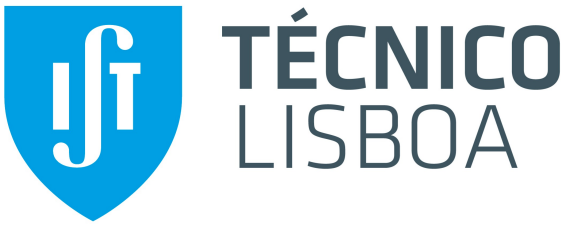

Instituto Superior Técnico

Sistemas de Controlo de Tráfego

2018-2019

Posicionamento do Recetor GPS

Projecto n° 1

Aluno:

Nuno Brandão, nº85232

Corpo docente:

Fernando Duarte Nunes

Índice

1 Introdução

O presente relatório relata uma simulação de um recetor GPS, com a análise dos erros de posição e de velocidade em diversos cenários. É estudado o efeito do número de satélites utilizados no erro de posição e de velocidade, e é feita uma simulação não considerando os erros ionosféricos.

O sistema GPS é um sistema de posicionamento por satélite que fornece a posição de um recetor móvel. Para tal, é necessário que o recetor tenha no mínimo 4 satélites visíveis, em que 3 deles são utilizados para a triangulação da posição do recetor (x_u, y_u, z_u) e o quarto é utilizado para a correção do relógio.

O recetor calcula a posição de cada satélite em cada momento. Uma vez que a velocidade das ondas eletromagnéticas é igual à velocidade da luz, são calculadas as distâncias entre o recetor e cada satélite. Esta distância denomina-se de pseudo-distância, que não corresponde à distância verdadeira uma vez que há erros provenientes da influência da ionosfera e da sincronização dos relógios associados ao seu cálculo. O erro de relógio é atenuado pelo $4^{\underline{o}}$ satélite, porém o erro da ionosfera é necessário ser minimizado em estações terrestres.

Os satélites utilizados corrrespondem aos N satélites que minimizam o parâmetro PDOP (*Position Dilution of Precision*), escolhidos dos satélites visíveis da posição do recetor.

2 Introdução Teórica

2.1 Determinação da constelação de satélites

A constelação de satélites em órbita é gerada a partir de informações de um Almanaque YUMA. Para o efeito, escolheram-se os dados referentes à semana 6 de Janeiro de 2008. Com estes dados, é possível calcular as ECEF (*Earth Centred - Earth Fixed*) dos satélites a partir de um referencial ECI (*Earth Centered Inertial Coordinates*).

O tempo no recetor é medido relativamente ao inicio da semana GPS. Seja t_{oe} o tempo de referência de efeméride, correspondendo ao tempo desde o inicio do calendário até à data em que os almanacs foram criados e seja t_{st} o instante em que o sinal é transmitido, a relação entre os tempos é $\Delta t = t_{oe} - t_{st}$ como é possível observar na figura 1:

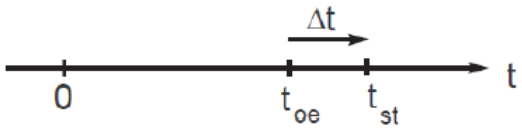

Figura 1: Relação entre tempos

Para o cálculo da posição dos satélites, foram adquiridos dados dos almanaques. Sejam as variáveis:

- e_0 - Excentricidade da órbita;
- α - Inclinação da órbita;
- $\dot{\Omega}$ - Taxa de variação da longitude do nó ascendente no instante t_{oe};
- $\sqrt{a}$ - Raiz quadrada do semi-eixo maior da órbita;
- Ω_0 - Ascensão do nó ascendente;
- ω - Argumento de perigeu;
- M_0 - Anomalia média no instante t_{oe};

A posição aproximada de um satélite GPS em coordenadas ECEF é então dada por:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} cos\theta cos\Omega - sen\theta sen\Omega cos\alpha \\ cos\theta cos\Omega + sen\theta cos\Omega cos\alpha \\ sen\Omega sen\theta \end{bmatrix} \quad (1)$$

em que $\theta = \upsilon + \omega$ é o argumento da latitude (ângulo entre o nó ascendente e a posição satélite no plano da órbita), υ é a anomalia verdadeira, Ω é a longitude no referêncial ECI e α é a inclinação do plano da órbita, de acordo com a figura 2.

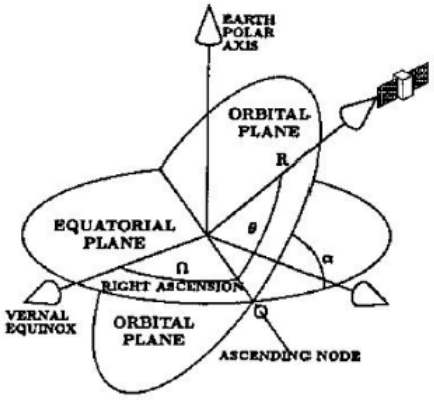

Figura 2: Esquema de coordenadas ECI

A anomalia verdadeira, υ, é dada por:

$$\upsilon = arctan_2(\frac{\sqrt{1-e_0^2}senE}{cos(E)-e_0} \tag{2}$$

em que E é a anomalia excêntrica, que é obtida através da iteração da equação de *Kepler* assumindo que a anomalia média $M = E - e_0 sen\ (E)$ é conhecida.

No instante de tempo em que o sinal é transmitido, t_{st}, a anomalia média é dada através da equação seguinte:

$$M = M_0 + n\Delta t \tag{3}$$

em que M_0 é a anomalia média no instante de referência de ephemeris t_oe, e $n = \frac{2\pi}{T} = \sqrt{\frac{\mu}{A^3}}$ é uma constante, em que T é o periodo da órbita do satélite, $\mu = 3.986 \times 10^{14} m^3/s^2$ é a constante gravitacional da terra e A é o comprimento do semi-eixo maior da órbita.

A longitude do nó ascendente no instante t_{st} é dada por:

$$\Omega = \Omega_0 + \dot{\Omega}\Delta t - \dot{\Omega}_e t_{st} \tag{4}$$

em que $\dot{\Omega}_e = 7.2921151467 \times 10^{-5} rad/s$ é a velocidade de rotação da terra.

Por fim, o raio da órbita é dado por:

$$R = A[1 - e_0 cos(E)] \tag{5}$$

2.2 Sistema de Coordenadas

O sistema de coordenadas utilizado na determinação da posição dos satélites é o sistema inercial *ECI*.

A base para a conversão entre coordenadas foi realizada com o modelo físico WGS-84 *(world geodetic system 1984)* expressando-se correspondentemente os termos latitude, longitude e altitude como (ϕ, λ, h). Posto isto, o cálculo das coordenadas ECEF do recetor (x_u, y_u, z_u) é dado por:

$$\begin{bmatrix} x_u \\ y_u \\ z_u \end{bmatrix} = \begin{bmatrix} (N+h)cos(\phi)cos(\lambda) \\ (N+h)cos(\phi)sin(\lambda) \\ (N(1-e^2)+h)sin(\phi) \end{bmatrix} \tag{6}$$

com

$$N = \frac{a}{\sqrt{1 - e^2 sin^2(\phi)}} \tag{7}$$

sendo a o semi eixo maior do plano do equador e e a excentricidade.

Por outro lado, é possível obter a latitude, longitude e altitude a partir das coordenadas em ECEF

$$\begin{bmatrix} \lambda \\ \phi \\ h \end{bmatrix} = \begin{bmatrix} arctan_2\left(\frac{y_u}{x_u}\right) \\ arctan_2\left(\frac{z_u}{p}\left(1 - e^2\frac{N}{N+h}\right)^{-1}\right) \\ \frac{p}{cos(\phi)} - N \end{bmatrix} \tag{8}$$

$$p = \sqrt{x_u^2 + y_u^2} \tag{9}$$

Na figura 3 apresenta-se o sistema de coordenadas ENU (*East-North Up*). Este sistema permite converter as coordenadas de um dado sistema em coordenadas relativas a um ponto P_u onde ϵ representa o ângulo de elevação, α o azimuth e (x,y,z) as coordenadas ENU de um dado satélite.

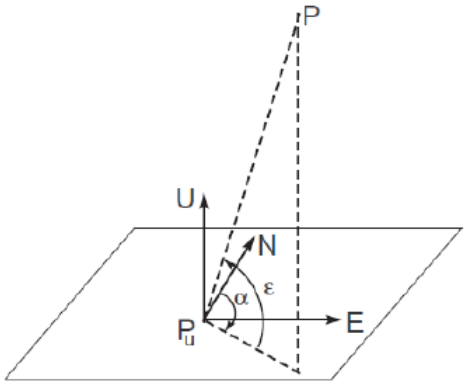

Figura 3: Sistema de coordenadas *ENU* (East-North Up)

$$\epsilon = arcsin\left(\frac{z}{\sqrt{x^2 + y^2 + z^2}}\right) \tag{10}$$

$$\alpha = arctan2(x, y) \tag{11}$$

Para fazer a conversão entre os diferentes sistemas de coordenadas utilizou-se as seguintes funções MATLAB:

- ecef2enu: recebe como argumentos as coordenadas ECEF do satélite, latitude, longitude e altitude do recetor, e o modelo físico utilizado WGS-84, retornando as coordenadas em ENU;

- enu2ecef: recebe como argumentos as coordenadas ENU do satélite, latitude, longitude e altitude do recetor, retornando as coordenadas ECEF dos satélites;

- ecef2lla: recebe como argumento as coordenadas ECEF, retornando a latitude, longitude e a altitude;

- lla2ecef: recebe a latitude, longitude e altitude, retornando as coordenadas ECEF.

2.3 Determinação da posição dos satélites e do recetor

A posição dos satélites visíveis é obtida com base no ângulo de elevação do satélite em relação ao horizonte local, em coordenadas ENU. A figura seguinte representa a elevação e o azimute de um satélite:

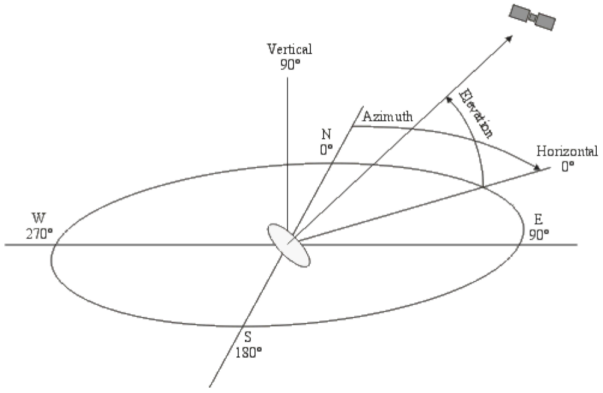

Figura 4: Coordenadas ENU dos satélites

Comparando posteriormente a elevação de cada satélite com um ângulo padrão, é possível separar os satélites que entram na sub-constelação de satélites visíveis.

