UNIVERSIDADE DA BEIRA INTERIOR
Engenharia

Prevenção Automática de Colisão Estática em Navegação de Trajetórias em 4D

José Carlos Gonçalves Félix

Dissertação para obtenção do Grau de Mestre em
Engenharia Aeronáutica
(Ciclo de Estudos Integrado)

Orientador: Prof. Doutor Kouamana Bousson

Covilhã, outubro de 2015

"Try to learn something about everything, and everything about something"

Thomas Henry Huxley

Dedicatória

Dedico à minha família, em especial aos que já cá não estão, que sempre me apoiaram e ansiaram por este momento.

Agradecimentos

O maior agradecimento que posso fazer é aos meus 2 melhores amigos: Mãe e Pai, pois se eu trabalhei para este momento ao longo dos últimos 5 anos, eles trabalharam para ele desde que nasci, e claro, também ao meu grande irmão Francisco, a minha principal fonte de motivação.

À restante família, Avós, Tios, Tias, Primos e Primas, e o meu pequenino afilhado, quer os que estão na ilha lá no meio do Atlântico, quer os Transmontanos, pois sem dúvida nenhuma que sem eles também não teria conseguido chegar onde cheguei, porque a eles devo a minha personalidade e educação, construída desde pequeno.

Agradeço também a uma segunda família, da qual 3 grandes amigos faziam parte há mais anos: a todos aqueles que em 2010 entraram comigo neste curso de Engenharia Aeronáutica e aos que na Covilhã conheci entretanto, de onde tenho de referenciar os meus 2 colegas de casa Luís e Márcio, sem esquecer a Rita, e que após 5 anos digo com o maior orgulho que são a minha segunda família, a malta que sempre me fez sentir em casa e que para sempre fica no meu coração.

Por último, agradeço à peça fulcral neste processo, a pessoa que me deu toda a liberdade e confiança para conseguir concluir esta dissertação, e que com o seu sentido de humor sempre animava nos momentos que pareciam de maior dificuldade: o meu orientador, o professor Doutor Kouamana Bousson.

Resumo

No mundo atual do controlo de tráfego aéreo uma das maiores preocupações é a minimização dos atrasos existentes em diversos aeroportos, certas vezes provocados por alterações repentinas nas trajetórias devido à existência de alguma zona interdita à navegação aérea/obstáculos estáticos. Esta dissertação teve como objetivo a investigação de diversos métodos para gerar uma trajetória que conseguisse prever e evitar a colisão de uma aeronave com um obstáculo estático tendo em conta sempre o objetivo principal de reduzir o tempo total de uma trajetória com 5 *waypoints*.

Numa primeira fase estudaram-se apenas teoricamente quatro métodos de geração de trajetória: por curvas de *Bézier*, por *Splines Cúbicas*, por *B-Splines*, e pelo método de *Overhauser*. Nesta fase concluiu-se que nem as curvas de *Bézier*, nem as *B-Splines* provocariam resultados satisfatórios, pois o primeiro método tem um controlo global, tornando difícil controlar as distâncias de passagem de determinados pontos, e o segundo método é demasiado pesado.

Apenas com o método de *Overhauser* e das *Splines Cúbicas* aproximou-se a trajetória inicialmente interpolada, onde se confirmou que ambos os métodos cumpriam com requisitos de passagem a uma distância máxima dos *waypoints* intermédios.

Num segundo teste de simulação numérica a estes dois métodos, o *software MATLAB®* apresentou dificuldades em cumprir com condições extra, que implicavam o respeito pelas velocidades mínimas e máximas da aeronave utilizada para o estudo, na minimização do comprimento total do método da *spline cúbica*. Por esta razão o estudo prosseguiu apenas com o método de *Overhauser*.

Na simulação final, concluiu-se que o método de *Overhauser* é capaz de evitar colisão com um obstáculo, no máximo dois, dependendo, ainda assim, do tamanho dos obstáculos a evitar, uma vez que este método usa apenas polinómios de grau 2, o que o torna algo limitado. Observou-se também uma redução no comprimento total da trajetória de 7,83% face ao comprimento total obtido para a trajetória interpoladora, e uma redução na ordem dos 43% comparativamente ao tempo inicial. Estudos futuros poderão apurar diferentes estratégias para utilização do método da *Spline Cúbica* na prevenção de colisão, o que certamente oferece uma solução mais satisfatória, uma vez que é um método com mais coeficientes, logo, melhor flexibilidade.

Palavras-Chave

Minimização do tempo; Minimização do comprimento de trajetória; Trajetória 4D; *Overhauser*; *Bézier*; *B-Splines*; *Splines Cúbicas*; Prevenção de Colisão.

Abstract

In today's world of air traffic control a major concern is the minimization of existing delays at various airports, sometimes caused by sudden changes in trajectory due to the existence of any protected zone to air navigation/statical obstacles. This work aimed to investigate several trajectory generation methods that can predict and prevent the collision of an aircraft with a statical obstacle, always taking into account the main goal of reducing the total time of a trajectory with five waypoints.

Initially four methods trajectory generation were compared theoretically: Bezier curves, Cubic Splines, B-splines, and the Overhauser method. On this stage it was concluded that neither the Bezier curves nor B-splines would lead to satisfactory results because the first method has overall control, making it difficult to control the distance pass certain points, and the second method is too heavy.

The trajectory initially interpolated was then approached with the Overhauser method and Cubic Splines Method, which has confirmed that both methods met with maximum distance requirements on intermediate waypoints.

On a second numerical simulation test with these two methods, the MATLAB software presented difficulties to comply with additional conditions, which implied compliance for the minimum and maximum speeds of the aircraft used for the study, to minimize the total length of the Cubic Spline Method. For this reason, the study continued only with the Overhauser method.

In the final simulation, it was concluded that the Overhauser method is able to avoid collision with an obstacle, at most two, depending yet the size of the obstacles to be avoided, since this method uses only polynomials of degree 2, which makes it somewhat limited. There was also a 7.83% reduction in the total length of the trajectory over the total length obtained for the interpolating path, and a reduction of around 43% compared to the initial time. Future studies will determine different strategies for using the cubic spline method for collision avoidance, which certainly gives a more satisfactory solution, since it is a method with more coefficients thus better flexibility.

Keywords

Time Minimization; Trajectory Length Minimization; Obstacle Avoidance; 4D Trajectory; Overhauser; Bézier; B-Splines; Cubic Splines.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

DCA – *Departamento de Ciências Aeroespaciais*

GPS – *Global Positioning System*

MILP – *Mixed Integer Linear Programming*

ORCA – *Optimal Reciprocal Collision Avoidance*

UAV – *Unmanned Air Vehicle*

Capítulo 1 - Introdução

1.1. Problema da Navegação de Trajetórias em 4D

Ao longo dos últimos anos os UAVs têm vindo a ganhar um papel cada vez mais importante na realização de atividades que antes eram apenas realizadas por aeronaves tripuladas. A vigilância aérea em zonas de guerra, pelos riscos inerentes que por vezes acarreta, é uma das atividades que tem vindo a ser relegada para o domínio das aeronaves não-tripuladas. À semelhança desta, também outras atividades, como a inspeção aérea de linhas elétricas; operações de busca, etc., podem ser executadas por UAVs.

Aquando a escolha da rota a operar pelo UAV, por vezes é possível existir num determinado dia, um inesperado obstáculo que se encontre no caminho, como por exemplo: numa inspeção de linha elétrica poderá haver uma árvore no caminho traçado; ou numa vigilância urbana, apesar de se terem informações sobre os edifícios existentes, uma antena poderá encontrar-se na rota; ou até mesmo espaço aéreo restringido pela Força Aérea.

É importante dotar o UAV de um sistema de deteção e prevenção automática de colisão estática com estes obstáculos, que permita recalcular a trajetória de forma a evitar conflitos. Este sistema terá de ter em conta não só a posição espacial da aeronave, mas também a posição temporal, o que deixa antever que a trajetória será trabalhada em 4D.

A navegação em 4D consiste nas 3 dimensões espaciais (x, y, z) mais o tempo como 4ª dimensão. Isto significa que qualquer atraso é na verdade uma distorção da trajetória quer como uma mudança de altitude quer como alteração da posição horizontal. O conceito da navegação em 4D baseia-se na integração de tempo para a trajetória de uma aeronave (por exemplo). O objetivo principal desta navegação é o de assegurar um voo numa trajetória ótima e sem restrições, sendo que para isso a aeronave é obrigada a cumprir com muita precisão uma hora de chegada extremamente precisa num ponto designado (chamados *waypoints*).

Para alcançar o destino a tempo, a trajetória 4D conta com uma constante passagem por pontos perfeitamente delineados, e que podem ser constantemente recalculados conforme a aeronave faz o percurso.

Uma vez que o percurso é otimizado através dos *waypoints* para assegurar não só as 3 dimensões ótimas, mas também o cumprimento da 4ª dimensão (tempo), traz imensas vantagens em várias áreas:

- Minimização dos atrasos, o que melhora as operações de tráfego aéreo e garante maior satisfação da parte dos passageiros;
- Otimização dos planos de voo, que permite economizar no combustível e consequentemente também reduzir as emissões poluentes;
- Uma vez que as trajetórias são mais bem controladas é possível prever, com maior antecedência, eventuais planos de voo cruzados, de modo a que se possam proceder às devidas alterações o mais rápido possível, evitando assim a colisão aérea.

1.2. Desenvolvimento histórico dos estudos em 4D

A existência de trabalhos nesta área em 4D é relativamente escassa, e os estudos realizados incidem sobretudo sobre prevenção de colisão entre aeronaves tripuladas no plano horizontal (2D) e em 3D.

Muitos dos estudos realizados nos últimos anos incidem sobretudo na problemática da prevenção de colisão no novo método de controlo de tráfego aéreo (*free-flight*), uma vez que a permissão concedida às aeronaves para que escolham livremente as suas trajetórias no espaço aéreo aumenta a possibilidade de conflitos entre trajetórias.

Apesar de este trabalho incidir sobre a prevenção de colisão com obstáculos estáticos, aplicada a UAVs, há certos conceito aliados ao *free-flight* que são de relevante interesse e podem ser aplicados neste estudo:

- As duas zonas virtuais: *alert zone* e *protected zone* [1] que funcionam da seguinte forma: se uma aeronave entra na *alert zone* duma outra aeronave, a maior das zonas, o sistema alerta a proximidade dessa aeronave de modo a que se preste especial atenção de modo a evitar que a *protected zone* seja invadida, pois esta é considerada a zona na qual não deverá haver nenhuma perturbação.
- Bousson [2] apresentou um método de prevenção de colisão com um modelo de controlo preditivo considerando o tráfego global numa determinada área, de modo a que a solução do conflito com uma aeronave não crie possíveis colisões com outras. Também no presente trabalho é importante assegurar que o desvio que a aeronave terá de fazer para evitar o *obstáculo A* não implique conflitos com um outro obstáculo, pelo que o

método de prevenção de colisão terá de ter um ponto de vista global, e não apenas local (apenas em relação a um obstáculo).

No que toca à prevenção de colisão para UAVs, Beard & McLain [3] propuseram um algoritmo para otimizar as trajetórias percorridas por várias aeronaves não tripuladas (sistema Multi-UAV) constituintes de uma equipa numa missão de busca, tendo em conta possíveis obstáculos que pudessem encontrar. Contudo este algoritmo está dependente da cooperação entre as aeronaves, quer para evitar as colisões, quer para otimizar a trajetória, para além de que está aplicado apenas no plano horizontal. Por sua vez, Alejo et. al. [4] formularam um método baseado em *"3D Optimal Reciprocal Collision Avoidance"* (*ORCA*) para um sistema Multi-UAV de quadrotors. Neste estudo, à semelhança do trabalho anteriormente referido, cada quadrotor evita a colisão usando informações das posições e velocidades relativas dos outros UAVs. A solução para conflitos com obstáculos estáticos é conseguida por meio da execução de um método reativo, razão pela qual este é aplicável em tempo real, computando uma solução ótima assegurando uma minimização do afastamento em relação ao obstáculo, porém não há uma trajetória com *waypoints* previamente definida e o modelo dinâmico do voo é apenas para quadrotors, incluindo portanto, apenas restrições holonómicas.

Outro estudo, realizado em 2002 por Richards & How [5] tinha também proposto um método 2D, aplicável tanto a sistemas Multi-UAV, como a apenas um UAV, que recorre a *MILP* (*Mixed Integer Linear Programming*) para otimizar a trajetória, contudo o modelo da dinâmica do voo concebido utilizou apenas restrições lineares o que limita desde logo as manobras possíveis para evasão tornando o método impraticável para qualquer adaptação em tempo real, de modo que as trajetórias por ele geradas necessitavam de uma análise humana de modo a averiguar se seria ou não possível a aeronave executá-las.

