_array = zeros(3,3,nmax);
Cll_array = zeros(3,3,nmax);
Cne_array = zeros(3,3,nmax);
Cel_array = zeros(3,3,nmax);

% Lista das constantes para serem armazenadas no workspace
constToWorkspace = {'DELTA_TL', 'MAX_TEMPO', 'DELTA_TM', 'DELTA_TN', ...
        'lmax', 'mmax', 'nmax', 'RAD2DEG', 'DEG2RAD', 'uZN_N', 'Cln'};
```

Figura 1 – Arquivo algoritmo4.m – Parte 2/8

```
% Lista dos arrays para serem armazenados no workspace
varToWorkspace = {'latitude', 'longitude', 'h', 'posicao', ...
  'zeta_array', 'epslon_array', 'wIE_N_halfN_array', 'rhoZN_halfN_array', ...
  'Fc_halfN_array', 'Cll_array', 'Cne_array', 'Cel_array', 'zeta_array', ...
  'epslon_array', 'wIE_N_halfN_array', 'rhoZN_halfN_array', 'Fc_halfN_array', ...
  'Cll_array', 'Cne_array', 'Cel_array', 'alphaM_array', 'betaM_array', ...
  'phiM_array', 'Cbb_array', 'Cbl_array', 'vM_array', 'deltaVRotM_array', ...
  'deltaVSculM_array', 'deltaVSF_B_M_array', 'deltaVSF_L_M_array', ...
  'v_N_M_array', 'deltaR_N_M_array', 'deltaVCor_N_M_array', ...
  'gp_N_halfM_array', 'wIE_N_halfM_array', 'rhoZN_halfM_array', ...
  'Fc_halfM_array', 'v_N_halfM_array'};


% DEBUG -> Figura para ver evolucao entre os sistemas de coordenadas
% fg = figure;

for n=1:nmax
    somaDeltaR_N = [0;0;0];
    for m=1:mmax
        % Inicializacao das variaveis
        deltaAlpha_old = [0;0;0];
        alpha_old = [0;0;0];
        beta_old = [0;0;0];
        vl_old = [0;0;0];
        deltaVl_old = [0;0;0];
        deltaVScul_old = [0;0;0];
        for L=1:lmax
            % Indice do instante L em relacao a simulacao total
            % Indice LN 1:MAX_TEMPO/DELTA_TL
            indiceLN = (n-1)*mmax + (m-1)*lmax + L;

            % Leitura do girometro
            deltaAlpha(:, L) = [0;0;0];
            alpha(:, L) = alpha_old + deltaAlpha(:, L);

            deltaBeta(:, L) = 0.5*cross((alpha_old + (1/6)*deltaAlpha_old), ...
deltaAlpha(:,L));
            beta(:, L) = beta_old + deltaBeta(:, L);

            % Geracao das medidas do acelerometro
            Cbn = Cbl_old*Cln;
            Cnb = Cbn';

            % Leitura do acelerometro
            deltaVl(:, L) = -Cbn*vetorGravidade(latitude_old, h_old)*DELTA_TL;
            vl(:, L) = vl_old + deltaVl(:, L);

            delVScul(:, L) = 0.5 * ( cross((alpha_old + (1/6)*deltaAlpha_old),...
deltaVl(:, L)) + cross((   vl_old + (1/6)*deltaVl_old)    , deltaAlpha(:, L)) );
            deltaVScul(:, L) = deltaVScul_old + delVScul(:, L);

            % Definiçao dos valores 'old' (L-1)
            deltaAlpha_old(:) = deltaAlpha(:, L);
            alpha_old(:) = alpha(:, L);
            beta_old(:) = beta(:, L);
            vl_old(:) = vl(:, L);
            deltaVl_old(:) = deltaVl(:, L);
            deltaVScul_old(:) = deltaVScul(:, L);
        end
```

Figura 1 – Arquivo algoritmo4.m – Parte 3/8

```
            alphaM = alpha(:, lmax);
            betaM = beta(:, lmax);

            % Vetor de rotacao entre os sistemas B e L
            phiM = alphaM+betaM;

            vM = vl(:, lmax);
            deltaVRotM = 0.5*(cross(alphaM,vM)) ; % cross(alphaM,vM) ==> alphaM^vM
            deltaVSculM = deltaVScul(:, lmax);

            %================= Calculo da matriz de atitude =================
            % Geraçao da matriz Cbb de atualizaçao da matriz C devido a rotacao de B
            nphiM = norm(phiM);
            skewPhiM = skew(phiM);
            if nphiM == 0.0
                Cbb = eye(3,3);
            else
                Cbb = eye(3,3) + sin(nphiM)/nphiM*skewPhiM + (1-cos(nphiM))/ ...
                  (nphiM*nphiM)*skewPhiM*skewPhiM;
            end

            % Atualizacao da matriz Cbl
            Cbl=Cbl_old*Cbb;

            %==================== Calculo da velocidade ====================
            deltaVSF_B_M = vM + deltaVRotM + deltaVSculM;
            deltaVSF_L_M = Cbl*deltaVSF_B_M;

            % Estimacao dos valores halfM (m-1/2)
            gp_N_halfM = gp_N_oldN1 + ((m-0.5)/mmax) * (gp_N_oldN1 - gp_N_oldN2);
            wIE_N_halfM = wIE_N_oldN1 + ((m-0.5)/mmax) * (wIE_N_oldN1 - wIE_N_oldN2);
            rhoZN_halfM = rhoZN_oldN1 + ((m-0.5)/mmax) * (rhoZN_oldN1 - rhoZN_oldN2);
            squeeze(rhoZN_halfM); % Tirando as dimensoes nao utilizadas
            Fc_halfM = Fc_oldN1 + ((m-0.5)/mmax) * (Fc_oldN1 - Fc_oldN2);
            v_N_halfM = (1.5)*v_N_oldM1 - (0.5)* v_N_oldM2;

            deltaVCor_N_M = DELTA_TM * (gp_N_halfM - cross( (2*wIE_N_halfM + ...
uZN_N.* rhoZN_halfM + Fc_halfM*cross(uZN_N,v_N_halfM)), (v_N_halfM) ) );

            v_N_M = v_N_oldM1 + Cln*deltaVSF_L_M + deltaVCor_N_M;
            %v_N_M = [0;0;0];

            deltaR_N_M = 0.5*(v_N_M + v_N_oldM1)*DELTA_TM;
            somaDeltaR_N = somaDeltaR_N + deltaR_N_M;