Seja o a posição do recetor denominada por (x_u, y_u, z_u), t_u o desvio do relógio do recetor relativamente ao relógio do sistema GPS, e ϵ_i o erro associado à medida da posição do recetor devido aos efeitos da ionosfera. A posição de um recetor é então determinada recorrendo ás pseudo-distâncias relativas a N satélites. As pseudo-distâncias são calculadas a partir da equação 12. As estimativas das pseudo-distâncias são calculadas utilizando a mesma fórmula, mas com os valores estimados para a posição do recetor $(\hat{x_u}, \hat{y_u}, \hat{z_u})$ e para o erro do relógio $\hat{t_u}$.

$$\rho_N = \sqrt{(X_N - x_u)^2 + (Y_N - y_u)^2 + (Z_N - z_u)^2} + ctu + \epsilon_N \tag{12}$$

Uma vez que o erro do relógio e a posição do recetor são desconhecidos, estes podem ser aproximados pela sua estimativa somada de uma componente incremental, da forma:

$$\begin{bmatrix} x_u \\ y_u \\ z_u \\ t_u \end{bmatrix} = \begin{bmatrix} \hat{x_u} + \Delta x_u \\ \hat{y_u} + \Delta y_u \\ \hat{z_u} + \Delta z_u \\ \hat{t_u} + \Delta t_u \end{bmatrix} \tag{13}$$

O modelo incremental para as pseudo-distâncias é dado por:

$$\Delta\rho_i = \frac{\partial\rho_i}{\partial x_u}\Delta x_u + \frac{\partial\rho_i}{\partial y_u}\Delta y_u + \frac{\partial\rho_i}{\partial z_u}\Delta z_u \tag{14}$$

em que $\frac{\partial\rho_i}{\partial x_u} = \frac{x_u - X_i}{\hat{r_i}}$, $\frac{\partial\rho_i}{\partial y_u} = \frac{y_u - Y_i}{\hat{r_i}}$, e $\frac{\partial\rho_i}{\partial z_u} = \frac{z_u - Z_i}{\hat{r_i}}$, e a distância esperada entre satélite e recetor $\hat{r_i} = \sqrt{(x_i - x_u)^2 + (y_i - y_u)^2 + (z_i - z_u)^2}$.

Na forma matricial tem-se $\Delta\rho = G\Delta v$. Para N=4 tem-se o seguinte sistema:

$$\begin{bmatrix}\Delta\rho_1\\ \Delta\rho_2\\ \Delta\rho_3\\ \Delta\rho_4\end{bmatrix} = \begin{bmatrix}a_{x1} & a_{y1} & a_{z1} & 1\\ a_{x2} & a_{y2} & a_{z2} & 1\\ a_{x3} & a_{y3} & a_{z3} & 1\\ a_{x4} & a_{y4} & a_{z4} & 1\end{bmatrix}\begin{bmatrix}\Delta x_u\\ \Delta y_u\\ \Delta z_u\\ \Delta t_u\end{bmatrix} \tag{15}$$

em que $\Delta\rho_i = \hat{\rho}_i - \rho_i$ e $a_{xi} = \frac{\partial\rho_i}{\partial x_i}$. Ao considerar a geometria dos satélites fixa, esta relação traduz-se numa relação linear entre os erros de pseudo-distâncias e os erros de posicionamento e desvio do relógio do recetor. Os erros das pseudo-distâncias são considerados variáveis Gaussianas de média nula, pelo que assumindo uma geometria fixa Δv é também um vetor gaussiano de média nula cuja matriz de covariância é definida por:

$$cov(\Delta v) = \sigma^2_{UERE}(G^T G)^{-1} \tag{16}$$

em que $(G^T G)^{-1}$ é uma matriz representada como:

$$(G^T G)^{-1} = \begin{bmatrix}h_{11} & h_{12} & h_{13} & h_{14}\\ h_{21} & h_{22} & h_{23} & h_{24}\\ h_{31} & h_{32} & h_{33} & h_{34}\\ h_{41} & h_{42} & h_{43} & h_{44}\end{bmatrix} \tag{17}$$

O PDOP (*Position Dilution of Precision*) é então definido como:

$$PDOP = \sqrt{h_{11} + h22 + h33} \tag{18}$$

O PDOP é um indicador de qualidade e de precisão da posição num ambiente tridimensional. Representa uma amplificação do erro de posição, e por isso deve ser tão baixo quanto possível.

2.4 Método dos Mínimos Quadrados

O método dos mínimos quadrados é um método que faz uma estimativa procurando minimizar a soma dos quadrados dos resíduos. Como resíduo entenda-se a variação entre o valor estimado e o valor exato. É um método iterativo de rápida convergência em que, em cada iteração a posição calculada é feita com base numa estimativa anterior.

O método dos mínimos quadrados aplicado foi o seguinte:

1. A partir das posições conhecidas dos satélites e do recetor, são calculadas as pseudo-distâncias entre o recetor e os satélite.

2. Fornece-se uma estimativa inicial, de forma a que o algoritmo tenha um ponto de partida.

3. É determinada a matriz G.

4. São calculadas as pseudo-distâncias estimadas e determinam-se os erros de pseudo-distâncias.

5. Determina-se o erro de posição e a estimação.

6. Repete-se o algoritmo a partir do passo 2, utilizando agora as estimativas calculadas em cada iteração no passo 5.

Em caso de fim do tempo de simulação, ou de não haver o número de satélites necessário para a solução da equação de navegação (4), o algoritmo termina.

Os erros ionosféricos ($\rho_{iono} = \frac{10}{sen\epsilon}$) podem ou não ser considerados no algoritmo. No entanto, para cada medição de pseudo-distâncias há a adição de ruído Gaussiano (n) de média nula e variância inserida pelo utilizador. As pseudo-distâncias são dadas por:

$$\rho_i = \sqrt{(X_i - x_u)^2 + (Y_i - y_u)^2 + (Z_i - z_u)^2} + \rho_{iono} + n \tag{19}$$

em que (X_i, Y_i, Z_i) com $i = 1, ..., N$, são as coordenadas do satélite e (x_u, y_u, z_u) são as coordenadas do recetor.

A matriz G é então dada por:

$$G = \begin{bmatrix} -\frac{X_1 - \hat{x_u}}{\rho_1} & -\frac{Y_1 - \hat{y_u}}{\rho_1} & -\frac{Z_1 - \hat{z_u}}{\rho_1} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ -\frac{X_i - \hat{x_u}}{\rho_i} & -\frac{Y_i - \hat{y_u}}{\rho_i} & -\frac{Z_i - \hat{z_u}}{\rho_i} & 1 \end{bmatrix} \tag{20}$$

As pseudo-distâncias estimadas são dadas por:

$$\hat{\rho_i} = \sqrt{(X_i - \hat{x_u})^2 + (Y_i - \hat{y_u})^2 + (Z_i - \hat{z_u})^2} \tag{21}$$

em que $(\hat{x_u}, \hat{y_u}, \hat{z_u})$ são as coordenadas estimadas do recetor.

É possível então calcular o erro das pseudo-distâncias e de posição (ΔP):

$$\Delta\rho = \rho_i - \hat{\rho_i} \tag{22}$$

$$\Delta P = (G^T G)^{-1} \Delta\rho \tag{23}$$

Assim, a nova estimativa de posição do recetor é:

$$\hat{P}(k+1) = \hat{P}(k) + \Delta P \tag{24}$$

em que $\hat{P}(k)$ é a posição estimada anterior.

3 Algoritmo Implementado

Inicialmente, o programa pede ao utilizador para definir valores para a data, hora, ângulo de máscara de referência, e posição inicial do recetor (latitude, longitude e altitude) pretendidas para a simulação. De seguida, e com base na informação do almanaque YUMA utilizado, é calculada a posição de todos os satélites. As coordenadas iniciais do recetor são então convertidas para coordenadas ECEF, que permite calcular o número e a posição dos satélites visíveis da posição do recetor. É então dada a informação do número de satélites visíveis ao utilizador, e é pedido o número de satélites pretendidos para a simulação, que deverá ser superior a 4 e inferior ao número de satélites visíveis. Com a informação do número de satélites a utilizar pretendidos, são calculadas as posições e os ID's dos satélites que minimizam o PDOP, que serão os utilizados ao longo do percurso. Posto isto, é pedido ao utilizador para indicar a variância pretendida para o ruído, e se pretende considerar os erros ionosféricos. São medidas então as pseudo-distâncias entre os satélites que minimizam o PDOP e o recetor, e é calculada uma estimação para a posição inicial do recetor a partir do método de mínimos quadrados, tendo como inicio o centro geográfico de Portugal (37ºN, 7ºW, altitude 500m).

A frequência de amostragem é $1Hz$, pelo que a posição do recetor é atualizada a cada $1s$. Para cada instante de tempo ao longo de todo o trajeto é calculada a posição do recetor, sendo que a origem do referêncial é considerada a posição inicial do recetor. O movimento no percurso AB é considerado movimento uniformemente acelerado, no percurso BC é considerado movimento circular uniforme, e no percurso CD e DE é considerado movimento retilíneo uniforme. As máscaras de referência inseridas pelo utilizador são utilizadas para todos os percursos excepto para o DE, em que é utilizada uma máscara para que o número de satélites seja 4. As coordenadas de cada posição do recetor ao longo do percurso são convertidas para coordenadas ECEF e para LLA (longitude, latitude, altitude).

Uma vez que ao longo do movimento do recetor a posição dos satélites relativamente a este se altera, são calculadas as posições dos satélites para cada posição amostrada do recetor, com a restrição da máscara. Em cada posição o programa verifica que há o número de satélites pretendidos, inserido pelo utilizador, sendo que em caso de não haver, termina. No percurso DE o programa utiliza apenas os 4 satélites de maior elevação.

Havendo o número de satélites pretendidos disponíveis, são medidas as pseudo-distâncias entre os satélites e o recetor. Posteriormente, são então calculadas as estimativas para a posição do recetor em cada posição, através do método de mínimos quadrados.

Com as coordenadas da posição exata e da estimativa de mínimos quadrados do recetor, é calculado o erro absoluto e quadrático de posição do recetor. Em cada uma das 100 simulações é armazenado o erro quadrático médio de cada percurso, que possibilita a realização de um gráfico para a avaliação da precisão da estimação.

Por fim, cada posição do recetor é convertida para coordenadas ENU, que possibilita fazer uma estimação para a velocidade através da amostra de posições adjacentes. É calculado também o erro absoluto e quadrático para a velocidade. O erro quadrático da velocidade em cada posição é utilizado para gerar um gráfico que permite avaliar a precisão da estimação da velocidade.

4 Resultados Obtidos

4.1 Posição dos Satélites

Na simulação executada foi utilizado como data e hora de referência 28/11/2009, às 19:55:01, com um ângulo de máscara de referência de 10^{o} e sendo posição inicial exata do recetor:

- Latitude = 40^{o}N
- Longitude = 9^{o}W
- Altitude = 2000m

A posição dos 31 satélites do almanaque estão indicadas em termos de azimute e elevação na figura 5.

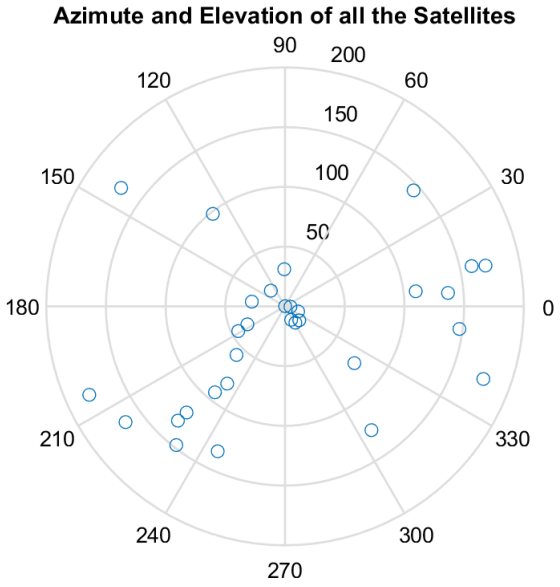

Figura 5: Posição inicial de todos os satélites disponíveis

Dada a restrição pelo ângulo de máscara, os satélites visíveis da posição inicial do recetor não correspondem aos disponíveis. Assim, a figura 6 representa a posição em termos de azimute e elevação dos satélites visíveis.