Para aplicações em tempo real com aeronaves não cooperativas, Carbone et. al. [6] apresentaram um algoritmo de tomada de decisão gerador de manobras evasivas no plano horizontal e no plano vertical, utilizando apenas um *input* de controlo (mudança de direção, ou mudança de altitude, ou aumento/diminuição de velocidade), empregando para isso o conceito do cone de colisão, sendo que não há, no artigo publicado, nenhuma prova com simulações reais que consiga aferir a taxa de sucesso do algoritmo proposto.

Um trabalho mais focado na prevenção automática de colisão estática em 3D é o que foi realizado por He e Iyer em 2006 [7] [8]. Este estudo estima a distância a um objeto por meio de uma câmara e recorre a um dispositivo GPS com a finalidade de obter conhecimento da velocidade total do UAV. Tópicos como: a estimativa do movimento no plano da imagem da câmara (num ambiente sem qualquer tipo de restrições); e estimativa das velocidades lineares e angulares do UAV, bem como da distância a qual o mesmo se encontra dos vários campos de

visão apresentam-se bem aprofundados. Infelizmente não há qualquer tipo de teste que comprove uma bem-sucedida navegação por meio de *waypoints* definidos evitando quaisquer tipo de obstáculos, contudo, o processamento de imagem e consequente divisão e classificação da mesma em blocos estruturais e não estruturais prova ser um bom método de mapeamento do cenário.

Numa abordagem diferente e comprovadamente bem-sucedida em prevenção de colisão, Mcfadyen et. al. [9] sugerem uma adaptação do conceito *Seek & Avoid* para UAVs. Para isso usam o controlo preditivo, utilizando uma câmara esférica de modo a torna-lo visual, recorrendo para isso a apenas a um ponto fixo. A prevenção de colisão é efetuada sem qualquer estimativa da distância do objeto, por meio das propriedades do movimento espiral cónico, ou seja, o vetor de estado e a função objetivo são concebidos para que a aeronave seja guiada ao longo de uma trajetória em espiral divergente. Uma vez encontrada uma posição do UAV que faça convergir a função objetivo, este volta à sua trajetória normal. Ora este método restringe as manobras de evasão e o movimento em espiral significa também uma constante alteração no controlo do UAV, o que consome energia extra.

Em relação à problemática de navegação em 4D, estudos realizados nos últimos cinco anos [10] visaram resolver o problema da otimização de trajetória na navegação 4D através de uma abordagem baseada nos métodos diretos com integração pseudoespectral com polinómios de Chebyshev. O uso de métodos diretos permitiu a transformação do problema de controlo ótimo em problemas de programação não linear. O controlo utilizado para levar a aeronave a percorrer a trajetória gerada com o menor desvio possível foi um controlo preditivo.

O trabalho aqui apresentado nesta dissertação pretende continuar estes estudos e acrescentar à trajetória otimizada com passagem por *waypoints* um método de prevenção de colisão com obstáculos estáticos.

1.3. Objetivos do Trabalho

O objetivo principal deste estudo consiste na elaboração de um método de geração de trajetórias 4D que realize a prevenção de colisão estática com uma obstáculo modelado por uma determinada forma geométrica com coordenadas previamente estabelecidas, tendo em conta a minimização dos atrasos inerentes ao processo de deteção e consequente resolução de conflito com os objetos fixos presentes na rota. Para que isto seja alcançado é preciso ter em conta os seguintes pontos da problemática levantada pelo objetivo principal acima descrito:

- Uma das formas de minimizar o tempo que a aeronave demora a percorrer a trajetória 4D, é através da minimização do comprimento total da trajetória, ou seja, em vez de interpolar todos os *waypoints*, a trajetória aproxima-se dos intermédios, tendo em conta uma distância máxima de passagem dos mesmos;

- Um dos pontos de crucial importância é o respeito pela velocidade mínima e máxima da aeronave, uma vez que da velocidade depende diretamente o valor do tempo, e desta forma, ao se respeitar essas condições garante-se uma minimização mais eficaz;

- Na fase de contorno dos obstáculos é importante realizar o menor desvio possível da trajetória previamente estabelecida, precisamente por causa da questão de minimizar atrasos e seguir ao máximo a trajetória otimizada.

Estes são os elementos que compõem o objetivo principal a cumprir na dissertação, por forma a garantir a geração de trajetórias com minimização do tempo que a aeronave demora a percorrer todo o percurso.

1.4. Estrutura da dissertação

O **Capítulo 2** apresenta a definição do plano de voo em 4D, onde estão escolhidos os *waypoints* que serão parte da trajetória, e onde será utilizada inicialmente a aproximação de que o voo é todo feito em voo nivelado. É também nesse capítulo que se promove um estudo conceptual de 4 métodos de geração de trajetórias que, à partida, podem ser de elevado interesse para o objetivo da dissertação:

- Método das ***Splines Cúbicas***;
- Método das ***Curvas de Bézier***;
- Método das ***B-Splines***;
- Método de ***Overhauser***.

A exploração teórica das caraterísticas, vantagens e limitações destes métodos, tem como objetivo filtrá-los deixando apenas dois métodos que sejam mais relevantes para o estudo, justificando, comparativamente, o porquê da rejeição dos outros dois.

No **Capítulo 3** dá-se a modelação do problema, primeiramente apenas procurando aproximar a trajetória com passagem a uma distância máxima permissível face aos *waypoints* intermédios, para que se consiga minimizar o comprimento da trajetória. Numa 2ª modelação, desta feita mais complexa, ter-se-á em conta os limites de velocidade da aeronave para que também o tempo seja minimizado.

A simulação final, onde será testado se o objetivo final foi ou não alcançado, encontra-se no **Capítulo 4**, onde um algoritmo para a deteção de obstáculo/zona interdita à navegação é apresentado, tal como o algoritmo que previne a colisão com esse mesmo obstáculo. Posteriormente os resultados da simulação final são apresentados e analisados.

Por fim, no **Capítulo 5** encontram-se as conclusões sobre o estudo, e recomendações para possíveis futuros trabalhos, que tenham este estudo como base.

Capítulo 2 - Métodos de Geração de Trajetórias

Conforme referido anteriormente, a trajetória terá de ser gerada ao longo de *waypoints* previamente definidos em 4D. Isto significa que terão de ser respeitados os tempos predefinidos para cada *waypoint*, sendo por isso importante a inexistência de atrasos, portanto é fulcral que a trajetória seja otimizada para o mínimo comprimento possível entre cada *waypoint*.

Para além da minimização do atraso acima referida, é também preciso ter em conta as restrições da aeronave, uma vez que esta não poderá efetuar determinadas manobras, pelo que terá de haver especial atenção às velocidades mínimas e máximas, intervalos permissíveis dos ângulos das superfícies de controlo, e ao envelope de voo da aeronave. É de relativa importância atender à continuidade e suavidade das curvas geradas.

2.1. Plano de voo

Aqui será apenas exemplificado como o programa procede à geração da rota, utilizando coordenadas escolhidas aleatoriamente para uma rota entre Porto e Madrid.

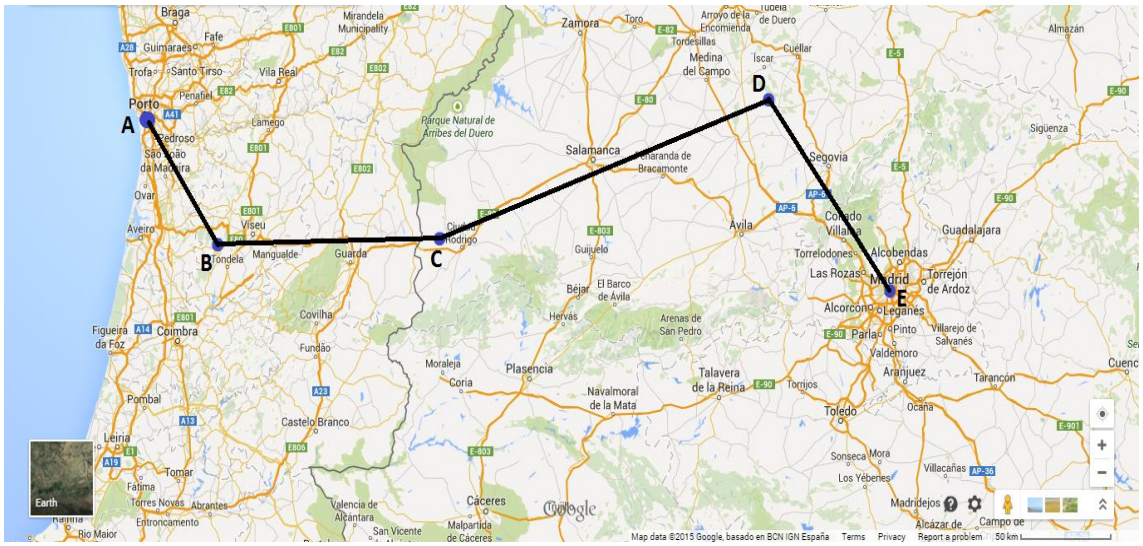

Figura 1 - Rota Exemplo Porto - Madrid.

Os *waypoints* A e E correspondem ao *Aeroporto Francisco Sá Carneiro* (Porto) e ao *Aeropuerto Adolfo Suárez* (Madrid – Barajas), respetivamente. Os *waypoints* B e D estão na fase de subida e descida, consecutivamente, e o *waypoint* C está em altitude cruzeiro. As coordenadas geodéticas utilizadas neste exemplo são:

- A (41°14'8'' N, 8°40'41'' W, 69 m);
- B (40°35'50'' N, 8°14'10'' W, 1900 m);
- C (40°49'18'' N, 6°46'15'' W, 5900 m);
- D (41°12'52'' N, 4°31'53'' W, 2586 m);
- E (40°28'20'' N, 3°33'39'' W, 610 m). [11]

Uma vez que as coordenadas dos *"waypoints"* são geodéticas, é necessário a sua conversão para coordenadas geocêntricas, pois, normalmente, os modelos de otimização de trajetórias (posteriormente descritos) requerem coordenadas geocêntricas. Esta conversão pode ser feita através do método apresentado por *Vermeille* [12]:

$$X = (h + n)\cos\varphi\cos\lambda \quad \textbf{(1)}$$

$$Y = (h + n)\cos\varphi\sin\lambda \quad \textbf{(2)}$$

$$Z = (h + n - e^2 n)\sin\varphi \quad \textbf{(3)}$$

onde:

- φ – latitude geodética;
- λ – longitude geodética;
- $n = \frac{a}{\sqrt{1-e^2\sin^2\varphi}}$ **(4)**

Utilizando o modelo esférico da Terra, até 100 km de altitude, consideramos $a = R$, em que R representa o raio da terra, e uma excentricidade (e) nula:

$$a = R = 6400000\,m \quad \textbf{(5)}$$

$$e = 0 \quad \textbf{(6)}$$

Os trajetos entre os *waypoints* escolhidos têm que respeitar as limitações das aeronaves, conforme dito acima, como por exemplo, o ângulo de trajetória γ, deverá ser menor que $\gamma_{máx}$ da aeronave, e neste caso, tendo em conta que:

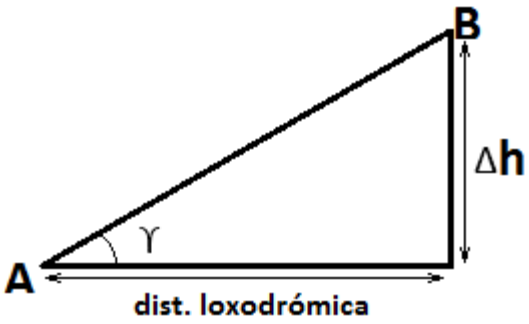

Figura 2 - Esquema de cálculo do ângulo de trajetória.

Calculam-se os ângulos de trajetória de cada trajeto:

Tabela 1 - Cálculo dos ângulos de trajetória.