            % Definicao dos valores 'old' (m-1) e (m-2)
            v_N_oldM2 = v_N_oldM1;
            v_N_oldM1 = v_N_M;
            v_east_old = v_N_oldM1(1);
            Cbl_old = Cbl;

            % DEBUG - Salvando as variaveis em arrays
            iArrayM = (n-1)*mmax+m;

            alphaM_array(:, :, iArrayM) = alphaM;
            betaM_array(:, :, iArrayM) = betaM;
            phiM_array(:, :, iArrayM) = phiM;
            Cbb_array(:, :, iArrayM) = Cbb;
            Cbl_array(:, :, iArrayM) = Cbl;
            vM_array(:, :, iArrayM) = vM;
            deltaVRotM_array(:, :, iArrayM) = deltaVRotM;
```

Figura 1 – Arquivo algoritmo4.m – Parte 4/8

```
            deltaVSculM_array(:, :, iArrayM) = deltaVSculM;
            deltaVSF_B_M_array(:, :, iArrayM) = deltaVSF_B_M;
            deltaVSF_L_M_array(:, :, iArrayM) = deltaVSF_L_M;
            v_N_M_array(:, :, iArrayM) = v_N_M;
            deltaR_N_M_array(:, :, iArrayM) = deltaR_N_M;
            deltaVCor_N_M_array(:, :, iArrayM) = deltaVCor_N_M;
            gp_N_halfM_array(:, :, iArrayM) = gp_N_halfM;
            wIE_N_halfM_array(:, :, iArrayM) = wIE_N_halfM;
            rhoZN_halfM_array(iArrayM) = rhoZN_halfM;
            Fc_halfM_array(:, :, iArrayM) = Fc_halfM;
            v_N_halfM_array(:, :, iArrayM) = v_N_halfM;
        end

        % Estimacao dos valores halfN (n-1/2)
        wIE_N_halfN = (1.5)*wIE_N_oldN1 - (0.5)* wIE_N_oldN2;
        rhoZN_halfN = (1.5)*rhoZN_oldN1 - (0.5)* rhoZN_oldN2;
        squeeze(rhoZN_halfN); % Tirando as dimensoes nao utilizadas
        Fc_halfN = (1.5)*Fc_oldN1 - (0.5)* Fc_oldN2;

        % Correcao da matriz de atitude devido ao movimento da Terra
        zeta = Cln' * (wIE_N_halfN*DELTA_TN + rhoZN_halfN*uZN_N + ...
                 Fc_halfN * cross(uZN_N,somaDeltaR_N) );
        nZeta  = norm(zeta);
        skewZeta = skew(zeta);
        Cll = eye(3) - (sin(nZeta)/nZeta)*skewZeta + ( (1-cos(nZeta))/nZeta ) * ...
skewZeta * skewZeta;
        Cbl_old = Cbl_old * Cll;

        % Calculo do vetor de rotacao da posicao e determinacao da matriz de posicao
        epslon = rhoZN_halfN*uZN_N*DELTA_TN + Fc_halfN*cross(uZN_N, somaDeltaR_N);
        nEpslon = norm(epslon);
        skewEpslon = skew(epslon);
        if nEpslon==0.0
            Cnn = eye(3);
        else
            Cnn = eye(3) + (sin(nEpslon)/nEpslon)*skewEpslon + ((1-cos(nEpslon))/ ...
               nEpslon) * skewEpslon * skewEpslon;
        end

        Cne = Cne_old * Cnn;
        Cel = (Cln*Cne)';

        % Determinacao da posicao do movel
        deltaH = dot(uZN_N,somaDeltaR_N);
        h = h_old + deltaH;
        latitude  = asin(-Cel(3,3));
        longitude = asin(-Cel(2,1));

        % Armazenamento da posicao em array
        % (latitude [graus], longitude [graus], altura [metros])
        posicao(:, :, n) = [latitude*RAD2DEG; longitude*RAD2DEG; h];

        % Salvando os valores oldN
        latitude_old = latitude;
        h_old = h;
        Cne_old = Cne;

        gp_N_oldN2 = gp_N_oldN1;
        gp_N_oldN1 = vetorGravidade(latitude_old, h_old);

        wIE_N_oldN2 = wIE_N_oldN1;
        wIE_N_oldN1 = velocidadeAngularTerra(Cne_old);
```

Figura 1 – Arquivo algoritmo4.m – Parte 5/8

```
        Fc_oldN2 = Fc_oldN1;
        Fc_oldN1 = matrizCurvatura(latitude_old, h_old);

        rhoZN_oldN1 = craftRate(v_east_old, latitude_old, h_old);
        rhoZN_oldN2 = rhoZN_oldN1;

        % DEBUG - Armazenamento das variaveis em arrays
        zeta_array(:, :, n) = zeta;
        epslon_array(:, :, n) = epslon;
        wIE_N_halfN_array(:, :, n) = wIE_N_halfN;
        rhoZN_halfN_array(n) = rhoZN_halfN;
        Fc_halfN_array(:, :, n) = Fc_halfN;
        Cll_array(:, :, n) = Cll;
        Cne_array(:, :, n) = Cne;
        Cel_array(:, :, n) = Cel;

        % DEBUG -> Plotando evolucao do sistema local (L) em relacao ao da Terra (E)
        % plotSistemas(fg, Cel);
    end

    %===== Geracao dos graficos da evolucao temporal das coordenadas da posicao =====
    latitude = squeeze(real(posicao(1,1,:)));
    longitude = squeeze(real(posicao(2,1,:)));
    h = squeeze(real(posicao(3,1,:)));
    t = (1:length(latitude))*DELTA_TN;
    % Geracao das strings para as figuras
    strSimParamFigFileName = ['Tl-' num2str(DELTA_TL) '_Tm-' num2str(DELTA_TM) ...
        '_Tn-' num2str(DELTA_TN) '_Tmax-' num2str(MAX_TEMPO)];
    strSimParamFigTitle = ['(Tl:' num2str(DELTA_TL) 's Tm:' num2str(DELTA_TM) ...
        's Tn:' num2str(DELTA_TN) 's Tmax:' num2str(MAX_TEMPO) 's)'];
    strFigFileNameLatitude = ['Latitude_' strSimParamFigFileName];
    strFigFileNameLongitude = ['Longitude_' strSimParamFigFileName];
    strFigFileNameAltura = ['Altura_' strSimParamFigFileName];
    strFigTitleLatitude = ['Latitude ' strSimParamFigTitle];
    strFigTitleLongitude = ['Longitude ' strSimParamFigTitle];
    strFigTitleAltura = ['Altura ' strSimParamFigTitle];