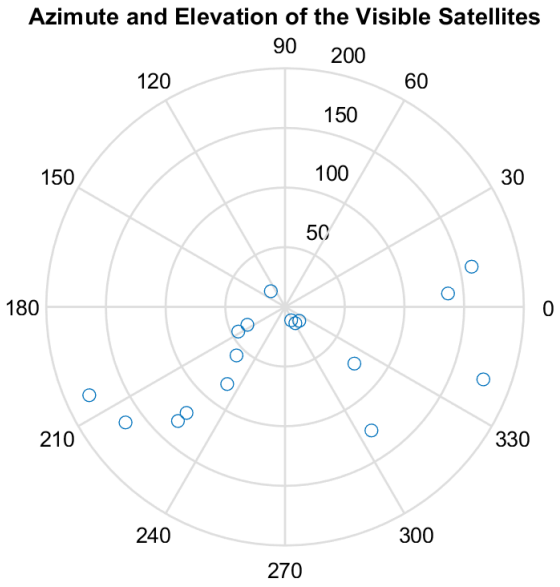

Figura 6: Posição inicial dos satélites visíveis

O programa emitiu então para o utilizador a mensagem de que estavam 17 satélites disponíveis. O número de satélites escolhidos para serem utilizados foi N=8. A escolha dos satélites utilizados foi feita com base nos que têm menor PDOP. Assim, a constelação dos satélites utilizados, que minimizam o PDOP está representada na figura 7. O PDOP mínimo obtido foi de 1,6180.

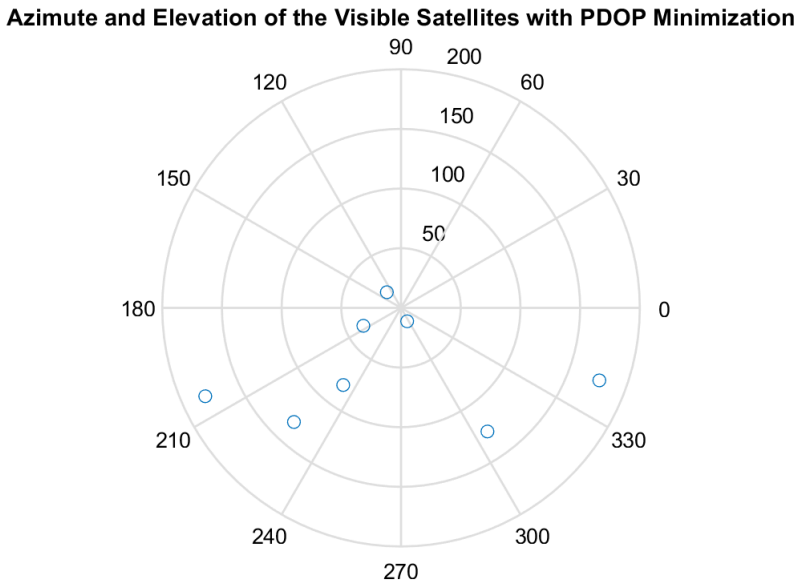

Figura 7: Posição inicial dos 8 satélites utilizados que minimizam o PDOP

A simulação foi repetida com os mesmos valores introduzidos, excepto o número de satélites

pretendido que foi aumentado para N=13. Com esta configuração, a constelação de satélites que minimiza o PDOP está representada na figura 8. Para este caso, verificou-se que o PDOP mínimo desceu para 1,4388.

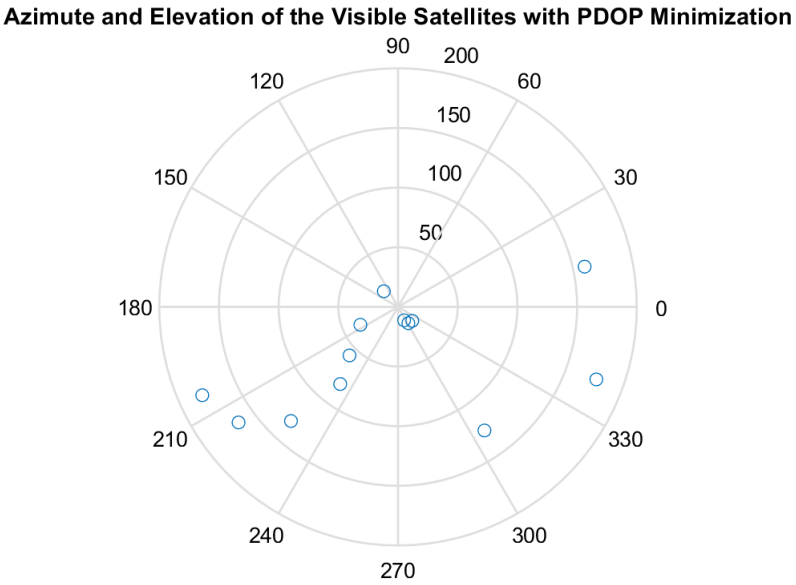

Figura 8: Posição dos 13 satélites utilizados que minimizam o PDOP

4.2 Implementação do Método de Mínimos Quadrados

Na simulação, para o cálculo das pseudo-distâncias foi definida a variância do ruído como $\sigma^2_{ruido} = 15$.

4.2.1 Com consideração dos erros ionosféricos

A contribuição dos erros ionosféricos para o cálculo das pseudo-distâncias é $\rho_{iono} = \frac{10}{sen\epsilon}$.

As posições exatas e estimadas, bem como os erros absolutos de posição e velocidade ao longo do percurso estão representados na figura 9.

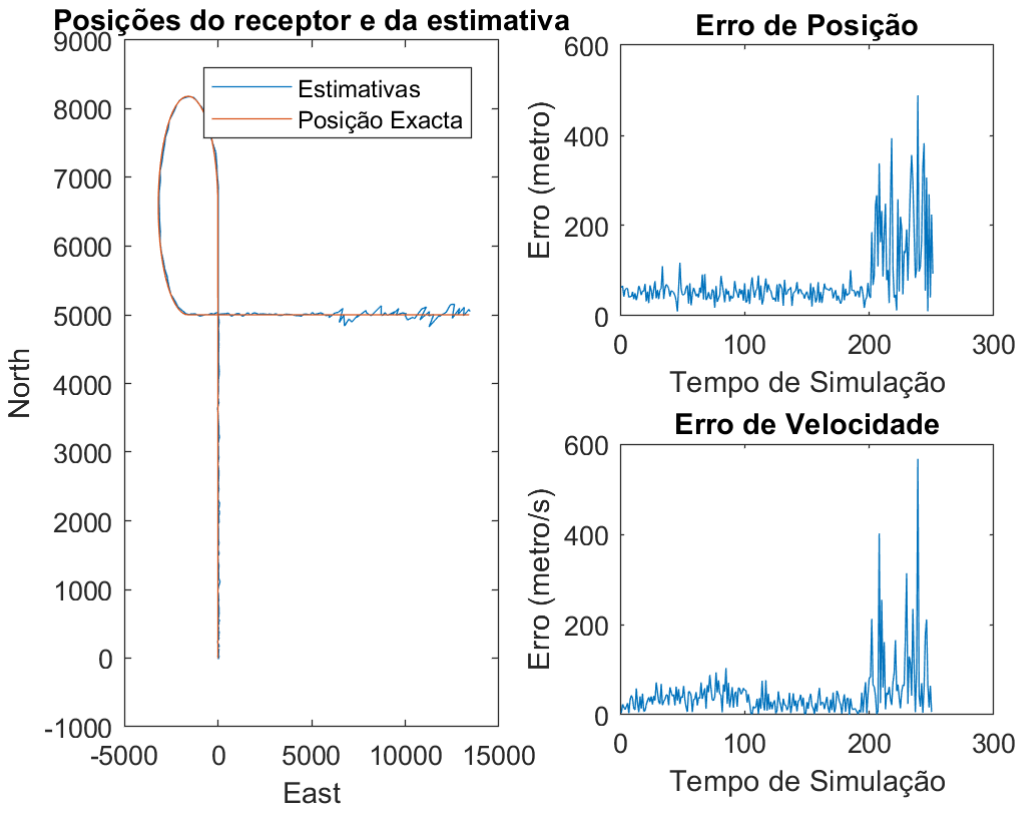

Figura 9: Posição exata e estimada e erros absolutos de posição e velocidade considerando erros ionosféricos

O erro de velocidade apresentado é relativo à estimação da velocidade e a velocidade real, que estão representadas na figura 10.

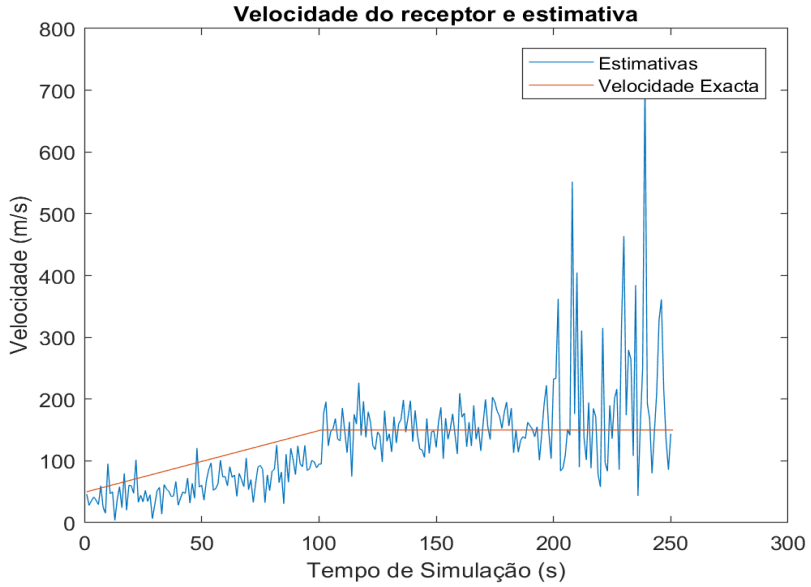

Figura 10: Velocidade exata e velocidade estimada

É facilmente visível o efeito do *Canyon Scenario* na estimação da posição do recetor e

consequentemente na velocidade. Verifica-se assim que a partir do momento em que o recetor começa o percurso com restrição aos 4 satélites de maior elevação, há um incremento dos erros de posição e velocidade.

O erro quadrático médio de um estimador é o quadrado do desvio entre o valor estimado e o verdadeiro. Foi armazenado o erro quadrático médio para cada percurso, para cada uma das 100 simulações. Assim, nas figuras seguintes, o erro quadrático médio e o erro de posição médio estão representados em função do número de simulações para o percurso AB, BC, CD e DE, respetivamente.