Trajeto	Variação da altitude Δh (m)	Distância Loxodrómica (m)	Ângulo ϒ (graus)
A - B	1831	80052.49	1.31
B - C	4000	125917.64	1.82
C - D	-3314	192761.05	-0.98
D - E	-1976	116010.70	-0.98

A distância loxodrómica é determinada por:

$$(AB)_{NM}^{loxo} = \frac{(\Delta\varphi_{AB})_{minutos}}{\cos V} \quad (7)$$

Com:

- Diferença de latitudes geodéticas dos pontos A e B: $\varphi_{AB} = |\varphi_B - \varphi_A|$ **(8)**
- Ângulo segundo o qual a loxodromia corta os meridianos: **(9)**
$$V = arctan_2((\Delta\lambda_{AB})_{minutos}, (\Delta\varphi^*_{AB})_{minutos})$$
- Diferença de longitudes geodéticas dos pontos A e B: $\Delta\lambda_{AB} = |\lambda_B - \lambda_A|$ **(10)**
- Diferença de latitudes geodéticas dos pontos A e B em minutos angulares: **(11)**
$$\Delta\varphi^*_{AB} = |\varphi^*_B - \varphi^*_A|$$
- Conversão de latitude para minutos angulares: **(12)**
$$\varphi^*_0 = \frac{10800}{\pi} \ln\left[\tan\left(\frac{\varphi_0}{2} + 45°\right)\right] \text{ (em minutos angulares)}$$

A passagem para 4D implica o cálculo dos tempos de passagem em cada waypoint, assumindo t=0 no primeiro (waypoint A). Os restantes tempos são calculados por:

$$t_{k+1} = t_k + \eta * \Delta t_{k+1} \quad \textbf{(13)}$$

Com:

- $1.2 \leq \eta \leq 1.4$ – Correção devido ao facto de se assumir velocidade média em cada trajeto, e a rajadas de vento, etc. – neste caso considera-se = 1.3 – valor médio; **(14)**
- Variação do tempo: $\Delta t_{k+1} = \frac{distância\ (W_{pk}, W_{pk+1})}{velocidade\ média\ \left(\frac{V_k + V_{k+1}}{2}\right)}$ **(15)**
- Velocidade no ponto *k*: $V_k = \sqrt{\frac{2W_k}{\rho * C_L * S}}$ **(16)**
- Densidade no ponto *k*

$$\rho(h_k) = \begin{cases} \rho_0(1 - 2.2558 \times 10^{-5} h_k)^{4.256060537} & se\ h < 11000 \\ 0.29707 \rho_0 e^{\left(1.576939464 \times 10^{-4}(11000 - h_k)\right)} & se\ 11000 < h < 20000\ m \end{cases} \quad \textbf{(17)}$$

- Densidade ao nível do mar: $\rho_0 = 1.225\ kg/m^3$ **(18)**
- Coeficiente de sustentação: $C_L = 1$ – Valor assumido para este exemplo; **(19)**
- Peso a considerar no ponto *k* (90% do peso à descolagem): $W_k = 0.9 \times W_{MTOW}$ **(20)**

O valor de W_k foi <u>assumido constante</u> ao longo de toda a rota (despreza-se o efeito do consumo de combustível no peso).

Como se pode observar, considera-se, **<u>por agora</u>** e para efeitos de simplificação, o voo como sendo **<u>nivelado</u>** ao longo de toda a rota, o que é uma aproximação aceitável, tendo em conta que os valores dos ângulos de trajetória calculados são muito pequenos, não ultrapassando os 2 graus em nenhum dos percursos entre *waypoints*.

Para efeitos de cálculo, consideram-se os seguintes valores caraterísticos de uma das configurações da aeronave desenvolvida no DCA "Olharapo":

Tabela 2 - Dados de uma configuração do "Olharapo".

Velocidade máxima de cruzeiro	30 m/s
Velocidade de descolagem	18 m/s
Velocidade de aterragem	19 m/s
Velocidade de perda	18 m/s
Peso à descolagem (MTOW)	15.3 kg
Coeficiente de sustentação	1.6
Área alar	0.8 m^2

Nota: Os valores de tempo são calculados dentro do próprio programa. De qualquer maneira seguem aqui os valores para este caso específico:

Tabela 3 - Tempos de passagem em cada waypoint.

Tempo "waypoint" A	0 segundos
Tempo "waypoint" B	6175 segundos
Tempo "waypoint" C	15957 segundos
Tempo "waypoint" D	30696 segundos
Tempo "waypoint" E	39248 segundos

2.2. Interpolação de trajetórias definidas por waypoints em 4D

Dentro das várias possibilidades existentes para geração de trajetórias otimizadas, neste estudo testaram-se quatro métodos em particular:

- Método de geração de trajetórias com ***Splines Cúbicas***;
- Método de geração de trajetórias com ***Curvas de Bézier***;
- Método de geração de trajetórias com ***B-Splines***;
- Método de ***Overhauser***.

Seguidamente procede-se à descrição de cada um destes métodos, bem como à sua respetiva simulação computacional em *MATLAB*®.

2.2.1. Método de Geração de Trajetórias com *Splines Cúbicas*:

Procura-se uma trajetória que passe por cada *"waypoint"* no instante especificado. As posições dos segmentos são geradas unindo três *"waypoints"* de cada vez, portanto para este caso em concreto, com 5 *"waypoints"*:

- Segmento 1: $P_{Wp1} - P_{Wp2} - P_{Wp3}$;
- Segmento 2: $P_{Wp2} - P_{Wp3} - P_{Wp4}$;
- Segmento 3: $P_{Wp3} - P_{Wp4} - P_{Wp5}$;

O vetor posição (*P*) de cada *"waypoint"* contém as suas coordenadas geocêntricas *x*, *y*, *z* e o tempo de passagem, pelo que é possível em cada segmento definir as trajetórias em 1 dimensão: *x(t)*, *y(t)* e *z(t)*.

Uma vez que cada segmento contém 3 pontos, a melhor forma de aproximar a trajetória entre eles é por meio de um polinómio de 3º grau.

Para o primeiro segmento (k=1), $P_{Wp1} - P_{Wp2} - P_{Wp3}$, com $t \in [t_0, t_2]$:

$$x(t) = a_0 + a_1(t - t_0) + a_2(t - t_0)^2 + a_3(t - t_0)^3 \quad \textbf{(21)}$$

$$y(t) = b_0 + b_1(t - t_0) + b_2(t - t_0)^2 + b_3(t - t_0)^3 \quad \textbf{(22)}$$

$$z(t) = c_0 + c_1(t - t_0) + c_2(t - t_0)^2 + c_3(t - t_0)^3 \quad \textbf{(23)}$$

Portanto para gerar as coordenadas *x*, *y* e *z* em cada instante é necessário determinar os coeficientes. Por exemplo, a determinação dos coeficientes a's necessita de 4 equações (pois temos 4 coeficientes). As equações são:

$$x(t_0) = a_0 + a_1(t_0 - t_0) + a_2(t_0 - t_0)^2 + a_3(t_0 - t_0)^3 = x_0 \quad \textbf{(24)}$$

$$x(t_1) = a_0 + a_1(t_1 - t_0) + a_2(t_1 - t_0)^2 + a_3(t_1 - t_0)^3 = x_1 \quad \textbf{(25)}$$

$$x(t_2) = a_0 + a_1(t_2 - t_0) + a_2(t_2 - t_0)^2 + a_3(t_2 - t_0)^3 = x_2 \quad \textbf{(26)}$$

$$\dot{x}(t_0) = a_1 + 2a_2(t_0 - t_0) + 3a_3(t_0 - t_0)^2 = dx_0 = \frac{x_1 - x_0}{t_1 - t_0} \quad \textbf{(27)}$$

Resolvendo temos então:

$$\begin{cases} a_0 = x_0 \\ a_1 = \dfrac{x_1 - x_0}{t_1 - t_0} \\ a_3 = \dfrac{x_2 - a_0 - a_1(t_2 - t_0) - \dfrac{x_1}{(t_1 - t_0)^2}(t_2 - t_0)^2 + \dfrac{a_0}{(t_1 - t_0)^2}(t_2 - t_0)^2 + \dfrac{a_1}{(t_1 - t_0)}(t_2 - t_0)^2}{\left((t_2 - t_0)^3 - \left((t_1 - t_0)(t_2 - t_0)^2\right)\right)} \\ a_2 = \dfrac{x_1 - a_0 - a_1(t_1 - t_0) - a_3(t_1 - t_0)^3}{(t_1 - t_0)^2} \end{cases} \quad \textbf{(28)}$$

Para um segmento k, com $t \in [t_{k-1}, t_{k+1}]$ vem:

$$x(t) = a_0^{(k)} + a_1^{(k)}(t - t_{k-1}) + a_2^{(k)}(t - t_{k-1})^2 + a_3^{(k)}(t - t_{k-1})^3 \quad \textbf{(29)}$$

$$y(t) = b_0^{(k)} + b_1^{(k)}(t - t_{k-1}) + b_2^{(k)}(t - t_{k-1})^2 + b_3^{(k)}(t - t_{k-1})^3 \quad \textbf{(30)}$$

$$z(t) = c_0^{(k)} + c_1^{(k)}(t - t_{k-1}) + c_2^{(k)}(t - t_{k-1})^2 + c_3^{(k)}(t - t_{k-1})^3 \quad \textbf{(31)}$$

As equações para resolução dos coeficientes são praticamente iguais, à exceção da última:

$$x^{(k)}(t_{k-1}) = a_0^{(k)} + a_1^{(k)}(t_{k-1} - t_{k-1}) + a_2^{(k)}(t_{k-1} - t_{k-1})^2 + a_3^{(k)}(t_{k-1} - t_{k-1})^3 = x_{k-1} \quad \textbf{(32)}$$

$$x^{(k)}(t_k) = a_0^{(k)} + a_1^{(k)}(t_k - t_{k-1}) + a_2^{(k)}(t_k - t_{k-1})^2 + a_3^{(k)}(t_k - t_{k-1})^3 = x_k \quad \textbf{(33)}$$

$$x^{(k)}(t_{k+1}) = a_0^{(k)} + a_1^{(k)}(t_{k+1} - t_{k-1}) + a_2^{(k)}(t_{k+1} - t_{k-1})^2 + a_3^{(k)}(t_{k+1} - t_{k-1})^3 = x_{k+1} \quad \textbf{(34)}$$

$$\dot{x}^{(k)}(t_{k-1}) = \dot{x}_{k-1}^{(k-1)}(t_{k-1}) \quad \textbf{(35)}$$

Tabela 4 - Coeficientes Polinomiais a determinar no cálculo das *Splines Cúbicas*.

Coeficientes Polinomiais *Spline Cúbica*				
Coeficientes da coordenada *x*	$a_0^{(k)}$	$a_1^{(k)}$	$a_2^{(k)}$	$a_3^{(k)}$
Coeficientes da coordenada *y*	$b_0^{(k)}$	$b_1^{(k)}$	$b_2^{(k)}$	$b_3^{(k)}$
Coeficientes da coordenada *z*	$c_0^{(k)}$	$c_1^{(k)}$	$c_2^{(k)}$	$c_3^{(k)}$

Nota: Para a exploração dos segmentos de trajetória, estes são representados da seguinte forma (para o caso em estudo):

- $P_{Wp1} - P_{Wp2}$;
- $P_{Wp2} - P_{Wp3}$;
- $P_{Wp3} - P_{Wp4} - P_{Wp5}$.

Este método tem as suas desvantagens, nomeadamente o difícil controlo local das curvas geradas, isto é, uma alteração súbita de um dos *waypoints* leva a que a curva se altere em boa parte, afetando quase toda a trajetória. Para os *waypoints* exemplo acima referidos, a

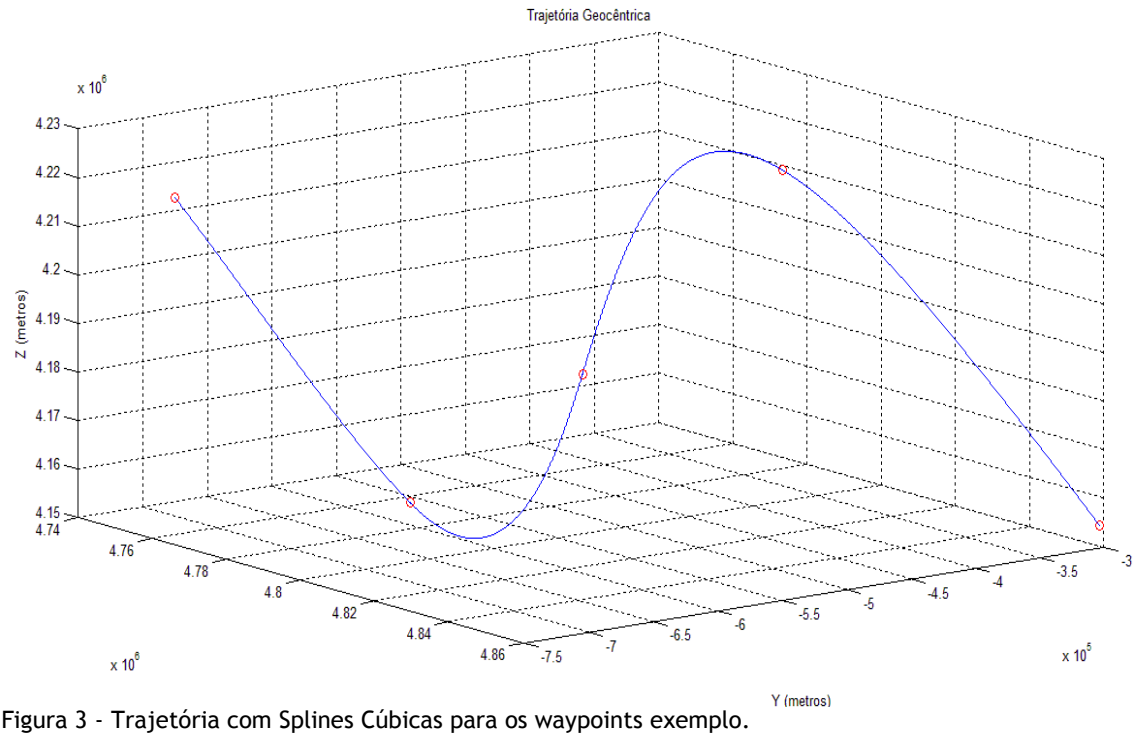

Figura 3 - Trajetória com Splines Cúbicas para os waypoints exemplo.

média de tempo que o algoritmo consumiu situa-se nos 4 segundos, muito por culpa das *splines* serem de 3º grau, o que força a resolução de um sistema de 4 equações e 4 coeficientes, o que é lento demais, para algo que se quer rápido o suficiente de modo a que na resolução de conflitos de colisão seja possível recalcular de forma eficiente as alterações de trajetória.