    % Figura com evolucao temporal da latitude
    figure; plot(t, latitude);
    set(gcf, 'FileName', strFigFileNameLatitude);
    set(gca, 'FontSize', 8);
    XLabel('Tempo (s)', 'FontWeight', 'bold');
    YLabel('Latitude (graus)', 'FontWeight', 'bold');
    Title(strFigTitleLatitude, 'FontWeight', 'bold');

    % Figura com evolucao temporal da longitude
    figure; plot(t, longitude);
    set(gcf, 'FileName', strFigFileNameLongitude);
    set(gca, 'FontSize', 8);
    XLabel('Tempo (s)', 'FontWeight', 'bold');
    YLabel('Longitude (graus)', 'FontWeight', 'bold');
    Title(strFigTitleLongitude, 'FontWeight', 'bold');

    % Figura com evolucao temporal da altura
    figure; plot(t, h);
    set(gcf, 'FileName', strFigFileNameAltura);
    set(gca, 'FontSize', 8);
    XLabel('Tempo (s)', 'FontWeight', 'bold');
    YLabel('Altura (m)', 'FontWeight', 'bold');
    Title(strFigTitleAltura, 'FontWeight', 'bold');
```

Figura 1 – Arquivo algoritmo4.m – Parte 6/8

```
    % Armazenamento das constantes e das variaveis no workspace
    for iConst=1:length(constToWorkspace)
        eval(['assignin(''base'', ''' constToWorkspace{iConst} ''', ' ...
constToWorkspace {iConst} ');'])
    end
    for iVar=1:length(varToWorkspace)
        eval(['assignin(''base'', ''' varToWorkspace{iVar} ''', ' ...
        varToWorkspace{iVar} ');'])
    end
return


%======================== Funcoes auxiliares ========================

% Retorna, no vetor de entrada 'raios', os raios de latitude
% e longitude da Terra, em função da latitude
function [rLatitude, rLongitude]=raiosTerra(latitude)
    %Constantes
    a = 6378137.0; %Raio maior da Terra, em [m]
    e = 0.0818; %Excentricidade da Terra
    rLatitude = a*(1.0-e*e) / ((1.0-e*e*sin(latitude)*sin(latitude))^1.5);
    rLongitude = a / ((1.0-e*e*sin(latitude)*sin(latitude))^0.5);
return

% Retorna a matriz de transformação do sistema E para o sistema L,
% em função da latitude e da longitude
function Cel = matrizCel(latitude, longitude)
    Cel= ...
[-sin(latitude)*cos(longitude),-sin(latitude)*sin(longitude),cos(latitude);
 -sin(longitude)              ,cos(longitude)               ,0.0;
 -cos(latitude)*cos(longitude),-cos(latitude)*sin(longitude),-sin(latitude)];
return


%Retorna o vetor gravidade, em função da latitude e da altura
function gravidade = vetorGravidade(latitude, h)
    %Modelo elipsóide da Terra
    gama_l = 9.780327*(1.0+0.0053024*sin(latitude)*sin(latitude)- ...
       0.0000058*sin(2.0*latitude)*sin(2.0*latitude));
    gama_hl = gama_l - (3.0877E-6 - ...
       0.0044E-6*sin(latitude)*sin(latitude))*h + 0.072E-12*h*h;
    gravidade=[0.0; 0.0; -gama_hl];
return


% Retorna o vetor velocidade angular da Terra, expresso no referencial N
function wIE_N=velocidadeAngularTerra (Cne)
    Cen = Cne';
    % Constante - Velocidade angular de rotação da Terra
    wIE_E= 0.0000729211511 * [0.0; 0.0; -1.0];
    wIE_N = Cen * wIE_E;
return


% Retorna a velocidade vertical de transporte (craft rate)
function rhoZN = craftRate(v_east, latitude, h)
    [rLatitude, rLongitude] = raiosTerra(latitude);
    rhoZN = (v_east) * tan(latitude) / (rLongitude + h);
return
```

Figura 1 – Arquivo algoritmo4.m – Parte 7/8

```
%Retorna matriz de curvatura da Terra
function Fc = matrizCurvatura(latitude, h)
    [rLatitude, rLongitude]=raiosTerra(latitude);
    Fc =  [ 1.0 / (rLatitude+h) ,  0.0                  ,  0.0 ;
             0.0                ,  1.0/(rLongitude+h)   ,  0.0 ;
            0.0                 ,  0.0                  ,  0.0 ];
return

function skewSymmetric = skew(vetor)
    skewSymmetric = [    0        -vetor(3)   vetor(2) ;
                      vetor(3)       0      -vetor(1) ;
                     -vetor(2)    vetor(1)      0     ];
end