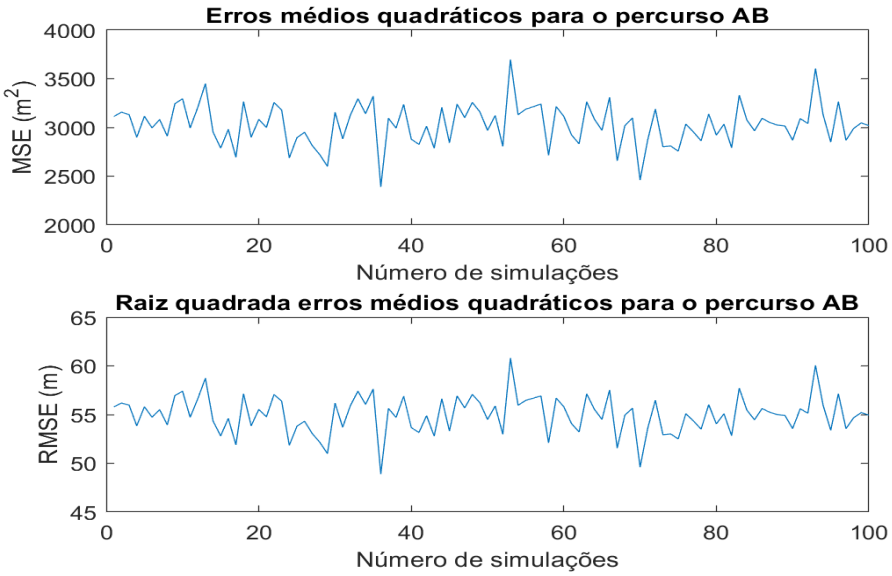

Figura 11: Erro quadrático e de posição médios para o troço AB

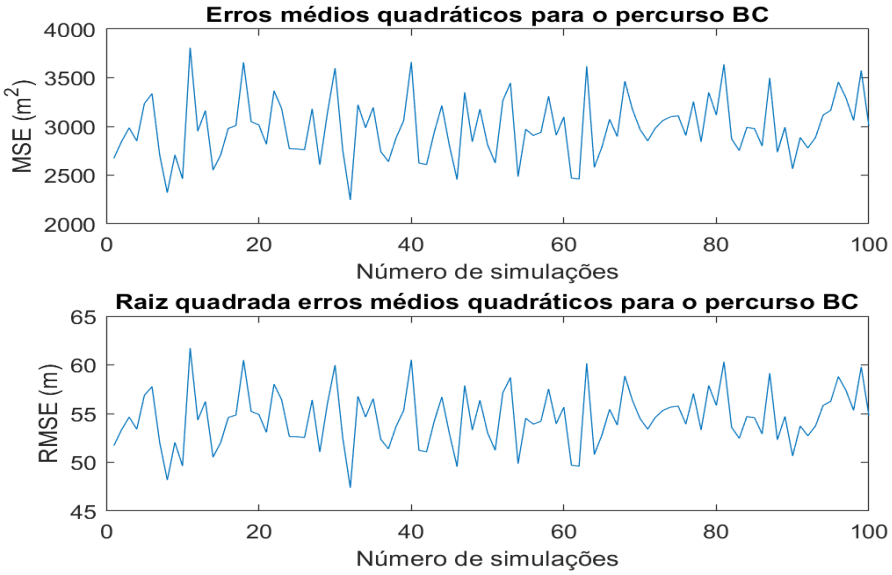

Figura 12: Erro quadrático e de posição médios para o troço BC

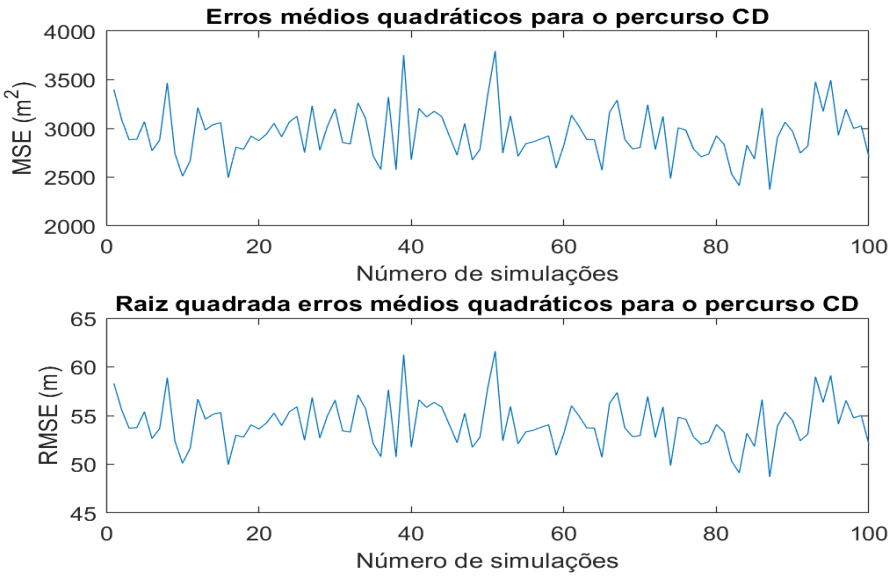

Figura 13: Erro quadrático e de posição médios para o troço CD

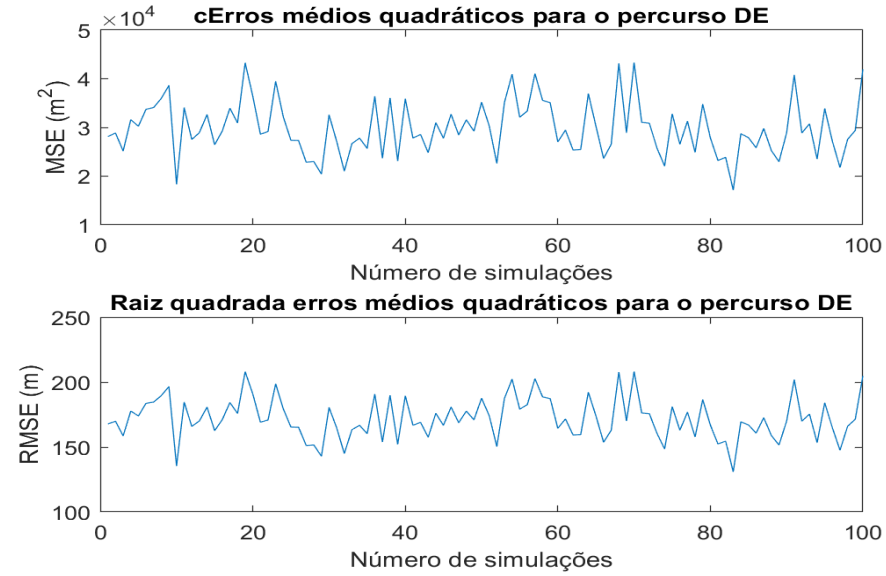

Figura 14: Erro quadrático e de posição médios para o troço DE

É possível verificar que os valores dos erros quadráticos médios e dos erros de posição são elevados, indicando um estimador nao muito preciso. Um dos principais fatores é o facto de em cada iteração o método dos mínimos quadrados passar muito pouca informação à estimativa seguinte. É possível também verificar que para o percurso DE os valores dos erros quadráticos médios são cerca de 10 vezes superiores, sendo os erros de posição cerca de 4 vezes superiores, relativamente aos restantes percursos. Novamente, esta diferença deve-se ao facto de apenas serem utilizados os 4 satélites de maior elevação no percurso DE.

4.2.2 Sem consideração dos erros ionosféricos

Fazendo novamente 100 simulações, mas ignorando os erros ionosféricos (indicado pelo utilizador), as posições exatas e estimadas, bem como os erros absolutos de posição e velocidade estão representados na figura 15.

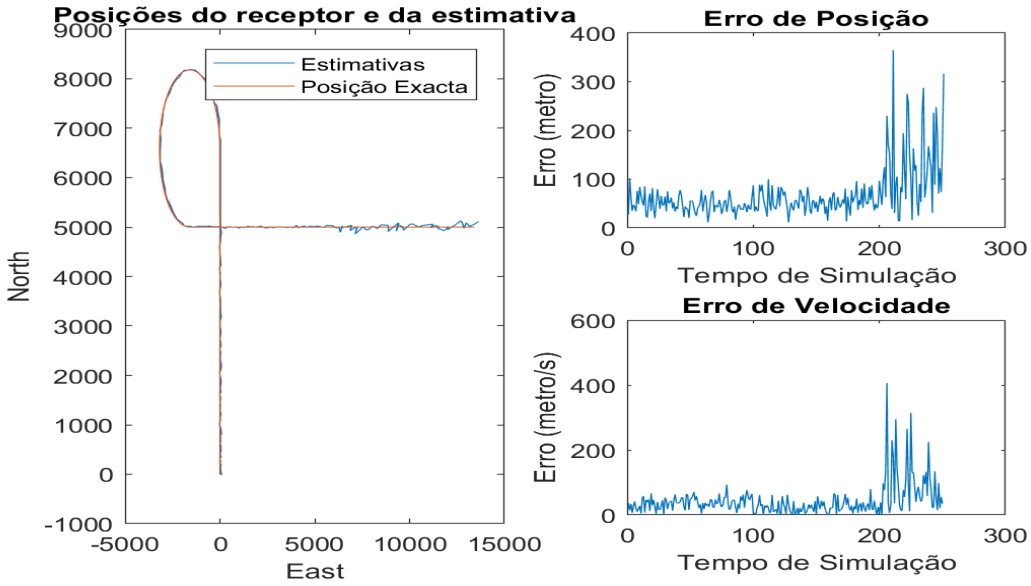

Figura 15: Posição exata e estimada e erros absolutos de posição e velocidade sem erros ionosféricos

O erro de velocidade é referente às grandezas representadas na figura 16.

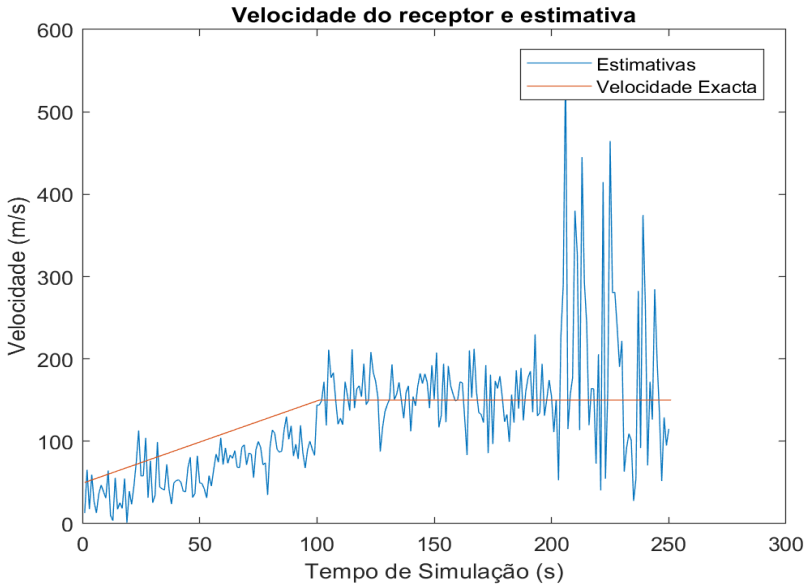

Figura 16: Velocidade exata e estimada para o percurso sem erros ionosféricos

Comparando as figuras 15 e 16 com as figuras 9 e 10, respetivamente, é possível verificar que os erros de posição e de velocidade diminuem.

Os gráficos do erro quadrático médio e do erro de posição para os trajetos AB, BC, CD e DE encontram-se representados nas figuras seguintes, respetivamente.