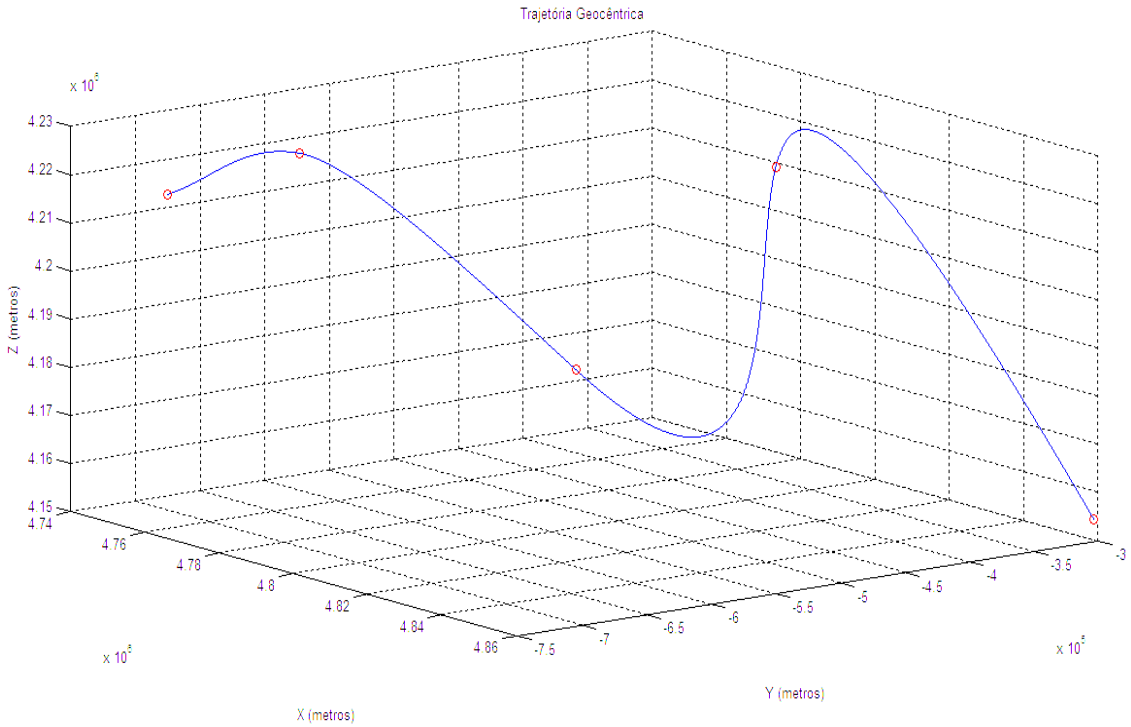

Figura 4 - Trajetória com Splines Cúbicas para waypoints exemplo com alteração no 2º waypoint.

Para além dos pontos negativos aqui referidos, há também que constatar o desvio relativamente elevado que, por exemplo, se nota na segunda imagem onde após o 3º *waypoint* se nota uma descida, ao longo do eixo z, superior a 10000 metros. A trajetória poderia ser mais suave nesse aspeto, isto é, não se demorar tanto tempo a orientar para o *waypoint* seguinte.

2.2.2. Método de Geração de Trajetórias com *Curvas de Bézier*:

Por volta de 1962, enquanto funcionário da *Renault*, Pierre Bézier apresentou um tipo de curvas ideais para o *design* em geral. Um conjunto de curvas que no seu cerne se resumem a um somatório de polinómios de *Bernstein* (de Sergei Natanovich Bernstein, 1912) multiplicados pelas coordenadas dos pontos. [13]

Este é um método de aproximação, pois apenas é garantida a passagem pelo ponto inicial e final, sendo que os intermediários contam para a ponderação (sendo chamados *atractor points*. O método de *Bézier* pode então ser caraterizado da seguinte forma:

- um conjunto de pontos discretos denominados *pontos de controlo*;
- um conjunto de funções base ou *blending functions*.

Se considerarmos um conjunto de 3 pontos P_0, P_1 e P_2, onde P_{x_i}, P_{y_i} e P_{z_i} indicam as coordenadas de cada ponto P_i, é possível determinar-se uma curva $K(u)$ através de uma expressão que faça a ponderação da contribuição de cada ponto de controlo:

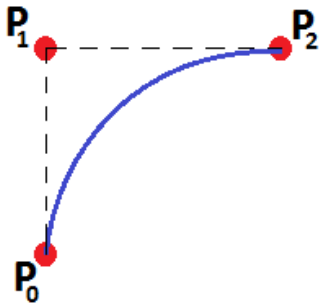

Figura 5 - Esquema de uma curva de Bézier para três pontos

Para projetar a curva de *Bézier* acima, é necessário parametrizar a mesma, pelo que utiliza-se um $0 \leq t \leq 1$, de modo a que se tenha um $P(t)$, tal que:

$$\begin{cases} P(0) = P_0 \\ P(1) = P_1 \end{cases} \tag{36}$$

Tendo os segmentos $P_0 \leftrightarrow P_1$ e $P_1 \leftrightarrow P_2$, seleciona-se um ponto aleatório (suponha-se a um terço de cada segmento) de modo a que se tenha $(1-t)$ à esquerda de cada um desses pontos, e t à direita:

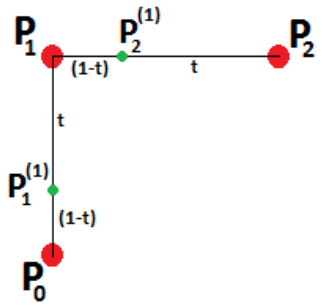

Figura 6 - Esquema 1 de construção da curva de Bézier de 3 pontos.

Após unirem-se os pontos $P_1^{(1)}$ e $P_2^{(1)}$ ficamos com um segmento, no qual se marca o ponto $P_2^{(2)}$, pelo qual irá então passar a curva de *Bézier*, ligando P_0, P_1 e $P_2^{(2)}$:

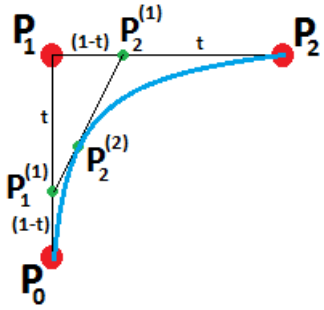

Figura 7 - Esquema 2 de construção da curva de Bézier de 3 pontos.

A partir destes esquemas retira-se então a equação da curva de *Bézier* gerada:

$$P(t) = (1-t).P_1^{(1)} + t.P_2^{(1)} \quad \textbf{(37)}$$

$$\text{Com } \begin{cases} P_1^{(1)}(t) = (1-t).P_0 + t.P_1 \\ P_2^{(1)}(t) = (1-t).P_1 + t.P_2 \end{cases} \quad \textbf{(38)}$$

Ao resolver fica:

$$P(t) = (1-t)^2.P_0 + 2.t.P_1.(1-t) + t^2.P_2 \quad \textbf{(39)}$$

$\boldsymbol{(1-t)^2}$; $\boldsymbol{2t(1-t)}$ e $\boldsymbol{t^2}$ são os **polinómios de *Bernstein*.**

Para o caso de construção de uma curva com 4 pontos, procede-se da mesma forma, ficando com a seguinte equação:

$$P(t) = t^3.P_3 + 3.t^2(1-t).P_2 + 3.t(1-t)^2.P_1 + (1-t)^3.P_0 \quad \textbf{(40)}$$

Ou

$$P(t) = \sum_{i=0}^{3} P_i . B_{i,3}(t) \tag{41}$$

Com os polinómios de *Bernstein*

$$\begin{cases} B_{0,3}(t) = (1-t)^3 \\ B_{1,3}(t) = 3.t.(1-t)^2 \\ B_{2,3}(t) = 3.t^2.(1-t) \\ B_{3,3}(t) = t^3 \end{cases} \tag{42}$$

Podemos ainda escrever a forma matricial da curva cúbica de *Bézier* que une os 4 pontos:

$$P(t) = [(1-t)^3 \quad 3t(1-t)^2 \quad 3t^2(1-t) \quad t^3] . \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix} \tag{43}$$

Simplificando fica:

$$P(t) = [1 \quad t \quad t^2 \quad t^3] . \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ 1 & 3 & -3 & 1 \end{bmatrix} . \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix} \tag{44}$$

Com base nestas deduções acima demonstradas, podemos então generalizar e escrever a **<u>definição geométrica das curvas de *Bézier*</u>** para *n+1* pontos de controlo ($P_0, P_1, \dots, P_n$):

$$P(t) = P_n^{(n)}(t) \tag{45}$$

Onde:

$$P_i^{(j)}(t) = \begin{cases} (1-t).P_{i-1}^{(j-1)}(t) + t.P_i^{(j-1)}(t) & se\ j > 0 \\ P_i & se\ j = 0 \end{cases} \quad e\ t \in [0,1] \tag{46}$$

Podemos também escrever a **definição analítica da curva de *Bézier*** para *n+1* pontos de controlo $(P_0, P_1, \dots, P_n)$:

$$P(t) = \sum_{i=0}^{n} P_i . B_{i,n}(t) \quad \textbf{(47)}$$

Onde:

- $B_{i,n}(t) = \binom{n}{i} t^i . (1-t)^{n-i}$ são os polinómios de grau *n*; **(48)**
- $\binom{n}{i} = \frac{n!}{i!(n-i)!}$ é um coeficiente binomial. [14] **(49)**

Há ainda um algoritmo relativamente mais simples de executar para a interpolação de curvas de Bézier: o **algoritmo *de Casteljau*** (1959).