% DEBUG -> Plota a relacao entre dois sistemas de coordenadas
function plotSistemas(fg, Cab)
    figure(fg);
    % Plotando sistema de coordenadas de referencia (A)
    plot3([1,0],[0,0],[0,0], 'k'), hold on;
    plot3([0,0],[1,0],[0,0], 'k');
    plot3([0,0],[0,0],[0,1], 'k');
    % Plotando sistema de coordenadas B no sistema de referencia A
    x=(Cab'*[1;0;0]);
    plot3([0, x(1)], [0, x(2)], [0, x(3)], 'r');
    y=(Cab'*[0;1;0]);
    plot3([0, y(1)], [0, y(2)], [0, y(3)], 'g');
    z=(Cab'*[0;0;1]);
    plot3([0, z(1)], [0, z(2)], [0, z(3)], 'b');
    set(gca, 'XLim', [-1,1]);
    set(gca, 'YLim', [-1,1]);
    set(gca, 'ZLim', [-1,1]);
    hold off;
    drawnow;
end
```

Figura 1 – Arquivo algoritmo4.m – Parte 8/8

IV. Análise do algoritmo

O primeiro teste realizado foi um caso bastante simples: um móvel parado na superfície da Terra (altura nula), com latitude e longitude zero. Para este caso, a leitura do girômetro deve ser nula e o acelerômetro deve indicar leitura referente à força de reação da superfície (já que ele ignora a ação da gravidade).

A variável referente à leitura do girômetro foi zerada e ajustada para sempre indicar essa constante. A variável referente à leitura do acelerômetro foi ajustada para indicar o oposto do valor da gravidade multiplicado por *Tl*, em cada iteração computacional *l*.

Além disso, foram utilizadas as seguintes constantes de simulação: *Tl* = 2,0s, *Tm* = 8,0s., *Tn* = 40,0s e *Tmáx* = 40000,0s. Essas constantes foram utilizadas também nas próximas simulações.

Os gráficos das coordenadas de posição obtidos foram os seguintes:

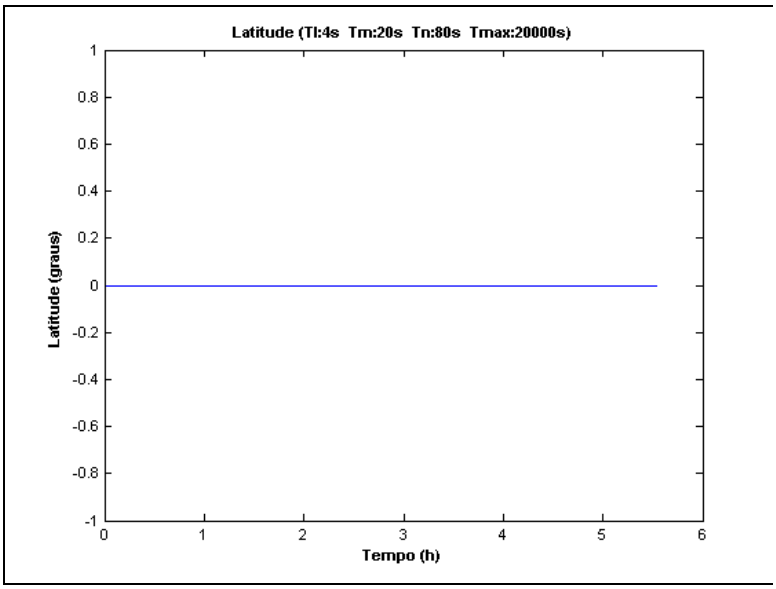

Figura 2.a – Comportamento da latitude com condições iniciais Latitutde = 0º, Longitude = 0º e Altura = 0m

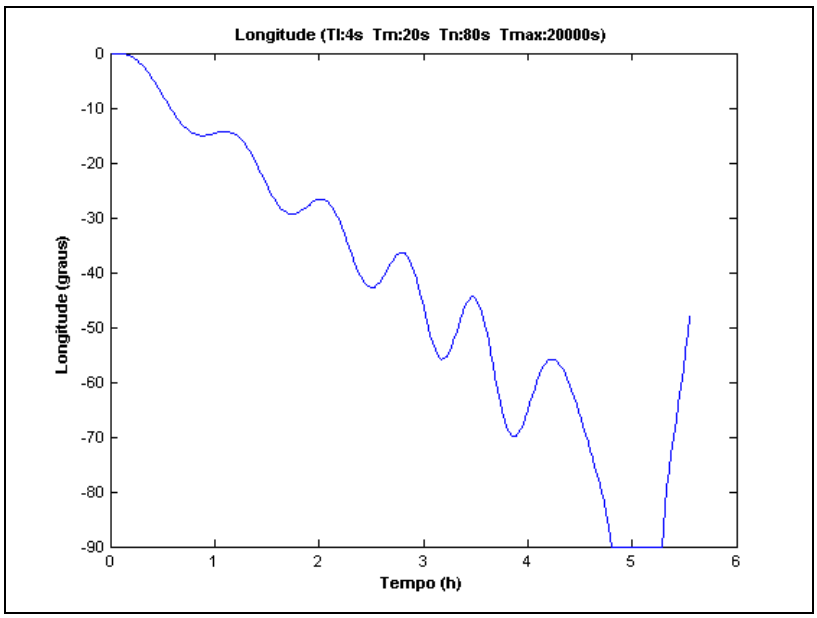

Figura 2.b – Comportamento da longitude com condições iniciais
Latitutde = 0º, Longitude = 0º e Altura = 0m

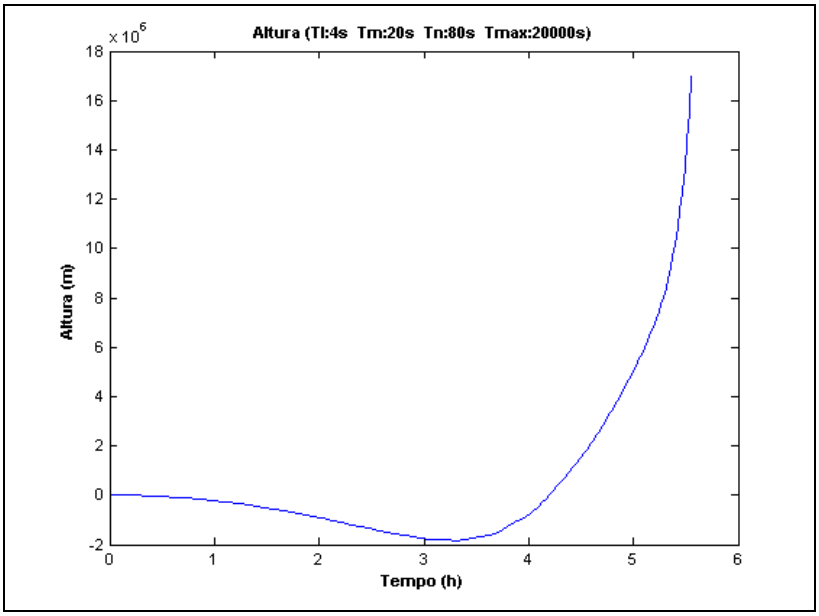

Figura 2.c – Comportamento da altura com condições iniciais
Latitutde = 0º, Longitude = 0º e Altura = 0m

Esperava-se que as coordenadas mantivessem tendência constante ao longo do tempo, aceitando-se uma pequena margem de tolerância para erros de aproximação do algoritmo. Notou-se, entretanto, que houve uma discrepância muito grande na longitude e na altura. No erro da longitude notou-se uma componente crescente, associada a uma componente oscilatória.

Simulou-se, então, um segundo caso, com latitude +30º, longitude +40º e altura 1000m. Os gráficos obtidos foram os seguintes:

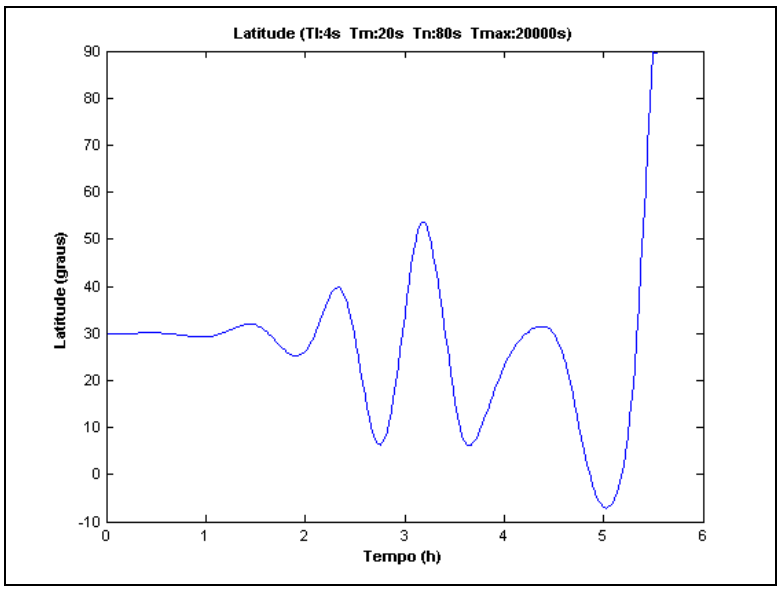

Figura 3.a – Comportamento da latitude com condições iniciais