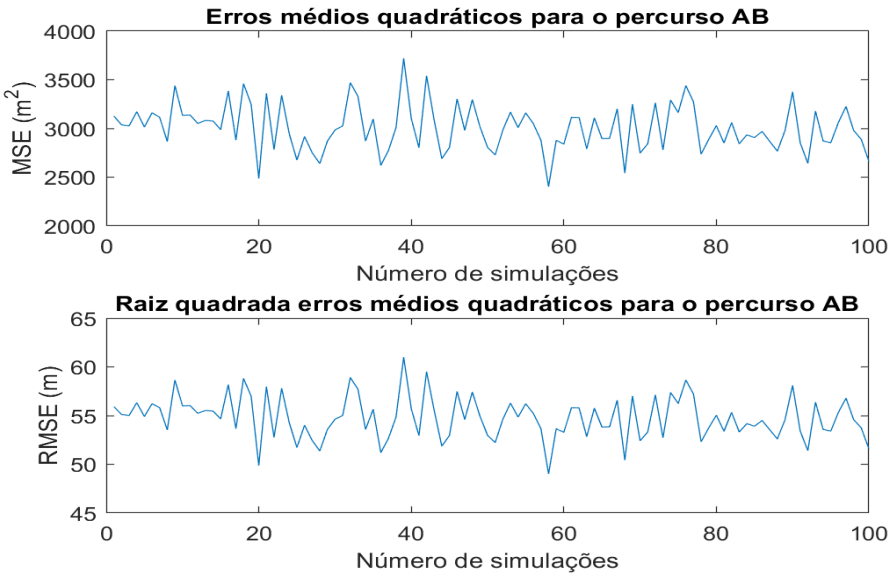

Figura 17: Erro quadrático e de posição médios para o troço AB sem erros ionosféricos

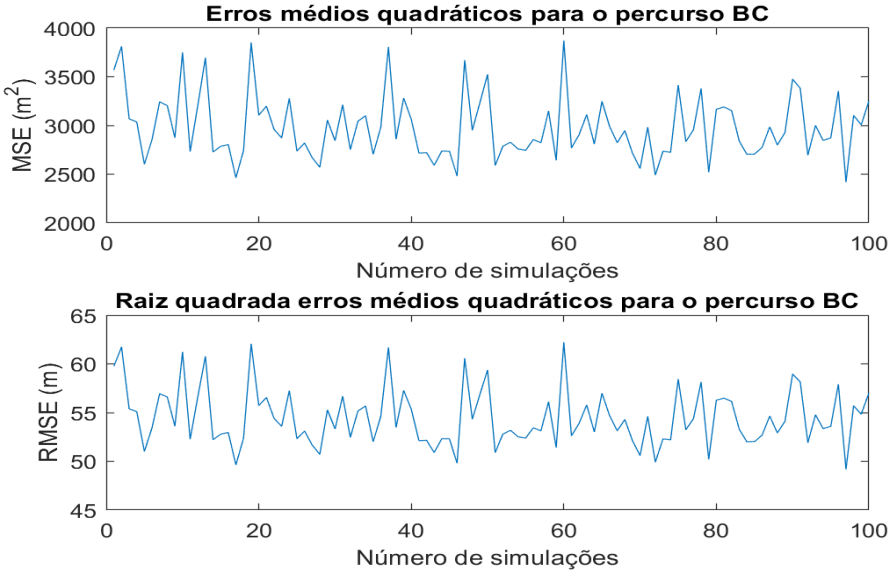

Figura 18: Erro quadrático e de posição médios para o troço BC sem erros ionosféricos

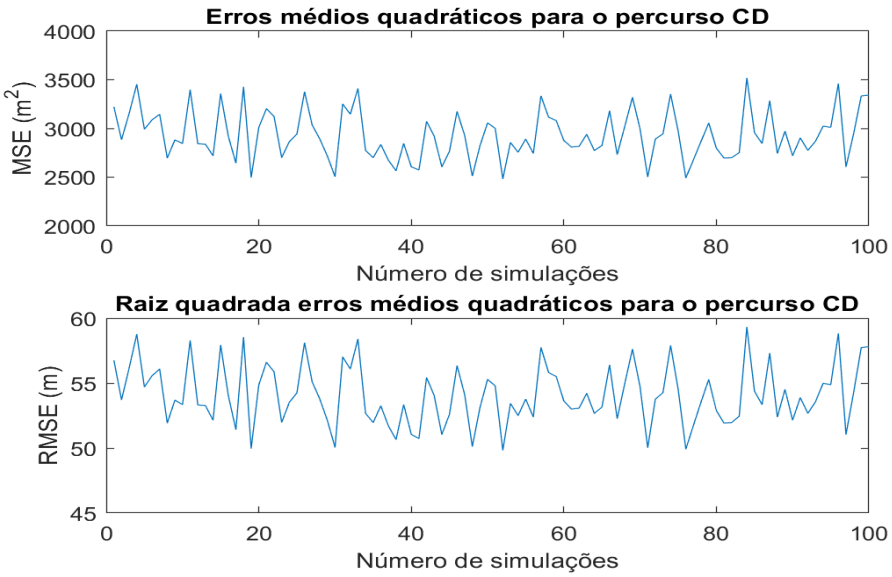

Figura 19: Erro quadrático e de posição médios para o troço CD sem erros ionosféricos

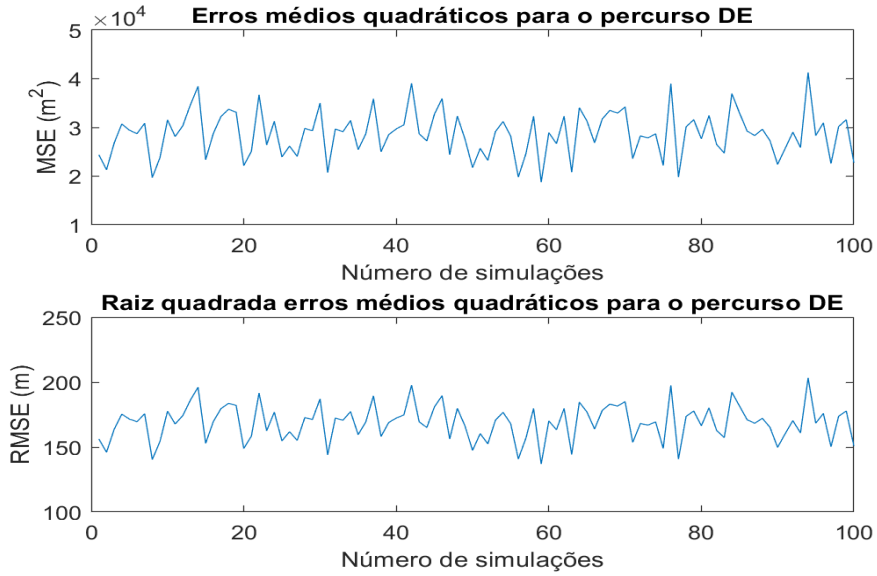

Figura 20: Erro quadrático e de posição médios para o troço DE sem erros ionosféricos

4.2.3 Com erros ionosféricos e sem *Canyon Scenario*

As 100 simulações foram feitas novamente, mas desta vez considerando que os satélites disponíveis ao longo de todo o percurso eram os mesmos, isto é, o percurso DE deixa de representar uma trajetória de *Canyon Scenario*.

As posições exatas e estimadas, bem como os erros absolutos de posição e velocidade estão representados na figura 21.

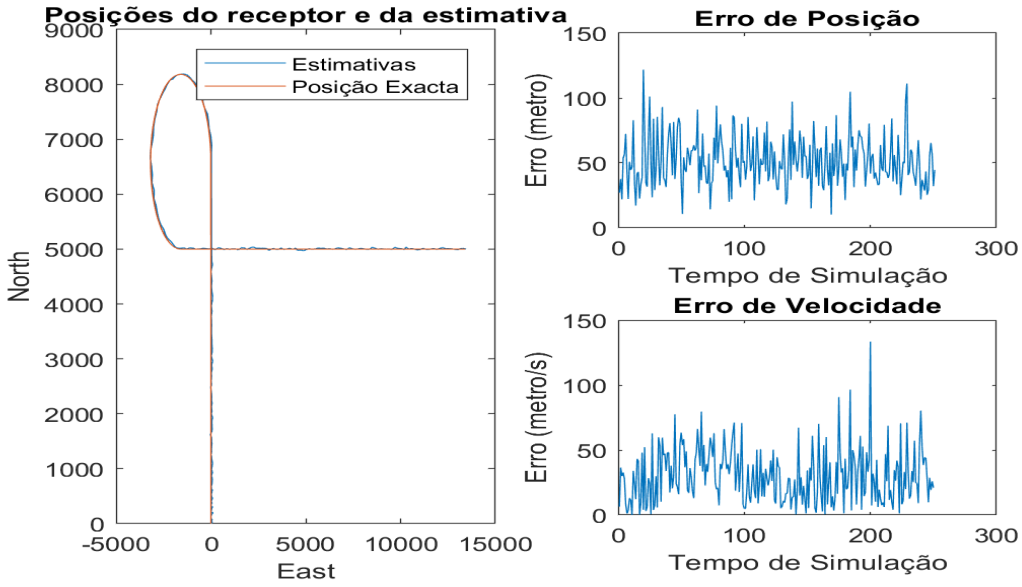

Figura 21: Posição exata e estimada e erros absolutos de posição e velocidade sem *Canyon Scenario* no percurso DE

O erro de velocidade apresentado é então relativo à velocidade real e à velocidade estimada, que estão representadas na figura 22.

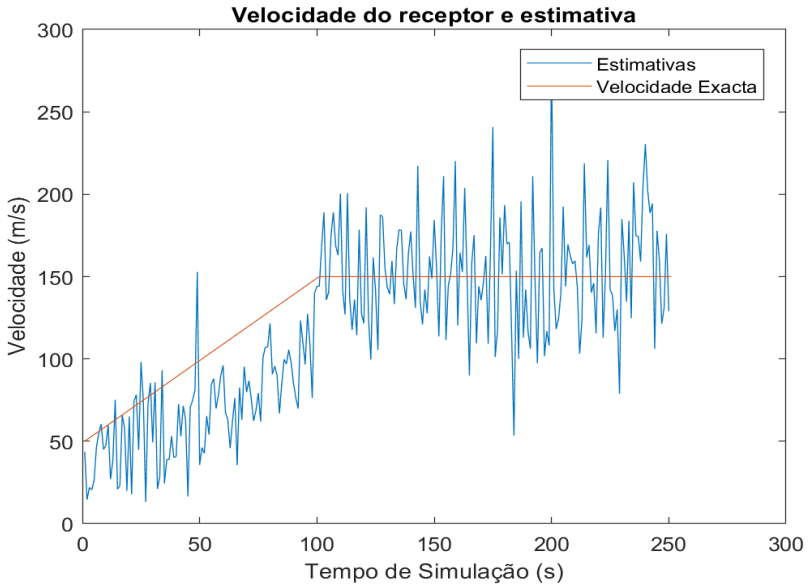

Figura 22: Velocidade exata e estimada para o percurso sem *Canyon Scenario*

Verifica-se um grande desvio entre os erros de posição e velocidade da figura 21 e da figura 9 relativos à trajetória DE. Uma vez que não há restrições neste percurso, o erro de velocidade e de posição evolui da mesma forma que evolui nos outros troços.

De seguida, o erro quadrático médio e o erro de posição médio estão representados em função do número de simulações para o percurso AB, BC, CD e DE, respetivamente.