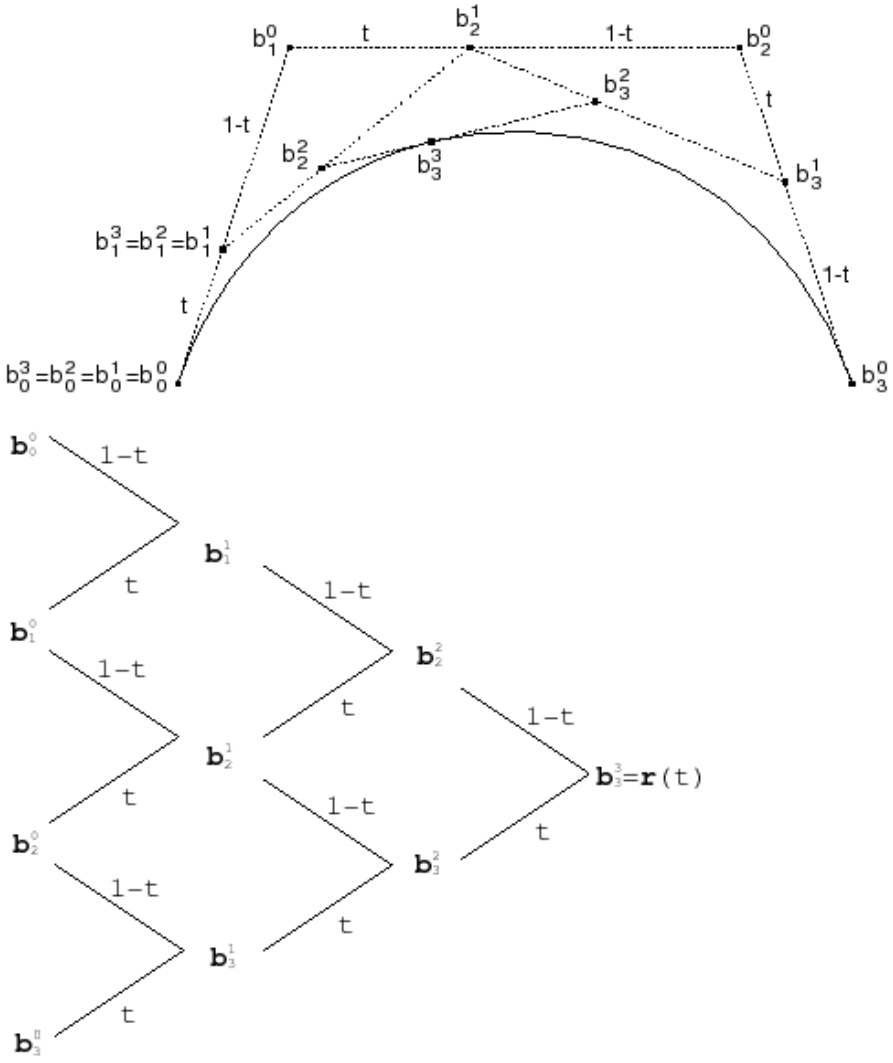

Figura 8 - Esquema do algoritmo de Casteljau. [15]

Neste método a equação $b_i^k(t) = (1-t)b_{i-1}^{k-1} + tb_i^{k-1};\ k = 1,2,\dots,n;\ i = k,\dots,n$ é utilizada recorrentemente para obter novos pontos de controlo, ou seja, no caso de b_3^3 temos: $b_3^3(t) = (1-t)b_2^2 + tb_3^2$, e assim sucessivamente tal como ilustrado na forma piramidal. Este algoritmo apesar de simples, requer ficar num *loop* para cada valor de t. [15]

As curvas de *Bézier* possuem duas grandes limitações:

- O controlo exercido pelos pontos de controlo é global, isto é, a movimentação de um ponto altera por completo toda a curva (com maior influência na vizinhança do ponto alterado);
- Não se pode usar uma curva cúbica de *Bézier* para aproximar um conjunto de *n* pontos. A única forma é usar vários segmentos de curva, ou aumentar o grau do polinómio, o que leva a uma maior complexidade matemática e computacional.

Para além destas limitações, há a dificuldade inerente de controlar o afastamento em relação aos pontos intermédios, e uma vez que tal não é possível, este método de geração de trajetórias perde o interesse para os casos em que é importante a passagem por determinados pontos intermédios ao longo da trajetória, precisamente o que é pretendido neste estudo, isto é, numa fase inicial a passagem por determinados *waypoints* intermédios (uma interpolação), e numa fase posterior a aproximação de trajetórias tendo em conta uma distância mínima. Por estas razões a exploração deste método neste estudo não passará desta descrição conceptual.

2.2.3. Método de Geração de Trajetórias com *B-Splines*:

Para solucionar os problemas inerentes às curvas de Bézier, pode-se gerar a trajetória por intermédio da utilização de B-Splines Cúbicas. Ora, de um modo geral, as B-Splines são junções de curvas de Bézier, que tem em conta a suavidade e a continuidade na junção das curvas.

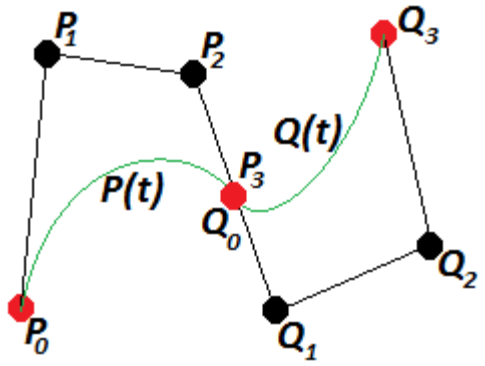

Figura 9 - Junção de Curvas de Bézier.

A dificuldade na união de curvas reside precisamente no ponto de junção de ambas $(P_3; Q_0)$, pois é necessário não só garantir a união e a continuidade entre ambas as curvas, como também é importante, para o caso da trajetória, garantir a suavidade da mesma.

Uma *B-Spline* exibe continuidade em diferentes níveis:

- C^0 – simplesmente contínua no ponto (continuidade posicional);
- C^1 - C^0 e a 1ª derivada é também contínua;
- C^2 - C^1 e a 2ª derivada é também contínua.

Sendo uma junção de curvas de *Bézier* é previsível que as definições (analítica e geométrica das *B-Splines* sejam semelhantes às anteriormente mostradas:

- **<u>Definição geométrica das *B-Splines*</u>**:

  Se tivermos *n+1* pontos de controlo $(P_0, P_1, \dots, P_n)$, uma ordem *k* e um conjunto de nós $(t_0, t_1, \dots, t_{n+k})$ então:

$$P(t) = P_l^{(k-1)}(t) \quad com\ \ t \in [t_l, t_{l+1}] \tag{50}$$

$$onde: P_i^{(j)} = \begin{cases} \left(1 - \tau_i^j\right).P_{i-1}^{(j-1)}(t) + \tau_i^j P_i^{(j-1)}(t) & se \quad j > 0 \\ P_i \qquad\qquad se\ j = 0 \end{cases} \tag{51}$$

$$e\ \tau_i^j = \frac{t - t_i}{t_{i+k-j} - t_i} \tag{52}$$

- **<u>Definição analítica das *B-Splines*</u>**:

  Se temos *n+1* pontos de controlo $(P_0, P_1, \dots, P_n)$, ordem *k* e um conjunto de nós $(t_0, t_1, \dots, t_{n+k})$ então:

$$P(t) = \sum_{i=0}^{n} P_i . N_{i,k}(t) \tag{53}$$

$$onde: \quad N_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} . N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} . N_{i+1,k-1}(t) \tag{54}$$

$$e: \quad N_{i,1}(t) = \begin{cases} 1 & se\ t \in [t_i, t_{i+1}] \\ 0 & de\ outra\ forma \end{cases} \qquad com\ t \in [t_{k-1}, t_{n+1}] \tag{55}$$

As funções *N* são as chamadas funções de mistura, e à semelhança do que foi feito para as curvas de *Bézier* também se pode organizar a estrutura de cálculo das *B-Splines* em forma de pirâmide.

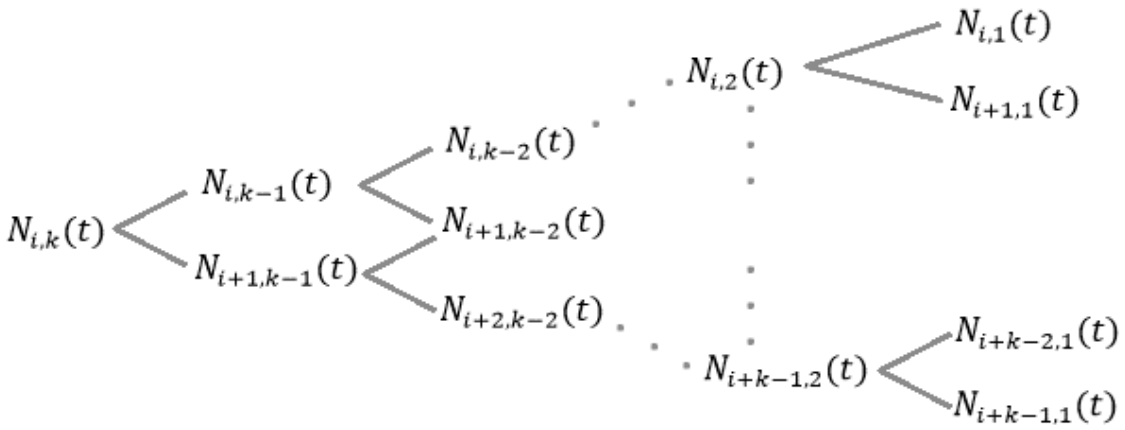

Figura 10 - Esquema piramidal das funções de mistura de B-Splines.

Como se pode observar é necessário calcular várias funções de mistura num *loop* o que faz com que este seja um método que consuma uma capacidade computacional maior que o método anterior das *splines cúbicas*, fazendo com que se torne obsoleto para utilização rápida, consumindo demasiados recursos caso se pretenda aumentar as distâncias entre *waypoints*.

2.2.4. Método de *Overhauser*:

A.W. Overhauser desenvolveu um método para interpolar pontos específicos numa curva, garantindo continuidade do declive em todas as junções. Esse método envolve a interpolação de conjuntos de três pontos com recurso uma parábola.

Considerando 4 pontos:

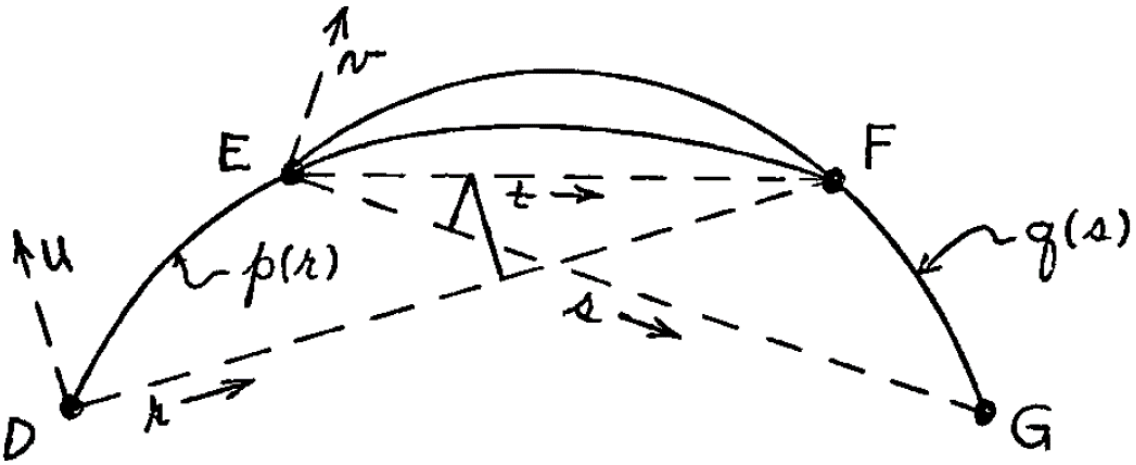

Figura 11 - Esquema do método Overhauser para 4 pontos. [16]

Como se pode ver na figura acima, os pontos *D, E, F* definem a parábola $\vec{p}(r)$, onde *r* é a distância ao longo da corda entre *D* e *F*. Assim, ficamos com a fórmula dessa primeira parábola:

$$u = \alpha . r . (d - r) \tag{56}$$

E de forma similar, os três pontos *E, F, G* definem a parábola $\vec{q}(s)$:

$$v = \beta . s . (e - s) \tag{57}$$

Aqui facilmente se percebe que se os 4 pontos fossem colineares, em vez de uma parábola ter-se-ia uma linha reta.