Latitutde = +30º, Longitude = +40º e Altura = 1000m

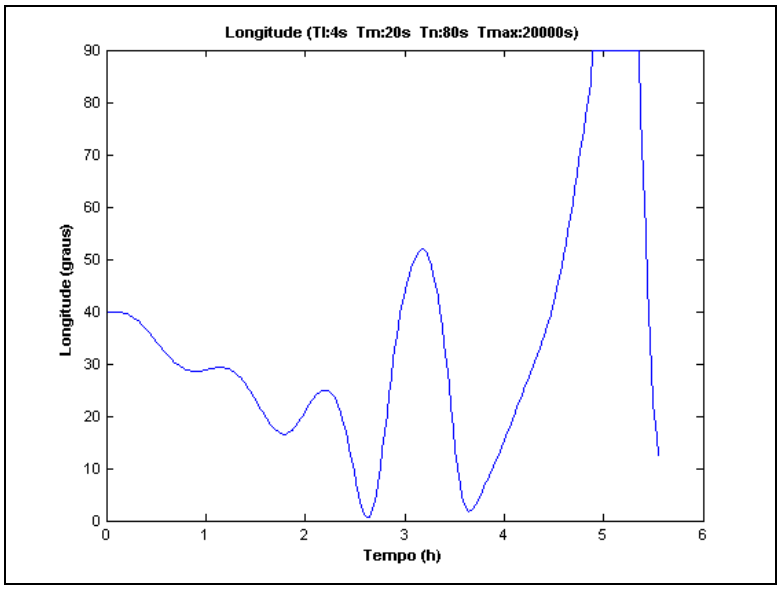

Figura 3.b – Comportamento da longitude com condições iniciais
Latitutde = +30º, Longitude = +40º e Altura = 1000m

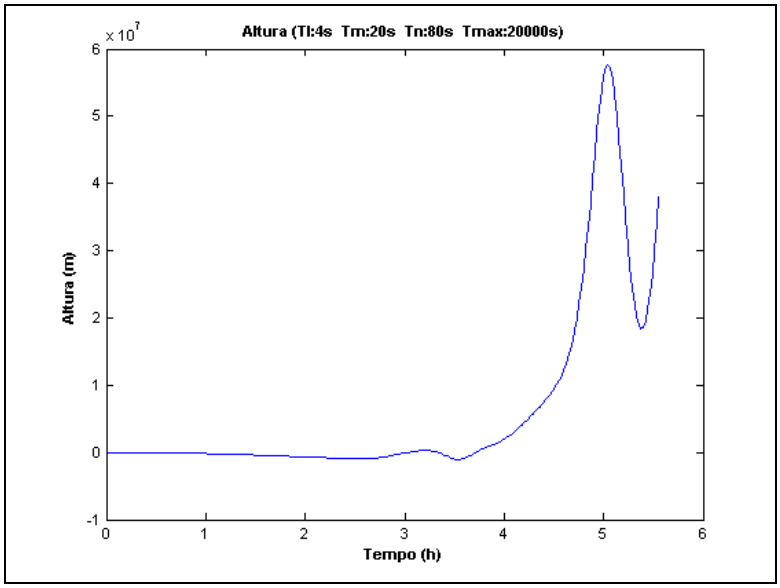

Figura 3.c – Comportamento da altura com condições iniciais
Latitutde = +30º, Longitude = +40º e Altura = 1000m

Com o aumento do módulo da latitude e da longitude a tendência de crescimento do erro foi ainda maior. Houve o aparecimento de ângulos complexos nas coordenadas, o que fez o algoritmo apontar resultados ainda mais espúrios.

Para investigar as possíveis causas de erro, fez-se uma análise das equações apresentadas neste trabalho aplicadas ao caso simulado e foram observadas as situações descritas a seguir:

→ Na equação 9, o incremento de força específica $\Delta v_{SF_{(m)}}^{B_{I(m-1)}}$ deve contabilizar apenas a soma das leituras do girômetro no intervalo *Tm*, o que é equivalente ao oposto da aceleração da gravidade integrada no intervalo *Tm*. Isso resulta no oposto do vetor gravidade multiplicado por *Tm*, expresso no referencial B.

→ Pelas equações 10 e 11, o incremento de força específica $\Delta v_{SF_{(m)}}^{B_{I(m-1)}}$ é passado para o referencial L, tornando-se $\Delta v_{SF_{(m)}}^{L}$, cujo valor é o oposto do vetor gravidade multiplicado por *Tm*, expresso no referencial L.

→ Na equação 12, o único efeito que é contabilizado em $\Delta v^{N}_{G/COR_{(m)}}$ é o efeito da gravidade, já que $v^{N}_{(m-1/2)}$ é zero. Assim, podemos notar que esse incremento de velocidade resulta no vetor gravidade multiplicado por *Tm*, expresso no referencial N.

→ Na equação 22, no segundo termo, o incremento de velocidade $\Delta v^{L}_{SF_{(m)}}$ é passado para o referencial N. Assim, o segundo e o terceiro termo da equação referida são exatamente opostos, fazendo com que a velocidade seja sempre constante. Como a velocidade inicial é zero, então isso faz com que a velocidade seja sempre zero.

Fazendo a depuração do código, foi observado que o segundo e o terceiro termo da equação 22 não se estavam anulando, fazendo com que houvesse uma variação da velocidade. Como uma velocidade não-nula faz com que o termo $\Delta v^{N}_{G/COR_{(m)}}$ não contabilize mais apenas o efeito da gravidade, pois passa também a contabilizar o efeito de Coriolis, a tendência é o segundo e o terceiro termo da equação 22 divergirem cada vez mais.

Para verificar se o problema era realmente esse, uma terceira simulação foi feita, forçando o termo $v^{N}_{(m)}$ ficar zerado (igualando esse termo a um vetor nulo, depois de ele ser calculado pelo algoritmo). A simulação foi realizada com latitude -35º30', longitude +17º15' e altura 1500m. Os gráficos obtidos foram os seguintes:

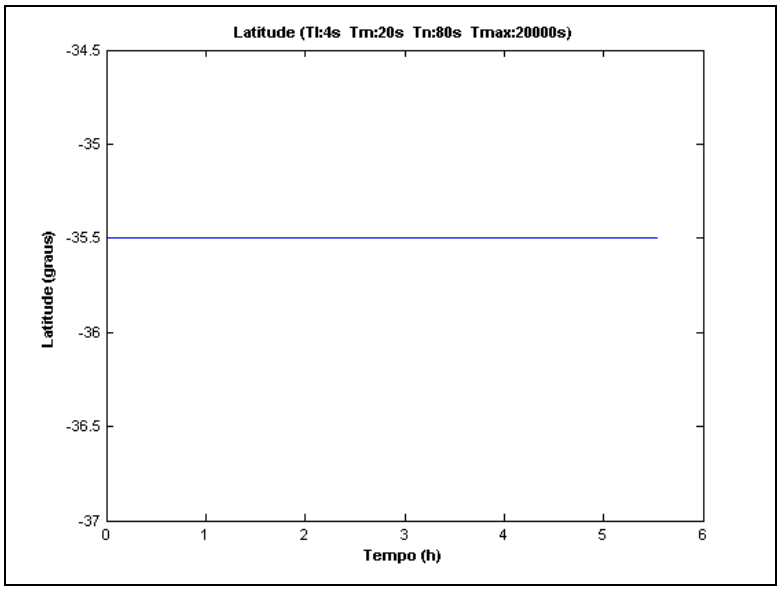

Figura 4.a – Comportamento da latitude com condições iniciais Latitutde = -35º30', Longitude = +17º15' e Altura = 1500m

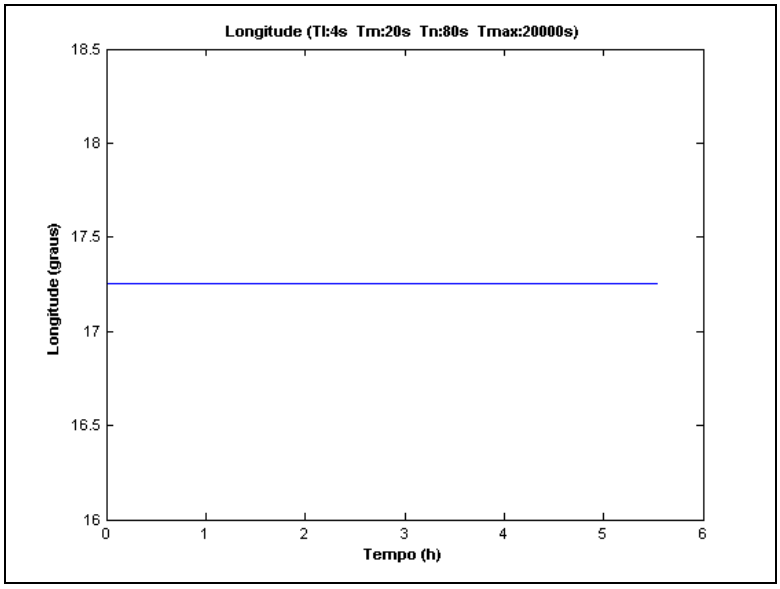

Figura 4.b – Comportamento da longitude com condições iniciais Latitutde = -35º30', Longitude = +17º15' e Altura = 1500m

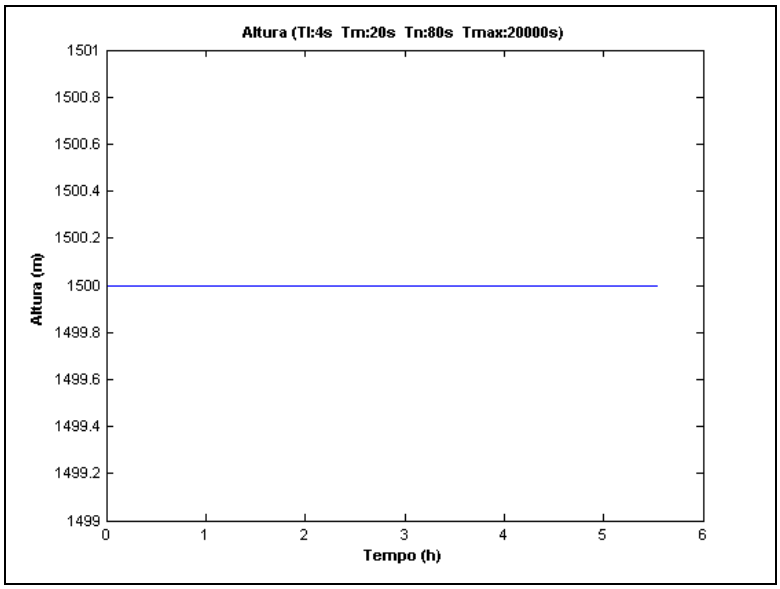

Figura 4.c – Comportamento da altura com condições iniciais Latitutde = -35º30', Longitude = +17º15' e Altura = 1500m

Notou-se que as coordenadas ficaram constantes, como era esperado. A mesma simulação foi feita para outras coordenadas iniciais, e o mesmo comportamento foi observado.

Assim, imagina-se que o erro esteja ocorrendo ou na contabilização das leituras do acelerômetro (incremento de força específica $\Delta v_{SF_{(m)}}^{B\,i(m-1)}$) ou no cálculo do incremento de velocidade $\Delta v_{G/COR_{(m)}}^{N}$, que contabiliza o efeito da gravidade e o efeito de Coriolis.

Os procedimentos de cálculo foram exaustivamente revisados, mas não se conseguiu notar nenhum erro aparente. As equações discutidas neste trabalho também foram exaustivamente revisadas, comparando-as com as equações originais propostas nas referências, mas também nenhum erro foi notado.

Diminuíram-se os tempos de amostragem, para observar seu efeito. Utilizaram-se os seguintes períodos de amostragem: *Tl* =0,01s, *Tm* = 0,05s, *Tn* = 0,2s e *Tmáx* = 1800,0s. A posição inicial utilizada foi (+50º, +60º, 0m). Os gráficos obtidos foram:

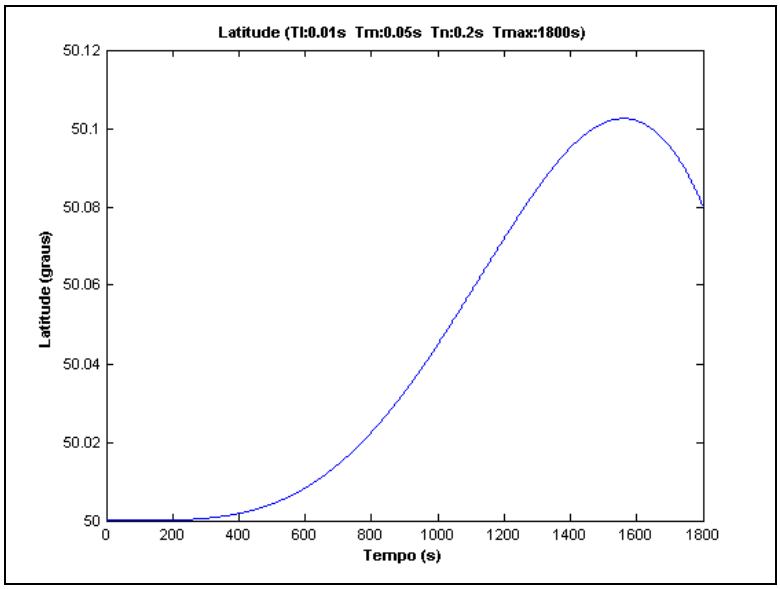

Figura 5.a – Comportamento da latitude com condições iniciais Latitutde = +50º, Longitude = +60º e Altura = 0m

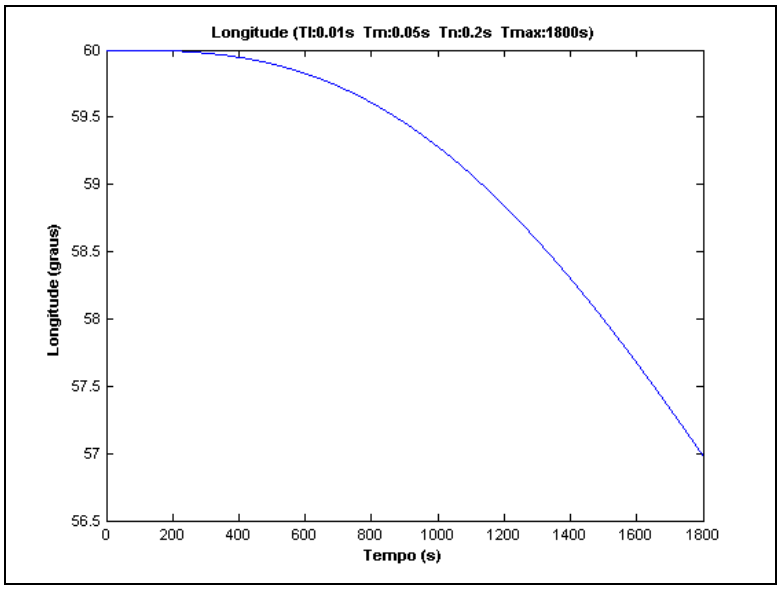

Figura 5.b – Comportamento da longitude com condições iniciais Latitutde = +50º, Longitude = +60º e Altura = 0m

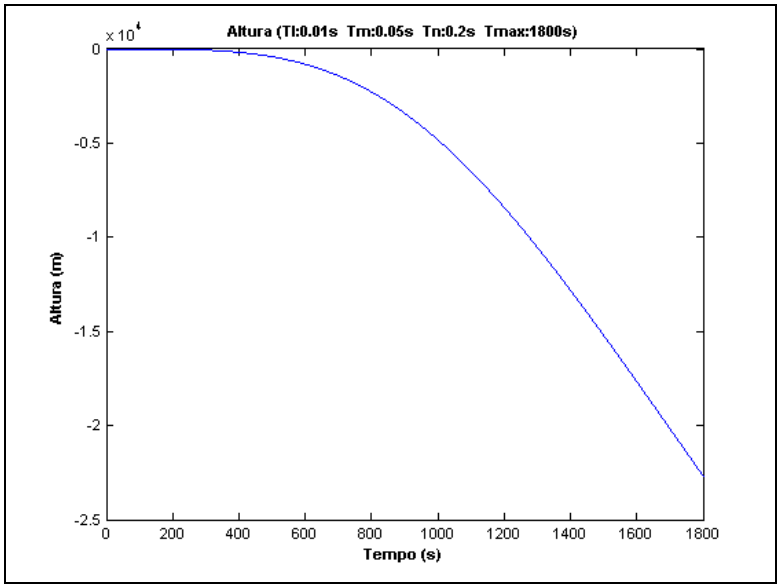

Figura 5.c – Comportamento da altura com condições iniciais Latitutde = +50º, Longitude = +60º e Altura = 0m

Como variações de alguns quilômetros na posição afetam pouco a latitude e a longitude, o impacto do erro é mais visível na altura.

Para verificar melhor o impacto do erro no caso simulado, é mostrado a seguir o gráfico do módulo da velocidade, medida no referencial N, em função do tempo.

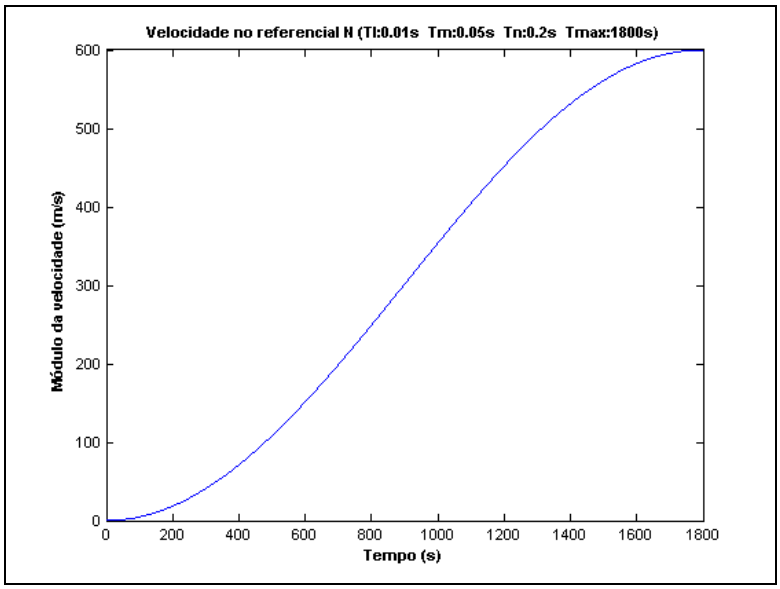

Figura 6 – Comportamento do módulo da velocidade no referencial N com condições iniciais Latitutde = +50º, Longitude = +60º e Altura = 0m