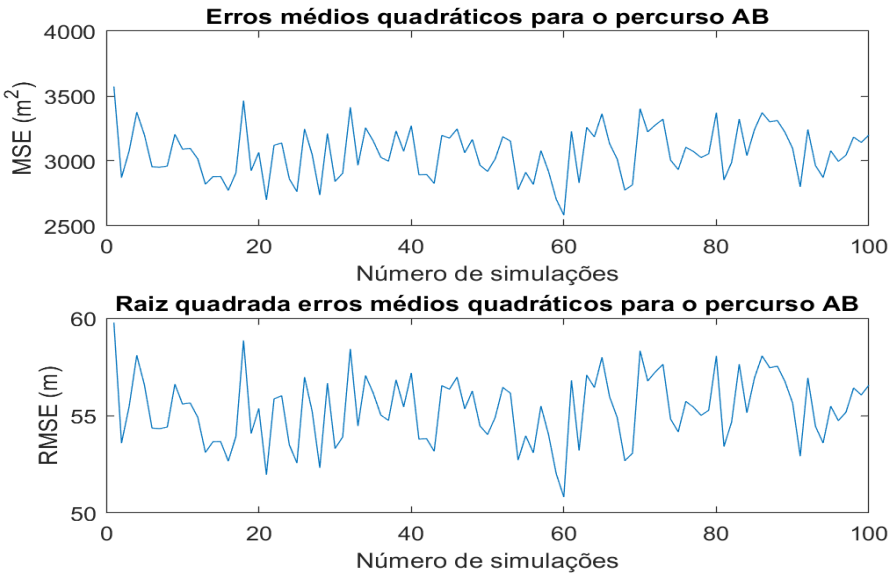

Figura 23: Erro quadrático e de posição médios para o troço AB

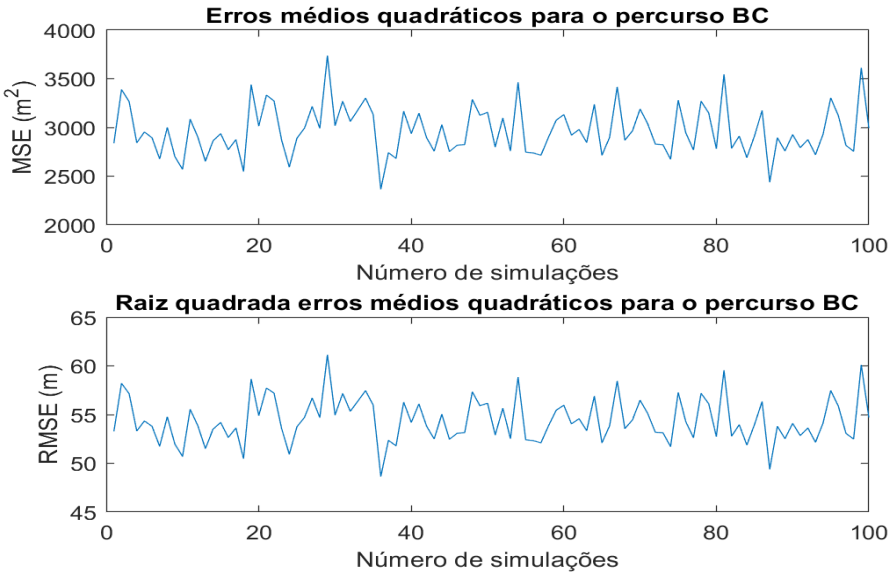

Figura 24: Erro quadrático e de posição médios para o troço BC

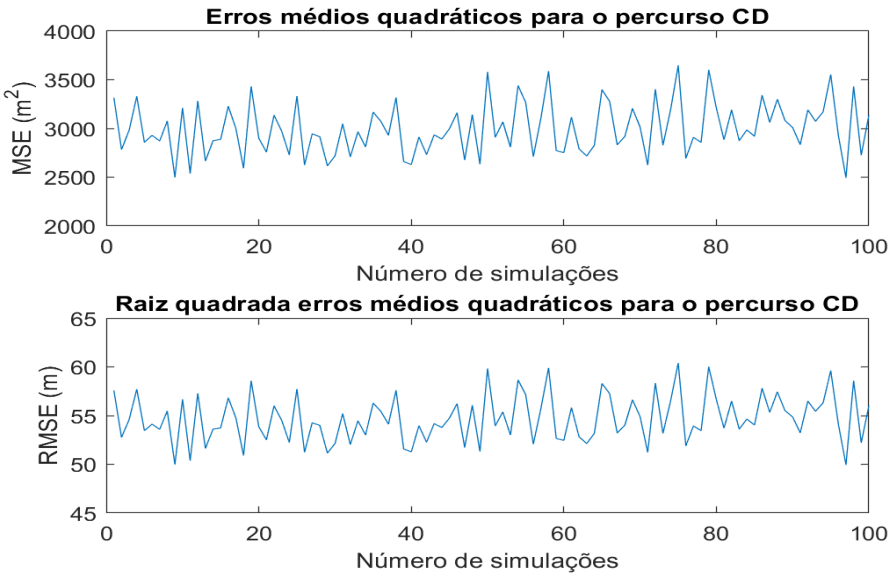

Figura 25: Erro quadrático e de posição médios para o troço CD

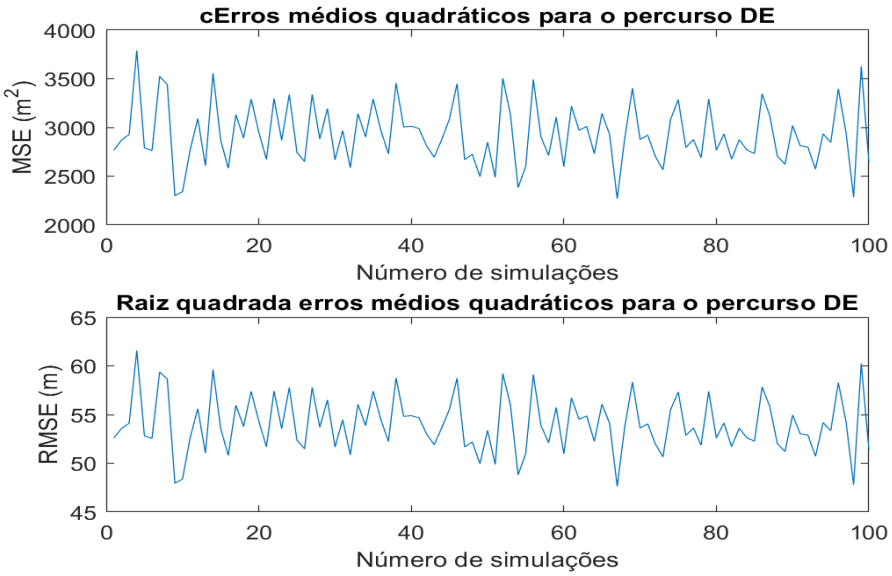

Figura 26: Erro quadrático e de posição médios para o troço DE

Verifica-se que os erros quadrático médio e de posição nos trajetos AB, BC, e CD se mantem dentro da mesma gama de valores. No entanto, comparando a figura 26 com a figura 14 verifica-se uma grande diminuição dos erros aquando da simulação sem *Canyon Scenario*, uma vez que assim o recetor escolhe utilizar os satélites que minimizam o PDOP. Assim, conclui-se que o decréscimo do número de satélites tem um elevado impacto na estimação da posição e da velocidade, e que a escolha dos satélites com a minimização do PDOP é de elevada importancia nos sistemas GPS.

5 Conclusões

O comportamento do método dos mínimos quadrados foi comprovado na resolução da equação de GPS para o cálculo da posição de um receptor. Foi possível verificar que este método apresenta resultados mais satisfatórios quanto maior for o número de satélites utilizados. FOi estudada a evolução do erro de posição e de velocidade num cenário em que inicialmente se usam 8 satélites que minimizam o PDOP e em que a parte final do percurso representa um *Canyon Scenario*, em que apenas os 4 satélites mais elevados estão disponíveis. Verificou-se que o erro de posição passa de cerca de 50m, aquando a utilização dos 8 satélites, para cerca de 200m com apenas 4. Assim, é realçada a importância de encontrar a combinação de satélites que melhor minimizar o PDOP.

Na análise do caso em que não se consideram erros ionosféricos foi possível concluir que estes contribuem para um aumento significativo do erro de posição e de velocidade. Assim, a minimização destes erros numa estação terrestre é importante para a navegação GPS.

Uma das desvantagens do método dos mínimos quadrados é o facto de se passar pouca informação entre iterações consecutivas. A utilização de filtros de Kalman permite melhores estimativas, uma vez que é tomado em consideração o movimento do recetor e o modelo de relógio.

Bibliografía

- Fernando Nunes, "Air Trafic Control Systems", DEEC-IST, 2018

ANEXO 1 - Código Matlab

Código principal

```
clc;

%Pr aloca de variveis
erro_pos=zeros(251,1);
erro_quadratico=zeros(100,251);
est_enu=zeros(251,3);
vel_est=zeros(251,1);
erro_vel=zeros(252,1);
erro_quad_vel=zeros(100,251);

mse_AB=zeros(1,100);
mse_BC=zeros(1,100);
mse_CD=zeros(1,100);
mse_DE=zeros(1,100);
mse_vel=zeros(1,100);

%% Inicializa de variveis
    c = 3e8;
    ano = 0;
    mes = 0;
    dia = 0;
    hora = 0;
    min = 0;
    sec = 0;
    mask = 10;
%% Abre o ficheiro que contm o almanaque
    afile = fopen('almanaque.txt','r');

    if afile == -1
        disp('Ficheiro no encontrado')
        disp(' ')
        return
    end

    A = importdata('almanaque.txt',':');
    [n,m] = size(A.data);
    numero_sat = (n+1)/14; %Nmero de satlites

    [id,square_A,e0,toa,alfa,rate_of_right_ascend,RAAN,arg_perigeu,mean_anom,af0,af1] = read_almanaque (A);

%% Data e hora de transmisso
fprintf('\n--------------------------\n\n');
fprintf('Insira a data exacta de GPS superior a 6 de Janeiro de 2008:\n\n');
test=0;

while (test ~= 1)

    data_s = input('Data em formato (DD/MM/YYYY): ', 's');
    test = isValidDate(data_s); %d = Enter date in 'DD/MM/YYYY' format

    if test == 0
        disp('Introduza novamente a Data no formato referido.')
    end

end
test=0;
while (test ~= 1)
    disp('');
    disp('00 <Hour< 23  00<Minutes<60 0<Seconds<60');
    hour_s = input('Hora em formato (HH:MM:SS): ', 's');
    test = isValidhour(hour_s);
    if test == 0
        disp('Introduza novamente a hora no formato referido')
    end
end
test=0;
```

```
while (test ~= 1)
    disp('');
    disp('introduza um angulo de mascara minimo de referencia');
    mask_s = input('Angulo de mascara em graus:', 's');
    mask = str2double(mask_s);
    if isnan(mask) == 1
        disp('Introduza novamente uma valor valido')
    else
        test = 1;
    end
end

ano= str2num(data_s(7:end));            %#ok<ST2NM>
mes= str2num(data_s(4:5));              %#ok<ST2NM>
dia= str2num (data_s (1:2));            %#ok<ST2NM>
hora= str2num (hour_s(1:2));            %#ok<ST2NM>
min= str2num (hour_s(4:5));             %#ok<ST2NM>
sec= str2num (hour_s(7:8));             %#ok<ST2NM>

t_st1 = 24*60*60*datenum(ano, mes, dia, hora, min, sec); %Tempo inicial de transmisso
t_sim = 0;


%% C lculo  da  p o s i  inicial dos satlites em ECEF
sat_matrix = calc_sat_pos(square_A,mean_anom,e0,arg_perigeu,RAAN,rate_of_right_ascend,alfa,t_sim,t_st1, numero_sat,id)
    ;

%% Coordenadas exatas do recetor em ECEF

% Coordenadas do recetor
% latitude = 40;
% longitude = -9;
% altitude = 2000;

fprintf('\n--------------------------');
fprintf('\n\nInsira a  p o s i  exacta do receptor:\n\n');
test=0;
while (test ~= 1)
        lat_recp_str = input('Latitude formato: DD-MM-SS-(N or S): ', 's');
        [test, latitude] = isValidcoordinates(lat_recp_str);
end
test=0;
while (test ~= 1)
        long_recp_str = input('Longitude formato: DD-MM-SS-(W or E): ', 's');
        [test, longitude] = isValidcoordinates(long_recp_str);
end
test=1;
while (test ~= 0)
        altitude_str = input('Altitude do receptor: ', 's');
        altitude = str2double(altitude_str);
        test = isnan(altitude) + imag(altitude);
end
test=1;

fprintf('Latitude (Graus):          %0.2f \n', latitude);
fprintf('Longitude (Graus):         %0.2f \n', longitude);
fprintf('Altitude (Graus):          %0.1f m\n', altitude);
fprintf('\nPrima uma tecla para continuar\n');
pause;

coordinates_receptor_e = lla2ecef([latitude longitude altitude]);
coord_recep = (coordinates_receptor_e);


%% C lculo  da  e l e v a  e do azimute dos satlites em  r e l a    p o s i  inicial do recetor

 [sat_E , sat_N , sat_U] = ecef2enu(sat_matrix(:,1),sat_matrix(:,2),sat_matrix(:,3),coord_recep(1), coord_recep(2) ,
     coord_recep(3) , referenceEllipsoid('wgs84'));
 sat_elev_azii = elev_azi(sat_E , sat_N , sat_U);

%% calculo dos satelites visiveis a partir da  p o s i  do receptor