A zona crucial do esquema acima, é claramente entre os pontos *E* e *F*, pois é nessa zona que teremos a junção entre as duas parábolas. Essa junção é definida pela equação $\vec{c}(t)$:

$$\vec{c}(t) = \left[1 - \left(\frac{t}{dist}\right)\right] . \vec{p}(r) + \left(\frac{t}{dist}\right) . \vec{q}(s) \tag{58}$$

Onde *dist* é a distância entre os pontos *E* e *F*. [16]

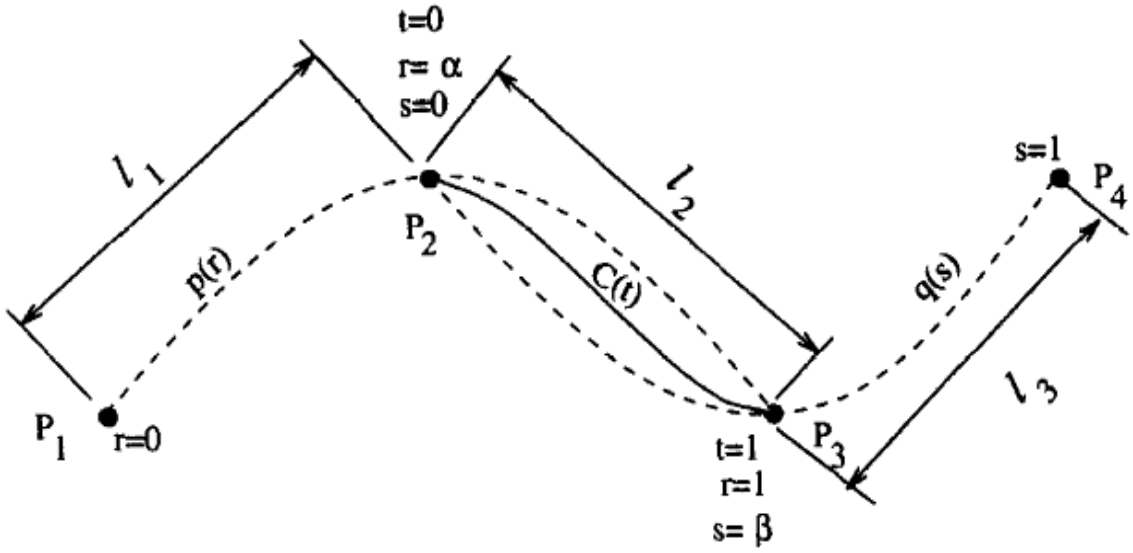

Figura 12 - Esquema ilustrativo do "blending" do método Overhauser. [20]

Fazendo uma adaptação deste método para 4D e generalizando de modo a que possa ser adaptado para qualquer número de pontos (*n* pontos, *n-1* segmentos), fica estruturado da seguinte forma:

- 1º segmento: $\begin{cases} x = a_1^1.t^2 + a_2^1.t + a_3^1 \\ y = b_1^1.t^2 + b_2^1.t + b_3^1 \\ z = c_1^1.t^2 + c_2^1.t + c_3^1 \end{cases}$ **(59)**
- Segmentos intermédios $1 < k \leq n+1$:

$$\begin{cases} x = \left(1 - \frac{t}{dist}\right).x_k + \left(\frac{t}{dist}\right).x_{k+1} \\ y = \left(1 - \frac{t}{dist}\right).y_k + \left(\frac{t}{dist}\right).y_{k+1} \\ z = \left(1 - \frac{t}{dist}\right).z_k + \left(\frac{t}{dist}\right).z_{k+1} \end{cases} \quad \textbf{(60)}$$

$$com: \begin{cases} x_k = a_1^k.t^2 + a_2^k.t + a_3^k \\ x_k = a_1^{k+1}.t^2 + a_2^{k+1}.t + a_3^{k+1} \end{cases} \text{ e de forma semelhante para } y \text{ e } z. \quad \textbf{(61)}$$

- Último segmento: $\begin{cases} x = a_1^{n-1}.t^2 + a_2^{n-1}.t + a_3^{n-1} \\ y = b_1^{n-1}.t^2 + b_2^{n-1}.t + b_3^{n-1} \\ z = c_1^{n-1}.t^2 + c_2^{n-1}.t + c_3^{n-1} \end{cases}$ **(62)**

Tabela 5 - Coeficientes Polinomiais a determinar no cálculo das parábolas do método de *Overhauser*.

Coeficientes Polinomiais *Overhauser*			
Coeficientes da coordenada *x*	$a_0^{(k)}$	$a_1^{(k)}$	$a_2^{(k)}$
Coeficientes da coordenada *y*	$b_0^{(k)}$	$b_1^{(k)}$	$b_2^{(k)}$
Coeficientes da coordenada *z*	$c_0^{(k)}$	$c_1^{(k)}$	$c_2^{(k)}$

Desta forma, vemos que o método de Overhauser tem uma implementação simples e com menos coeficientes para cálculo, comparativamente ao primeiro método descrito neste capítulo – método das *Splines Cúbicas*.

Tomando como exemplo, os *waypoints* descritos anteriormente (rota Porto – Madrid), e implementando o método de *Overhauser* no *MATLAB*®, os seguintes resultados são obtidos:

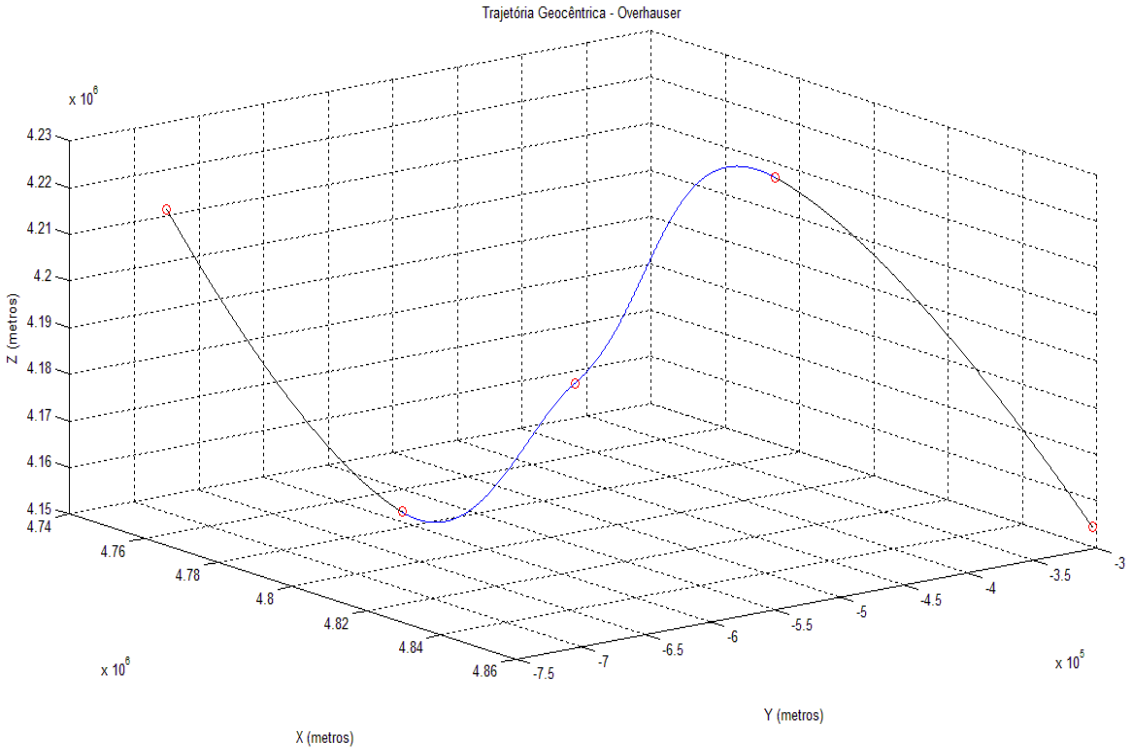

Figura 13 - Trajetória obtida com o método de Overhauser para os waypoints exemplo.

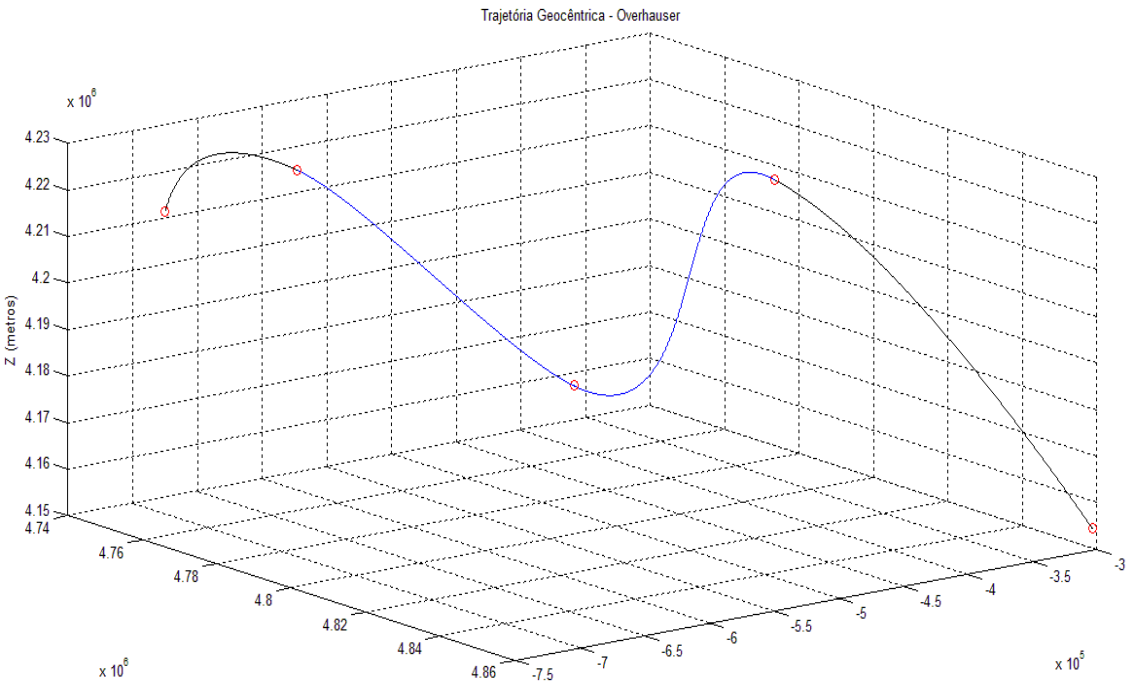

Figura 14 - Trajetória obtida com o método de Overhauser para waypoints exemplo com alteração no 2º waypoint.

Como se pode observar, pelo método de *Overhauser* o controlo é mais local, isto é, a alteração de um dos pontos não afeta necessariamente toda a curva, ficando a área afetada mais cingida à vizinhança do ponto alterado. Fica também confirmado a maior suavidade da trajetória, não se desviando tanto dos *waypoints* por onde é requerida que passe, como foi possível observar no exemplo da trajetória do método da *Spline Cúbica*. Esta caraterística inerente a este método deve-se sobretudo ao *blending* descrito anteriormente, o que permite criar uma trajetória suave, mesmo sendo esta composta por diferentes parábolas, pois o facto de haver mistura de parábolas entre os *waypoints* intermédios suaviza a transição entre elas, criando então a trajetória que se observa.

Conforme previsto, este é também um método rápido, sendo em média 4 vezes mais rápido que o método que utiliza *splines cúbicas*.

Capítulo 3 - Modelação de Geração Ótima de Trajetórias

3.1. Modelação do problema da geração de trajetórias de comprimento mínimo

Ora, conforme descrito anteriormente o objetivo principal é a geração de uma trajetória minimizadora de atrasos, ou seja, de comprimento mínimo. A trajetória atual, conforme descrita acima, é gerada com a obrigação de passar por cada *waypoint*, sendo então uma interpolação, pelo que não se pode considerar como estando otimizada. Para que seja possível obter uma trajetória de comprimento mínimo, e consequentemente minimizar os atrasos há que alterar as restrições, de modo a que o método seja aproximador.

Se considerarmos o primeiro *waypoint* como o aeroporto de partida, e o último *waypoint* como o de chegada, associamos desde já a hipótese de que são de passagem obrigatória, isto é, a trajetória terá necessariamente de passar por estes dois *waypoints*, no entanto, não é importante que passe exatamente nas coordenadas dos *waypoints* intermédios, e aqui está a chave de todo o processo de otimização. A ideia passa por criar uma espécie de esfera em torno de cada um desses *waypoints*, permitindo a aeronave passar a uma certa distância, desde que dentro da esfera, criando a possibilidade de ajustar a trajetória para que seja versátil ao ponto de minimizar o comprimento, conforme se pode ver no esquema 2D a seguir demonstrado, onde a linha de cor verde representa a trajetória otimizada e com um comprimento menor que o da homóloga de cor preta.

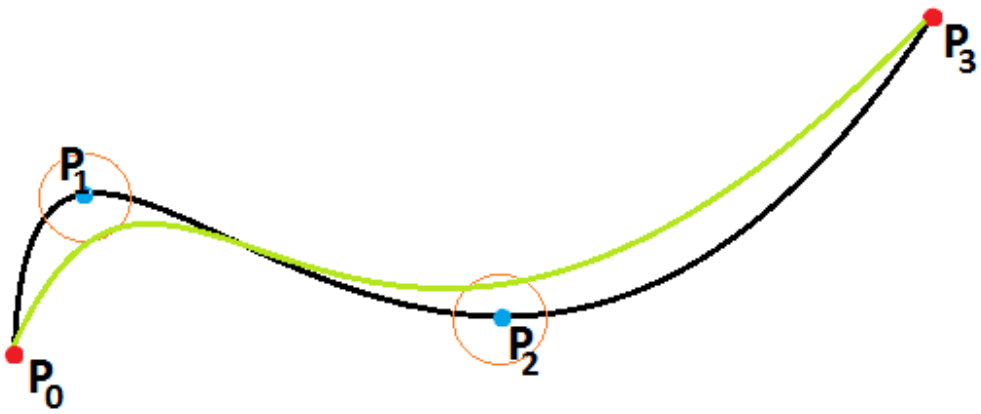

Figura 15 - Esquema ilustrativo da otimização em 2D.