Observa-se que o módulo da velocidade aumenta ao longo do tempo, o que torna o erro de posição divergente, já que a velocidade esperada era nula.

Com a diminuição dos períodos de amostragem, conseguiu-se diminuir bastante os erros, mas estes continuaram sendo significativos.

V. Conclusões e trabalhos futuros

Os sistemas digitais de navegação inercial apresentam-se como uma boa opção para a determinação da localização espacial de veículos. Vários algoritmos para navegação inercial são propostos na literatura. Este trabalho foi focado na implementação do algoritmo de aproximação proposto por Savage. Na primeira etapa do trabalho da Iniciação Científica (primeiro ano de pesquisa) discutiu-se a determinação da matriz de atitude do veículo. Nesta segunda etapa, discutiu-se o restante do algoritmo.

Foi apresentada a idéia de um algoritmo com diferentes velocidades para se efetuar os cálculos: utiliza-se uma parte rápida, para determinação dos efeitos de coning e de sculling, uma parte de velocidade média, para a determinação da matriz de atitude, da velocidade do móvel e da variação do vetor posição do móvel, e uma etapa lenta, para a correção da matriz de atitude e determinação da posição do móvel (determinação das coordenadas latitude, longitude e altura).

Apresentaram-se os equacionamentos do algoritmo, com complementações oriundas de outras referências, explicitando as equações não citadas nos artigos do Savage considerados para este trabalho.

Foi desenvolvida uma implementação em MatLab para o algoritmo proposto. O código foi desenvolvido de forma bastante clara e com comentários explicitando os procedimentos adotados em cada passagem. A migração do código para MatLab foi válida, pois conseguiu-se corrigir diversos erros e foi possível fazer uma análise detalhada do problema,

Após o desenvolvimento da implementação, procedeu-se a análise do algoritmo para o caso de um móvel parado. Notou-se que os resultados obtidos não foram os esperados. Fez-se, então, a análise das equações apresentadas, aplicadas ao caso testado, e foram discutidas possibilidades de localização do erro.

A primeira etapa do trabalho apresentou a implementação de outros algoritmos para obtenção da matriz de posição e também apresentou a implementação em C++ de uma classe Matriz que fornece a essa linguagem de programação facilidade de operação matricial semelhante àquela que o MatLab possui para as necessidades deste trabalho. Foram feitas diversas simulações para se avaliar o erro de cada algoritmo e esses erros foram então analisados e discutidos.