```

```
[nr_sat_mask, sat_mask_elev_azii, sat_mask_ecef] = sat_view(sat_matrix, sat_elev_azii, mask);

% Clculo dos ID's dos satlites que minimizam o PDOP
% sat_dim_pret=4; % Dimensao pretendida da sub constela

    verif=1;
    while(verif == 1)
        verif = 0;
        disp('');
        fprintf('Existem %i satlites visveis.', nr_sat_mask);
        disp('Quantos satlites pretende utilizar? (Mnimo 4)');

        sat_dim_pret = input('','s');
        sat_dim_pret = str2double(sat_dim_pret);
        disp('');

        if isnan(sat_dim_pret) == 1 || (sat_dim_pret > nr_sat_mask && sat_dim_pret < 4)
            disp ('Valor invlido. Insira um n de satlites vlido.');
            verif = 1;
        end
    end

[min_pdop , id_sat_min_pdop] = pdop_min(sat_mask_ecef, nr_sat_mask, coord_recep, sat_dim_pret);

%Constela dos satlites que minimizam o PDOP de dimenso 4
[sub_const_matrix] = sat_sub(sat_dim_pret, sat_mask_ecef, id_sat_min_pdop, nr_sat_mask);
sub_const_matrix(:,4) = id_sat_min_pdop(:,4);

% Elevacao e azimute dos sat escolhidos em rela ao receptor
 [sat_EE, sat_NN ,sat_UU] = ecef2enu(sub_const_matrix(:,1), sub_const_matrix(:,2), sub_const_matrix(:,3), coord_recep
      (1), coord_recep(2), coord_recep(3), referenceEllipsoid('wgs84'));
 [sub_const_matrix_elev_azi ] = elev_azi(sat_EE, sat_NN, sat_UU);


 %% Medi das pseudo-distncias

 coord_recep(4) = 0; % 4 a varivel  o tempo

 %Sele da varincia do rudo

    verif=1;
    while(verif == 1)
        verif=0;
        disp('Indique a varincia pretendida do rudo na medi das pseudo-distancias:');
        var_ruido = input('','s');
        var_ruido = str2double(var_ruido);

        if isnan(var_ruido) == 1
             disp ('Valor invlido. Insira um valor correcto');
             verif = 1;
        end
    end

    verif=1;
    while(verif == 1)
        verif = 0;
        disp('Pretende considerar os erros ionosfricos?');
        disp('1 - Sim');
        disp('0 - No');

        iono = input('','s');
        iono = str2double(iono);
        disp('');

        if isnan(iono) == 1 || (iono ~= 0 && iono ~=1)
            disp ('Valor invlido. Insira um valor correcto');
            verif = 1;
        end
    end

%Clculo das pseudo-distncias
[pseu_meas] = pseudoranges_calc(sub_const_matrix,coord_recep,sat_dim_pret,iono,var_ruido);
```

```
208
209
210 %% Estima inicial do recetor
211
212 %Estimativa inicial utilizada — Centro geogrfico de portugal
213 lat_cg_pt=37;
214 long_cg_pt=7;
215 alt_cg_pt=500;
216 est_ini=lla2ecef([lat_cg_pt long_cg_pt alt_cg_pt]);
217 est_ini(4) = c*4;
218
219 %Estima inicial do recetor
220 est_rec_ini = least_sq_alg(sub_const_matrix,est_ini,coord_recep,sat_dim_pret,var_ruido);
221 est_rec = est_rec_ini;
222
223 %% Percurso percorrido
224
225 %Clculo da posi do avio ao longo do percurso
226 percurso = path();
227
228 for z=1:100    %100 Simulaes
229 est_enu(size(percurso,1), 3)=0;   %Matriz para as estimativas
230 erro_pos(size(percurso,1),3)=0; %Matriz para os erros de posi
231
232 for k=1:size(percurso,1)
233
234     %Coordenadas EC EF e LLA do percurso realizado
235     [p_ecef(1), p_ecef(2), p_ecef(3)] = enu2ecef(percurso(k,1), percurso(k,2), percurso(k,3), latitude, longitude,
            altitude, referenceEllipsoid('wgs84'));
236     p_lla = ecef2lla([p_ecef(1) p_ecef(2) p_ecef(3)]);
237
238     %Posi dos satlites em ECEF para cada instante de simula
239     t_sim=k;
240     sat_matrix = calc_sat_pos(square_A,mean_anom,e0,arg_perigeu,RAAN,rate_of_right_ascend,alfa,t_sim,t_st1, numero_sat
            ,id);
241
242     %Eleva e Azimute dos satlites para cada posi do recetor
243     [sat_E , sat_N , sat_U] = ecef2enu(sat_matrix(:,1),sat_matrix(:,2),sat_matrix(:,3), p_lla(1), p_lla(2) , p_lla(3)
            , referenceEllipsoid('wgs84'));
244     sat_elev_azi = elev_azi(sat_E , sat_N , sat_U);
245    %
246     %Clculo dos satlites visveis em cada instante com a restri da
247     %mscara
248     [nr_sat_mask, sat_mask_elev_azi, sat_mask_ecef] = sat_view(sat_matrix, sat_elev_azi, percurso(k,6));
249
250     %Check number of satelites available
251     if nr_sat_mask >= sat_dim_pret
252         [min_pdop , id_sat_min_pdop] = pdop_min( sat_mask_ecef, nr_sat_mask, p_ecef, sat_dim_pret);
253
254         %Coordenadas ECEF dos sat que minimizam o PDOP
255         [sub_const_matrix] = sat_sub(sat_dim_pret, sat_mask_ecef, id_sat_min_pdop, nr_sat_mask);
256     else
257         disp('N de satlites visiveis insuficientes. O programa terminou.')
258         break
259     end
260
261     %Introduz o ID dos satlites na 4 coluna da matriz
262     sub_const_matrix(:,4) = id_sat_min_pdop(:,4);
263
264     p_ecef(4)=est_rec_ini(4);
265     %Mede as pseudo—distncias entre o recetor e os satlites escolhidos
266     [pseu_meas] = pseudoranges_calc(sub_const_matrix, p_ecef, sat_dim_pret, iono, var_ruido);
267
268     %Clculo da posi do recetor atravs do alg. de Minimos Quadrados
269     %Garante que se for o percurso DE que n de sat = 4
270     if k>201
271         est_rec = least_sq_alg(sub_const_matrix, est_rec, p_ecef, 4, var_ruido);
272     else
273         est_rec = least_sq_alg(sub_const_matrix, est_rec, p_ecef, sat_dim_pret, var_ruido);
274     end
275
276     %Erro absoluto e quadrtico cometido com a estimativa
```

```
    erro_pos(k)=sqrt((est_rec(1)-p_ecef(1))^2+(est_rec(2)-p_ecef(2))^2+(est_rec(3)-p_ecef(3))^2);
    erro_quadratico(z,k) = erro_pos(k)^2;

    mse_AB(z)=mean(erro_quadratico(z,1:101));
    mse_BC(z)=mean(erro_quadratico(z,101:151));
    mse_CD(z)=mean(erro_quadratico(z,151:201));
    mse_DE(z)=mean(erro_quadratico(z,201:251));

    %Calculo das coord ENU das posies estimadas
    [est_E, est_N, est_U]=ecef2enu(est_rec(1), est_rec(2), est_rec(3), latitude, longitude, altitude,
         referenceEllipsoid('wgs84'));
    est_enu(k,:)=[est_E est_N est_U];
    est_enu(252,:)=[0 0 0];

    vel_est(k)=sqrt((est_enu(k,1)-est_enu(k+1,1))^2+(est_enu(k,2)-est_enu(k+1,2))^2);
    erro_vel(k)=sqrt((vel_est(k)-percurso(k,4))^2);
    erro_quad_vel(z,k)=erro_vel(k)^2;

    mse_vel(z)=mean(erro_quad_vel(z,:));
end

 est_enu=est_enu(1:end-1,:);
 vel_est=vel_est(1:end-1);
 vel_est(101)=vel_est(100);
 erro_vel=erro_vel(1:end-1);
 erro_vel(101)=erro_vel(100);

end
```