Portanto está agora claro que o problema passa por definir uma trajetória que passe por *N waypoints* (P_0, P_1, …, P_k), enquanto o comprimento dessa mesma trajetória é minimizado:

$$L_{0,k} = \int_{t_0}^{t_k} \|\dot{c}(t)\| dt = J(\theta) \tag{63}$$

O objetivo passa por minimizar a função $J(\theta)$ sujeita às seguintes condições:

- $P_0 = c(t_0)$ **(64)**
- $P_k = c(t_k)$ **(65)**
- $\|P_i - c(t_i)\|^2 \leq r^2, \quad com\ 0 < i < k$ **(66)**

Como já foi visto anteriormente, a trajetória é gerada por segmentos de três pontos que depois são unidos, assim sendo também a função $J(\theta)$ terá de ser tratada como uma função por partes, onde cada parte da função contempla um segmento passante por 3 *waypoints* (para simplificar $\varphi(t) = \|\dot{c}(t)\|$):

$$J(\theta) = \int_{t_0}^{t_k} \varphi(t)\, dt = \int_{t_0}^{t_2} \varphi_1(t)\, dt + \int_{t_1}^{t_3} \varphi_2(t)\, dt + \cdots + \int_{t_{k-2}}^{t_k} \varphi_k(t)\, dt \tag{67}$$

Nota: $J(\theta)$ representa as diferentes funções de integração em função de t, isto é, $f(x)$, $f(y)$ e $f(z)$.

Para calcular o comprimento de um arco seja de parábola ou de uma spline cúbica, a abordagem é a seguinte:

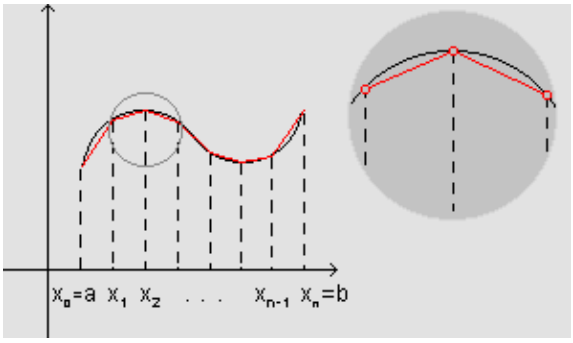

Figura 16 - Abordagem para cálculo do comprimento da função. [18]

Considerando a partição assinalada, temos que o comprimento é dado por:

$$L = \lim_{n\to\infty} \sum_{i=1}^{n} |P_{i-1}P_i| \tag{68}$$

$$|P_{i-1}P_i| = \sqrt{\Delta x_i^2 - \Delta y_i^2} = \sqrt{(x_i - x_{i-1})^2 + \left(f(x_i) - f(x_{i-1})\right)^2}$$
$$= \sqrt{(x_i - x_{i-1})^2 \left[1 + \frac{\left(f(x_i) - f(x_{i-1})\right)^2}{(x_i - x_{i-1})^2}\right]} \tag{69}$$

Pelo Teorema do Valor Médio [17], como f é contínua e com derivada contínua, podemos escrever:

$$f(x_i) - f(x_{i-1}) = f'(\bar{x}_i) \,.\, (x_i - x_{i-1}) \leq> f'(\bar{x}_i) = \frac{f(x_i) - f(x_{i-1})}{(x_i - x_{i-1})}, \tag{70}$$

$$onde\ x_{i-1} \leq \bar{x}_i \leq x_i \tag{71}$$

Assim fica:

$$|P_{i-1}P_i| = \sqrt{1 + [f'(\bar{x}_i)]^2}.\Delta x_i \tag{72}$$

Desta forma o comprimento da curva é dado por:

$$L = \int_a^b \sqrt{1 + [f'(x)]^2}.dx \text{ [18]} \tag{73}$$

3.1.1. Algoritmo de geração de trajetórias de comprimento mínimo

O presente algoritmo foi somente aplicado a dois dos métodos anteriormente descritos, o *Método de Overhauser* e o *Método da Spline Cúbica*, por serem os mais versáteis para a tarefa presente, e que oferecem mais garantias de cumprirem os requisitos estabelecidos, por motivos já apresentados. Prevê-se um comportamento mais versátil no caso da *spline cúbica* neste

ajuste, uma vez que contém mais um coeficiente que o polinómio de segundo grau de *Overhauser*.

Para a minimização da função $J(\theta)$ utiliza-se o comando ***fmincon*** da *Optimization Toolbox* do *MATLAB*® que de uma forma geral carateriza-se pela minimização de funções tendo em conta determinadas restrições:

$$minimizar\ J(\theta)\ de\ modo\ a\ que: \begin{cases} c(\theta) \leq 0 \\ ceq(\theta) = 0 \\ A.\theta \leq b \\ Aeq.\theta = beq \\ lb \leq \theta \leq ub \end{cases} \quad [19] \qquad \textbf{(74)}$$

O algoritmo é então descrito da seguinte forma:

1. Conforme descrito previamente em **2.2.1** e em **2.2.4** os coeficientes dos polinómios (tabelas 4 e 5) para o caso da interpolação são calculados por resolução de um sistema. Para cada segmento *k* os coeficientes da interpolação (*a*, *b* e *c* para o caso do método de *Overhauser* e *a*, *b*, *c* e *d* para o caso do método da *Spline Cúbica*) irão servir como o vetor *Coef_i* de partida para a minimização, ficando o comando da seguinte forma: *New_Coef=fmincon(@function, Coef_i, [], [], [], [], [], [],@constraints);*

2. A função a minimizar (*@function*) é dada por $L = \int_i^{i+2} \sqrt{1 + [f'(t)]^2}.dt$ (integral obtido pela quadratura de Gauss-Kronrod – comando ***quadgk*** do *MATLAB*®), onde $f'(t)$ é:

- no caso do método de *Overhauser*: $f'(t) = \sqrt{(2.a.t + b)^2 + 1}$ **(75)**
- método da *spline cúbica*: $f'(t) = \sqrt{(3.a.t^2 + 2.b.t + c)^2 + 1}$ **(76)**

3. As restrições a ter em conta (*@constraints*) dependem do segmento onde o algoritmo se encontra:

- Para o caso do método de *Overhauser*:
  - Segmento 1: $\begin{cases} P_0 = f_1(t_0) \\ P_1^2 - f_1(t_1)^2 \leq r^2 \\ P_2^2 - f_1(t_2)^2 \leq r^2 \end{cases}$ **(77)**
  - Segmentos intermédios ($1 < int < n$): $\begin{cases} P_i^2 - f_{int}(t_i)^2 \leq r^2 \\ P_{i+1}^2 - f_{int}(t_{i+1})^2 \leq r^2 \\ P_{i+2}^2 - f_{int}(t_{i+2})^2 \leq r^2 \end{cases}$ **(78)**

- Último segmento *n*: $\begin{cases} P_j^2 - f_n(t_j)^2 \leq r^2 \\ P_{j+1}^2 - f_n(t_{j+1})^2 \leq r^2 \\ P_{j+2} = f_n(t_{j+2}) \end{cases}$ **(79)**

- No caso do método da *spline cúbica*, existem **condições extra**, pois ao contrário do método de *Overhauser*, não existe uma "costura de funções", isto é, na exploração dos segmentos a segunda *spline* começa onde a primeira termina, e assim sendo é necessário garantir a continuidade entre funções:

  - Segmento 1: $\begin{cases} P_0 = f_1(t_0) \\ P_1^2 - f_1(t_1)^2 \leq r^2 \\ P_2^2 - f_1(t_2)^2 \leq r^2 \end{cases}$ **(80)**
  - Segmentos intermédios ($1 < int < n$): $\begin{cases} f_{int}(t_i) = f_{int-1}(t_i) \\ f'_{int}(t_i) = f'_{int-1}(t_i) \\ P_{i+1}^2 - f(t_{i+1})^2 \leq r^2 \\ P_{i+2}^2 - f(t_{i+2})^2 \leq r^2 \end{cases}$ **(81)**
  - Último segmento *n*: $\begin{cases} f_n(t_j) = f_{n-1}(t_j) \\ f'_n(t_j) = f'_{n-1}(t_j) \\ P_{j+1}^2 - f_n(t_{j+1})^2 \leq r^2 \\ P_{j+2} = f_n(t_{j+2}) \end{cases}$ **(82)**

  Nota 1: O número de segmentos (*k*) é dado pelo número de *waypoints*-2;
  Nota 2: O raio *r* é determinado consoante o valor em metros, da distância a que se acha permissível a aeronave passar, em relação a um *waypoint* intermédio.

4. Tendo calculado os novos coeficientes (apresentados nas tabelas 4 e 5) pode então ser gerada a nova trajetória aproximada e otimizada.

3.1.2. Exemplos numéricos (Simulação 1)

Com recurso aos *waypoints* acima descritos (plano de voo Porto – Madrid) e com base nos dados caraterísticos de uma das configurações da aeronave *"Olharapo"*, o algoritmo acima descrito é então simulado em *MATLAB*® de forma a se proceder não só à geração da trajetória por interpolação, bem como a geração da trajetória por aproximação, e consequentemente otimizada. Esta simulação foi realizada com o *Método de Overhauser* e com o *Método da Spline Cúbica*.

- Método de Overhauser

O raio da chamada esfera permissível utilizado neste exemplo foi de 5000 metros, ou seja, a aeronave pode passar com uma distância até 5 km dos *waypoints* B, C e D.

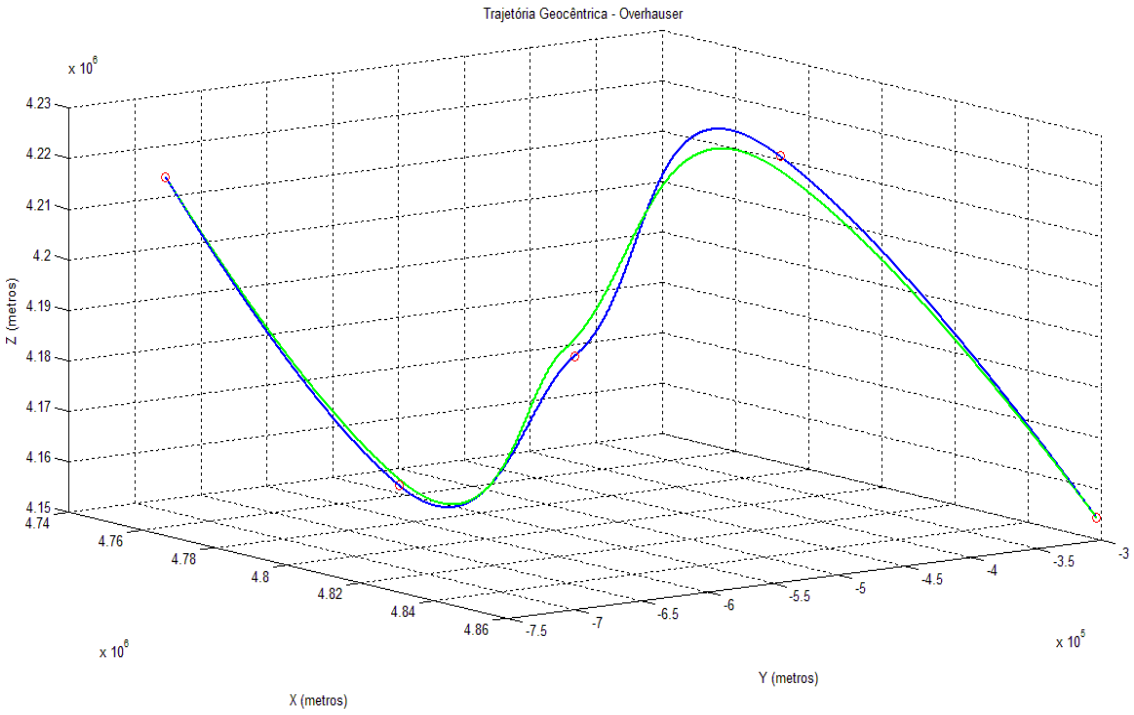

Figura 17 - Trajetória Geocêntrica - Overhauser otimizada.

Como podemos ver na figura, a cor **azul** está a trajetória interpoladora, que passa obrigatoriamente pelos *waypoints* e a cor **verde** está a trajetória otimizada que apenas tem a obrigatoriedade de passar pelos *waypoints* inicial e final. Esta trajetória aproximada tem um menor comprimento em relação à anterior, o que demonstra que a minimização dos atrasos foi alcançada. No **Anexo A** estão, disponíveis para consulta, as tabelas com os valores dos coeficientes antes e após a minimização (coeficientes da interpolação e da aproximação, respetivamente), bem como 3 vistas da trajetória.

Para um melhor entendimento e análise deste método, foi acrescentada um cálculo dos comprimentos das trajetórias obtidas, que recorre a um método de integração numérica – a fórmula quadrática de Gauss – Kronrod (comando ***quadgk*** no *MATLAB*®) - de forma a ser possível calcular a redução atingida:

Tabela 6 - Comparação de comprimentos de trajetória interpoladora e aproximadora - Método Overhauser (raio permissível de 5 km).