O fato de o resultado final obtido pelo algoritmo completo ser diferente do esperado não invalidou os esforços desprendidos nesses dois anos de pesquisa. Durante esse tempo o autor adquiriu conhecimento significativo em cinemática vetorial, além de ter ganhado bastante familiaridade com os conceitos de navegação inercial.

Os trabalhos apresentados como resultado final da pesquisa apresentam as aproximações e expõem, de forma bastante organizada e clara, a seqüência das equações a serem implementadas.

A implementação apresentada do algoritmo de obtenção da posição é uma contribuição importante, pois representa um ponto de partida valioso para que se possa obter uma ferramenta que forneça resultados corretos.

Como proposta para trabalhos futuros sugere-se a análise e correção da implementação desenvolvida neste trabalho. Além disso, propõe-se estudar os erros na determinação da posição do móvel devido às aproximações propostas por Savage.

Outra proposta é estudar os algoritmos propostos na literatura para correção dos cálculos do sistema inercial através de interação com um sistema GPS.

Por fim, sugere-se fazer a migração dos códigos em MatLab para C++ para futura utilização como software embarcado.

VII. Referências Bibliográficas

[1] PAIVA, B. M. Algoritmos para Integração em Sistemas Inerciais Solidários (Strapdown). In *Anais do III Seminário de Iniciação do INPE – SICINPE2004*, Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2004.

[2] SAVAGE, P.G. Strapdown Inertial Navigation Integration Algorithm Design Part 1: Attitude Algorithms, *Journal of Guidance, Control, and Dynamics*, v. 21, n. 1, p. 19-28, 1998.

[3] SAVAGE, P.G., Strapdown Inertial Navigation Integration Algorithm Design Part 2: Velocity and Position Algorithms, *Journal of Guidance, Control and Dynamics*, v. 21, n. 2, p. 208-221, 1998.

[4] HUGHES, P.C., *Spacecraft Attitude Dynamics*, John Wiley, New York, 1986.

[5] RADIX, J.C.. *Systèmes Inertiels à Composant Liés "Strap-Down"*. Cépadues Éditions, France, 1980.

[6] WALDMANN, J. *Apostila do Curso de Sistemas de Navegação Inercial*, Instituto Tecnológico de Aeronáutica. São José dos Campos, Brasil, 1994.

[7] WALDMANN, J. Attitude determination algorithms, computational complexity, and the accuracy of terrestrial navigation with strapdown inertial sensors. In: *Anais do XIV Congresso Brasileiro de Automática*, Natal, Brasil, p. 2367-2372, 2002.

[8] BRITTING, K. R. *Inertial Navigation System Analysis*, Wiley-Interscience, New York, 1971.

[9] FARRELL, J. A. e BARTH, M., *The Global Positioning System and Inertial Navigation*, McGraw-Hill, New York, 1998.

[10] SIOURIS, G. M. *Aerospace Avionics Systems – A Modern Synthesis*, Academic Press, San Diego, 1993.