Funções utilizadas

```
%% read_almanaque.m
function [id,square_A,eccent,toa,alfa,rate_of_right_ascend,RAAN,arg_perigee,mean_anom,af0,af1] = read_almanaque (A)

    %Variveis auxiliares
    n = 0;
    k = 1;
    i = 1;
    verif = 1;

    [n,m] = size(A.data);

    r = (n+1)/14;

    %Inicializa dos parmetros contidos no almanaque
    id = (1/r);
    eccent = zeros(1,r);
    toa = zeros(1,r);
    alfa = zeros(1,r);
    rate_of_right_ascend = zeros(1,r);
    square_A = zeros(1,r);
    RAAN = zeros(1,r);
    arg_perigee = zeros(1,r);
    mean_anom = zeros(1,r);
    af0 = zeros(1,r);
    af1 = zeros(1,r);

    while(i<n)
            id(k) = A.data(i);
            i = i+2;
            eccent(k) = A.data(i);
            i=i+1;
            toa(k) = A.data(i);
            i=i+1;
            alfa(k) = A.data(i);
            i=i+1;
            rate_of_right_ascend(k) = A.data(i);
            i=i+1;
            square_A(k) = A.data(i);
```

```
            i=i+1;
            RAAN(k) = A.data(i);
            i=i+1;
            arg_perigee(k) = A.data(i);
            i=i+1;
            mean_anom(k) = A.data(i);
            i=i+1;
            af0(k) = A.data(i);
            i=i+1;
            af1(k) = A.data(i);
            i=i+3;

            k = k+1;
    end

end


%% Calc_sat_pos.m

% Fun  que d  a posi  actual dos satlites
function[sat_matrix] = calc_sat_pos(square_A,mean_anom,e0,arg_perigeu,RAAN,rate_of_right_ascend,alfa,t_sim,t_st1,r,id)

    t_st = t_st1 + t_sim;      %Tempo de transmisso do sinal
    t_oe = 24*60*60*datenum(2000, 1, 23, 0, 0, 0); %Tempo de referencia da ephemeris
    cons_grav = 3.986005e14; %ctg de gravidade da terra

    delta_t = t_st-t_oe;
    %Corre  do delta t
    if delta_t > 302000
        delta_t = delta_t - 604800;
    end

    %Matriz onde vai ser inserida a posi  dos satlites
    sat_matrix = zeros(r,3);

    %clculo da constela  de satlites - coordenadas ECEF
    for k=1:r


        n(k) = sqrt(cons_grav/square_A(k)^6);      %Mean motion
        mean_anom_aux = mean_anom(k) + n(k)*delta_t;  %Anomalia Mdia
        eccent_anom = Calc_EA(mean_anom_aux,e0(k));  %Excentricidade
    %Anomalia verdadeira
        true_anom = atan2(sqrt(1-e0(k)^2)*sin(eccent_anom),cos(eccent_anom)-e0(k));
        teta = true_anom + arg_perigeu(k);
    %Longitude do n  ascendente
        omega = RAAN(k) + rate_of_right_ascend(k)*delta_t - 2*pi/86164*t_st;
    %Raio da  ita
        orbit_rad = square_A(k)^2*(1-e0(k)*cos(eccent_anom));

        sat_matrix(k,1) = orbit_rad*cos(teta)*cos(omega)-orbit_rad*sin(teta)*sin(omega)*cos(alfa(k));
        sat_matrix(k,2) = orbit_rad*cos(teta)*sin(omega)+orbit_rad*sin(teta)*cos(omega)*cos(alfa(k));
        sat_matrix(k,3) = orbit_rad*sin(teta)*sin(alfa(k));
        sat_matrix(k,4) = id(1,k);

    end

end


%% Sat_view.m

% Determina quais os satelites que estao visiveis, ou seja, os que tem maior
%elevacao que o angulo de mascara

function [nr_sat_mask,sat_mask_elev_azi,sat_mask_ecef] = sat_view(sat_matrix, sat_elev_azi, mask)

i=1;
for k=1:size(sat_elev_azi,1)
    if sat_elev_azi(k,1)>= mask
        sat_mask_elev_azi(i,:) = sat_elev_azi(k,:);
```

```
        sat_mask_ecef(i,:) = sat_matrix(k,:);
        i=i+1;
    end
end

nr_sat_mask=size(sat_mask_elev_azi,1);

end


%% Sat_sub.m

function[sub_const_matrix] = sat_sub(dim_sub_const,sat_matrix,id_min,r)

        sub_const_matrix = zeros(dim_sub_const,3);

        for i=1:dim_sub_const
        for k=1:r
        if sat_matrix(k,4) == id_min(i)
           sub_const_matrix(i,1) = sat_matrix(k,1) ;
           sub_const_matrix(i,2) = sat_matrix(k,2);
           sub_const_matrix(i,3) = sat_matrix(k,3);
        end
        end
        end

end

%% least_sq_alg.m

function coord_finais = least_sq_alg(sub_const_matrix,est_ini,coord_recep,sat_dim_pret,var_ruido)

      pseu_meas = pseudoranges_calc(sub_const_matrix,coord_recep,sat_dim_pret,1,var_ruido);
      pseu_est = pseudoranges_calc(sub_const_matrix,est_ini,sat_dim_pret,0,var_ruido);
      dist = calc_dist(sub_const_matrix,est_ini,sat_dim_pret);

%Clculo da matriz G
        for k=1:sat_dim_pret
            MG(k,1) = -(sub_const_matrix(k,1)-coord_recep(1))/dist(k);
            MG(k,2) = -(sub_const_matrix(k,2)-coord_recep(2))/dist(k);
            MG(k,3) = -(sub_const_matrix(k,3)-coord_recep(3))/dist(k);
            MG(k,4) = 1;
        end
      G = MG;

      Ginv = inv(G'*G);
      delta_pseu = pseu_meas - pseu_est;
      delta_pos = (Ginv * G') * delta_pseu';
      delta_pos_i = delta_pos';
      est_ini = est_ini + delta_pos_i;

  coord_finais= est_ini;
end


%% pdop_min.m

% Calcula a constelacao de satelites que tem a menor PDOP
function[min_pdop, id_sat_min_pdop] = pdop_min(sat_vis, nr_sat, coordinates, dim_sub_const)

    % Inicializacao
    r = zeros(1,nr_sat);

    aux = nchoosek(1:nr_sat,dim_sub_const); % Corresponde a todas as combinacoes possiveis de satelites
    max = nchoosek(nr_sat,dim_sub_const); % Corresponde ao numero maximo de combinacoes

    G = zeros(dim_sub_const, 4);
    pdop = zeros(1,max);
    id_sat_min_pdop = zeros(1,dim_sub_const);

    for j=1:nr_sat
        % Calcula as distancias receptor - satelite cada combinacao
```

```
        r(j) = sqrt((coordinates(1)-sat_vis(j,1))^2 + (coordinates(2)-sat_vis(j,2))^2 + (coordinates(3)-sat_vis(j,3))
             ^2);
    end

    for i=1:max
        for n=1:dim_sub_const
            % Matriz G para cada combinacao satelites
            G(n,1) = (coordinates(1)-sat_vis(aux(i,n),1))/r(aux(i,1));
            G(n,2) = (coordinates(2)-sat_vis(aux(i,n),2))/r(aux(i,2));
            G(n,3) = (coordinates(3)-sat_vis(aux(i,n),3))/r(aux(i,3));
            G(n,4) = 1;
        end

        % Calcula o PDOP da combinacao
        H = inv(G.'*G);
        pdop(i) = sqrt(H(1,1) + H(2,2) + H(3,3));
    end

    % Encontra o PDOP minimo e a posicao na matriz
    [min_pdop, pos] = min(pdop(:));


    for k=1:dim_sub_const
        % Determina os IDs atraves da posicao na matriz onde estava o PDOP
        % minimo, e vai buscar a matriz das coordenadas dos satelites (que tambem continha os IDs)
        id_sat_min_pdop(k)= sat_vis(aux(pos,k),4);
    end

end

%% pseudoranges_calc.m

function[pseu_meas] = pseudoranges_calc(sub_const_matrix,coord_recep,sat_dim_wanted,iono,var_ruido)
pseu_meas(sat_dim_wanted)=0;
    for k=1:sat_dim_wanted
        pseu_meas(k) = sqrt((coord_recep(1)-sub_const_matrix(k,1))^2 + (coord_recep(2)-sub_const_matrix(k,2))^2 + (
             coord_recep(3)-sub_const_matrix(k,3))^2) + coord_recep(4);

        LLA = ecef2lla ([coord_recep(1) coord_recep(2) coord_recep(3)]);

        [sat_E,sat_N,sat_U] = ecef2enu(sub_const_matrix(k,1),sub_const_matrix(k,2),sub_const_matrix(k,3),LLA(1),LLA(2)
             ,LLA(3),referenceEllipsoid('wgs84'));
        [sat_elev_azi] = elev_azi(sat_E,sat_N,sat_U);

        elev = sat_elev_azi(1,1);
        noise=var_ruido*randn();     %Rudo
        pseu_meas(k) = pseu_meas(k) + noise;
        if iono == 1
            pseu_meas(k) = pseu_meas(k) + 10/sind(elev);
        end
    end
end


%% path.m

function [P]=path()

    f=1; % Frequncia a que as posies so actualizadas (1Hz)
    x_0=0;
    y_0=0;
    z_0=0;
    v_0 = 50; %Velocidade inicial (50m/s)
    m=10;     %Angulo de mscara
    P=[x_0 y_0 z_0 v_0 0 m]; % Inicializa do percurso

    %Percurso AB
    t=100; %dura do percurso em segundos
    a=1; %acelera constante

    for i = 1:t
        x=x_0;
```

```
        y=y_0+(a*i^2)/2;
        z=z_0;
        v=v_0+a*i;
        P=[P;x y z v a m];
    end

   % Percuso BC
    t=50; % d u r a  do percurso em segundos
    a=0;  % Velocidade constante a partir de B
    w=(3*pi/2)/t;   % velocidade angular
    r=v/w;          % raio do percuso
    c = [x, y+r];
    for i = 1:(t/f)
        x = c(1)-r*(1-cos(i*w));
        y = c(2)+r*sin(i*w);
        P=[P;x y z v a m];
    end
    % Percurso CD
    t=50; % d u r a  do percurso em segundos
    x_c=x;
        for i = 1:t
        x=x_c+v*i;
        P=[P;x y z v a m];
    end
    % Percurso DE
    t=50; % d u r a  do percurso em segundos
    x_d=x;
    m=28;
    for i = 1:t
        x=x_d+v*i;
        P=[P;x y z v a m];
    end

end


%% elev_azi.m

% CONVERTE DE ENU PARA AZIMUTE E ELEVA O
function sat_elev_azi=elev_azi(e,n,u)

for i =1:size(e(:,1))

    azim(i,1)=rad2deg(atan2(e(i,1),n(i,1)));
    elev(i,1)=rad2deg(asin(u(i,1)/sqrt(e(i,1)^2+n(i,1)^2+u(i,1)^2)));

end
%el=rad2deg(atan2(u,sqrt(n^2+e^2)));
sat_elev_azi = [azim, elev];
end


%% DMStoD.m

% Converte graus, minutos e segundos para graus.
function [D] = DMStoD (DMS)

    D = zeros(size(DMS,1),1);

    for i=1:size(DMS,1)

        for j=1:size(DMS,2)

            D(i) = D(i) + DMS(i,j) / 60^(j-1);

        end
    end

end


%% Calc_EA.m
```

```
function[E] = Calc_EA(M,e)

    tol = 10^-8;
    Etemp = M;
    ratio = 1;

    while abs(ratio) > tol
        f_E = Etemp - e*sin(Etemp) - M;
        f_Eprime = 1 - e*cos(Etemp);
        ratio = f_E/f_Eprime;
        if abs(ratio) > tol
            Etemp = Etemp - ratio;
        else
            E = Etemp;
        end
    end

end


%% calc_dist.m

% Calcula distancia entre duas coordenadas
function[Delta] = calc_dist(sub_const_matrix,coordinates,sat_dim_pret)

for k=1:sat_dim_pret
    dist(k) = sqrt((coordinates(1)-sub_const_matrix(k,1))^2 + (coordinates(2)-sub_const_matrix(k,2))^2 + (coordinates
         (3)-sub_const_matrix(k,3))^2);
end
Delta = dist;

end


%% isValidhour.m

function res = isValidhour( hour )

if (length(hour)~=8)
    res=0;
    return;
end

HHstr = hour(1:2);
HH=str2double(HHstr);

if (HH<0 || HH>23 || isnan(HH)==1)
    res=0;
    return;
end

MMstr = hour(4:5);
MM=str2double(MMstr);
if (isempty(MM)==1 || (60<MM)&& (MM<0) || isnan(MM)==1)
   res=0;
   return;
end

SSstr = hour(7:8);
SS=str2double(SSstr);
if (isempty(SS)==1 || (60<SS)&& (SS<0) || isnan(SS)==1)
   res=0;
   return;
end

if (hour(3)~=':'||hour(6)~=':')
    res=0;
    return;
end
res=1;
end
```

```
%% isValidDate.m

function res=isValidDate(d)
% d = Enter date in 'DD/MM/YYYY' format
if (length(d)~=10)
    res=0;
    return;
end
yearstr=d(7:10);
year=str2double(yearstr);
if (year<1993 || isnan(year)==1)
    res=0;
    return;
end
monthstr=d(4:5);
month=str2double(monthstr);
if (isempty(month)==1 || isnan(month)==1)
   res=0;
   return;
elseif (month<1||month>12)
      res=0;
      return;
end
if (d(3)~='/'||d(6)~='/')
    res=0;
    return;
end
daystr=d(1:2);
day=str2double(daystr);
if (isempty(day)==1 || isnan(day)==1)
    res=0;
    return;
end
if (month==1||month==3||month==5||month==7||month==8||month==10||month==12)
    if (day<1||day>31)
        res=0;
        return;
    end
end
if (month==4||month==6||month==9||month==11)
    if (day<1||day>30)
        res=0;
        return;
    end
end

if (rem(year,400)==0)
    if (month==2)
        if (day<1||day>29)
            res=0;
            return;
        end
    end
else if (rem(year,100)==0)
        if (month==2)
            if (day<1||day>28)
                res=0;
                return;
            end
        end
else if (rem(year,4)==0)
        if (month==2)
            if (day<1||day>29)
                res=0;
                return;
            end
        end
    else
         if (month==2)
            if (day<1||day>28)
```

```
                res=0;
                return;
            end
        end
    end
    end
end

if day < 1 && month<7 && year < 1993
    res=0;
    return
end

res=1 ;
end


%% isValidcoordinates.m

function [test,coord_graus] = isValidcoordinates(coordinates)

if (length(coordinates)~=10)
    test=0;
    coord_graus=0;
    return;
end

if (coordinates(3)~='-'||coordinates(6)~='-'||coordinates(9)~='-')
    test=0;
    coord_graus=0;
    return;
end

dd = str2double(coordinates(1:2));
mm = str2double(coordinates(4:5));
ss = str2double(coordinates(7:8));
N_S= coordinates(10);

if N_S == 'N' || N_S =='E'
    coord_graus = DMStoD ([dd mm ss]);
    test=1;
    return
else if N_S == 'S' || N_S =='W'
    coord_graus = -DMStoD ([dd mm ss]);
    test=1;
    return
    end
end

if (abs(coord_graus)>180)
if (abs(coord_graus)>90)
    test=0;
    coord_graus=0;
    return;
end
end
end
```