Comprimento da Trajetória Interpoladora (km)	**Comprimento da Trajetória Aproximadora (km)**	**Redução (%)**
525,00	513,11	2,26

Uma vez que o raio permissível é de apenas 5 km, a redução de 2.26% que este método alcança ao aproximar a trajetória é já algo significativa, principalmente se tivermos em conta as limitações do método, pelo facto de ser um polinómio de grau 2, pelo que a sua versatilidade é pequena. Para demonstrar a efetividade desta aproximação no que toca à redução do comprimento da trajetória, num ensaio suplementar, aumentou-se o raio permissível para 10 km:

Tabela 7 - Comparação de comprimentos de trajetória interpoladora e aproximadora - Método Overhauser (raio permissível de 10 km).

Comprimento da Trajetória Interpoladora (km)	**Comprimento da Trajetória Aproximadora (km)**	**Redução (%)**
525,00	502,97	4,20

A redução neste caso é ligeiramente maior, como naturalmente esperado, no entanto a suavidade da trajetória do método é colocada em causa, pois como se pode ver no gráfico acima, na zona do 3º *waypoint* a trajetória apresenta uma mudança de direção algo agressiva.

Apenas de notar que, em ambas as simulações aqui apresentadas, foi permitido ao *MATLAB*® correr com: o número base de iterações (900); e com um número máximo de 3000 iterações. Entre a simulação com 900 iterações e a simulação com 3000 iterações há uma diferença na ordem da 5ª casa decimal de redução percentual do comprimento da trajetória, e os dados do *output* do *fmincon* revelam que a minimização de segmento que mais iterações consumiu até convergir ficou-se pelas 1820, pelo que torna-se desnecessário correr com mais iterações.

A diferença do aumento do grau polinomial torna possível uma trajetória mais flexível, algo que está comprovado em seguida com a simulação feita para o método da *Spline Cúbica*.

- Método da *Spline Cúbica*

Também neste método utilizou-se um raio de esfera permissível de 5000 metros para os *waypoints* B, C e D.

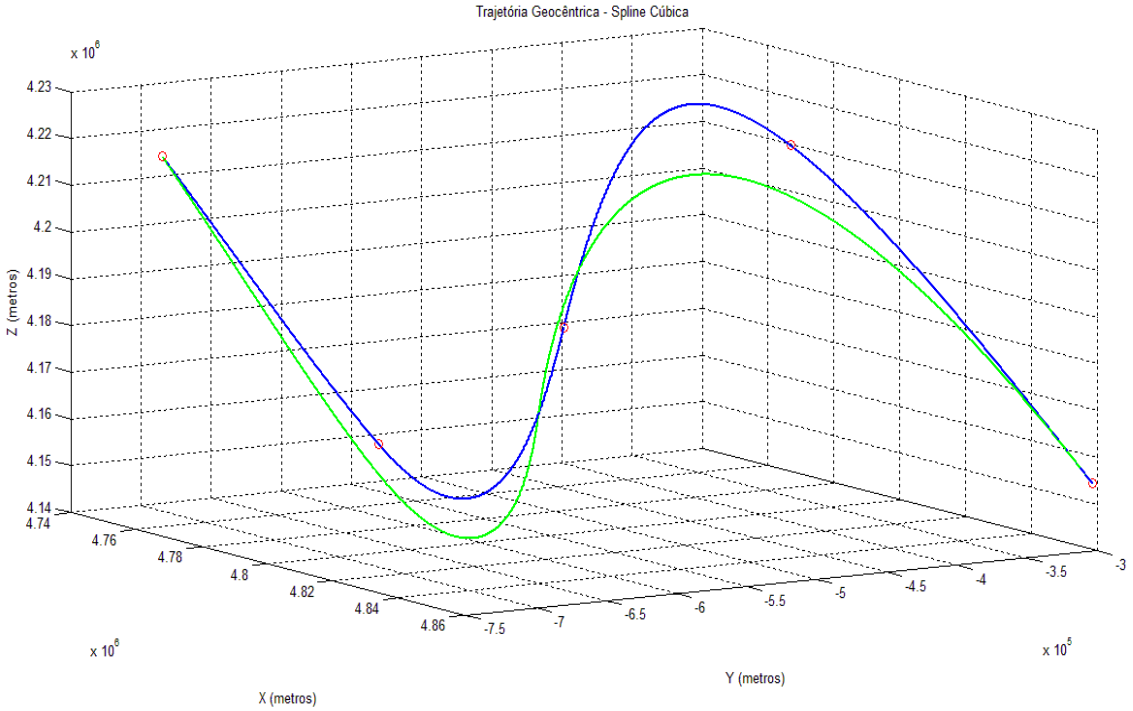

Figura 18 - Trajetória Geocêntrica - Spline Cúbica otimizada.

Podemos ver que a trajetória traçada com cor **verde** é a trajetória de menor comprimento, e portanto é a versão otimizada do trajeto a seguir.

Os coeficientes obtidos quer para a interpolação (**azul**) quer para a aproximação (**verde**) estão apresentados no **Anexo A**, bem como três vistas da trajetória que vemos acima em 3D.

Logo à primeira vista torna-se evidente a maior flexibilidade do método, uma vez que a aproximação está mais suave neste caso, comparativamente ao caso anterior com o método de *Overhauser*. É também fácil de perceber que a redução do comprimento de trajetória é superior à alcançada com o método anterior, o que pode ser confirmado com os valores calculados dos comprimentos de cada trajetória (quer a interpoladora, quer a aproximadora) de seguida apresentados.

Tabela 8 - Comparação de comprimentos de trajetória interpoladora e aproximadora - Método Spline Cúbica (raio permissível de 5 km).

Comprimento da Trajetória Interpoladora (km)	**Comprimento da Trajetória Aproximadora (km)**	**Redução (%)**
526,14	518,19	1,51

Uma redução na ordem dos 1,51 % é menos significativa que a obtida anteriormente no método de *Overhauser*, tendo em conta que a trajetória aproximadora permite à aeronave passar a um máximo de apenas 5 km de distância do *waypoint* correto. Num outro ensaio, no qual se permite à aeronave uma passagem mais distante (10 km) dos *waypoints* intermédios há, como esperado, uma maior redução do comprimento da trajetória:

Tabela 9 - Comparação de comprimentos de trajetória interpoladora e aproximadora - Método Spline Cúbica (raio permissível de 10 km).

Comprimento da Trajetória Interpoladora (km)	**Comprimento da Trajetória Aproximadora (km)**	**Redução (%)**
526,14	507,36	3,57

Numa análise a ambos os métodos aqui apresentados verifica-se uma desvantagem do método da *Spline Cúbica*: para além dos comprimentos das trajetórias traçadas originalmente (por interpolação) serem extremamente semelhantes, mas com um comprimento ligeiramente menor no caso do método de *Overhauser*, quando se aproxima a trajetória verifica-se a mesma tendência, onde o método da *Spline Cúbica* reduz com menor eficácia o comprimento total da trajetória o que é em certa parte um revés nas expetativas, mas que se deve sobretudo à questão falada no **Capítulo 2** da não existência de *blending* no método da *Spline Cúbica*. O que é visível, e que pode ser registado como uma vantagem do método da *Spline* é a criação uma trajetória mais suave, o que deixa antever que este método produzirá uma melhor solução no caso da existência de um obstáculo ou de uma zona interdita, que torne necessário um recalcular da trajetória num determinado segmento do trajeto.

3.2. Modelação do problema da geração de trajetórias com atenção às velocidades mínima e máxima da aeronave

Na simulação anterior, assume-se a aproximação (como referido no capítulo 2) de que o voo é todo ele nivelado (devido à pequena variação dos ângulos de trajetória), sem preocupação sobre se a velocidade mínima e máxima da aeronave ao longo da trajetória são respeitadas (respetivamente 18 e 30 m/s).

Ora, tendo em conta as fórmulas descritas na **secção "2.1. Plano de Voo"** as velocidades utilizadas para calcular os tempos de passagem em cada *waypoint* são as seguintes:

- *Waypoint 1* (aeroporto de partida): $V = 18\, m/s$ que corresponde à velocidade de descolagem atribuída;
- *Waypoint 2*: $V = 14{,}404\, m/s$;
- *Waypoint 3*: $V = 17{,}783\, m/s$;
- *Waypoint 4*: $V = 14{,}913\, m/s$;
- *Waypoint 5* (aeroporto de chegada): $V = 19\, m/s$ que corresponde à velocidade de aterragem.

Como podemos observar as velocidades com que a aeronave está a passar pelos *waypoints 2,3* e *4* são inferiores à velocidade mínima da aeronave, de modo que é necessário acrescentar mais condições às utilizadas na simulação anterior para que seja então respeitado o intervalo das velocidades da aeronave, e o tempo seja minimizado. Assim sendo é necessário ter a seguinte condição respeitada:

$$V_{mín}^2 \leq V_x^2 + V_y^2 + V_z^2 \leq V_{máx}^2 \quad \textbf{(83)}$$

É importante ter em atenção que anteriormente, os tempos de passagem por cada *waypoint* vinham já definidos do plano de voo, mas tendo como base as velocidades calculadas pelas fórmulas do voo nivelado. Como agora é necessário garantir o respeito pela velocidade mínima e máxima torna-se necessário calcular os tempos de passagem em cada *waypoint* (à exceção do primeiro, pois é igual a 0 segundos) dentro da minimização.

Desta forma, as condições-extra que são acrescentadas ao algoritmo anteriormente descrito, por causa da velocidade, são as seguintes:

- Segmento 1 $(P_1; P_2; P_3)$: $\begin{cases} V_{mín}^2 \leq Vx_1^2 + Vy_1^2 + Vz_1^2 \leq V_{máx}^2 \\ V_{mín}^2 \leq Vx_2^2 + Vy_2^2 + Vz_2^2 \leq V_{máx}^2 \\ V_{mín}^2 \leq Vx_3^2 + Vy_3^2 + Vz_3^2 \leq V_{máx}^2 \end{cases}$ **(84)**

- Segmentos intermédios $(P_k; P_{k+1}; P_{k+2})$: $\begin{cases} V_{mín}^2 \leq Vx_k^2 + Vy_k^2 + Vz_k^2 \leq V_{máx}^2 \\ V_{mín}^2 \leq Vx_{k+1}^2 + Vy_{k+1}^2 + Vz_{k+1}^2 \leq V_{máx}^2 \\ V_{mín}^2 \leq Vx_{k+2}^2 + Vy_{k+2}^2 + Vz_{k+2}^2 \leq V_{máx}^2 \end{cases}$ **(85)**

- Último segmento $(P_{n-2}; P_{n-1}; P_n)$: $\begin{cases} Vx_n^2 + Vy_n^2 + Vz_n^2 = V_{Land}^2 \\ V_{mín}^2 \leq Vx_{n-1}^2 + Vy_{n-1}^2 + Vz_{n-1}^2 \leq V_{máx}^2 \\ V_{mín}^2 \leq Vx_{n-2}^2 + Vy_{n-2}^2 + Vz_{n-2}^2 \leq V_{máx}^2 \end{cases}$ **(86)**

Nota: foi tomada a opção de dar a liberdade à minimização de definir a velocidade de descolagem, de forma a evitar possíveis demasiados constrangimentos no primeiro segmento (que afeta toda a restante trajetória).

Visto que é necessário o cálculo dos tempos de passagem por *waypoint*, há uma condição que necessita de ser acrescentada para todos os segmentos, que garante que o tempo é crescente:

$$t_i < t_{i+1} \quad \textbf{(87)}$$

Em relação à fórmula das componentes da velocidade, sabemos que esta é dada pela derivada da posição, pelo que:

- no caso de *Overhauser* temos:

$$\begin{cases} V_x = 2.a_1.t + a_2 \\ V_y = 2.b_1.t + b_2 \\ V_z = 2.c_1.t + c_2 \end{cases} \quad \textbf{(88)}$$

- no caso da *Spline* Cúbica temos:

$$\begin{cases} V_x = 3.a_3.t^2 + 2.a_2.t + a_1 \\ V_y = 3.b_3.t^2 + 2.b_2.t + b_1 \\ V_z = 3.c_3.t^2 + 2.c_2.t + c_1 \end{cases} \quad \textbf{(89)}$$

3.2.1. Exemplos numéricos (Simulação 2)

Apresentam-se em seguida os resultados obtidos na simulação efetuada com as alterações descritas anteriormente de forma a garantir o cumprimento da velocidade mínima e máxima.

- Método de *Overhauser*:

Mais uma vez, utilizou-se o raio permissível igual a 5000 metros, e sobrepôs-se o gráfico da trajetória que cumpre os requisitos da velocidade às trajetórias anteriormente existentes:

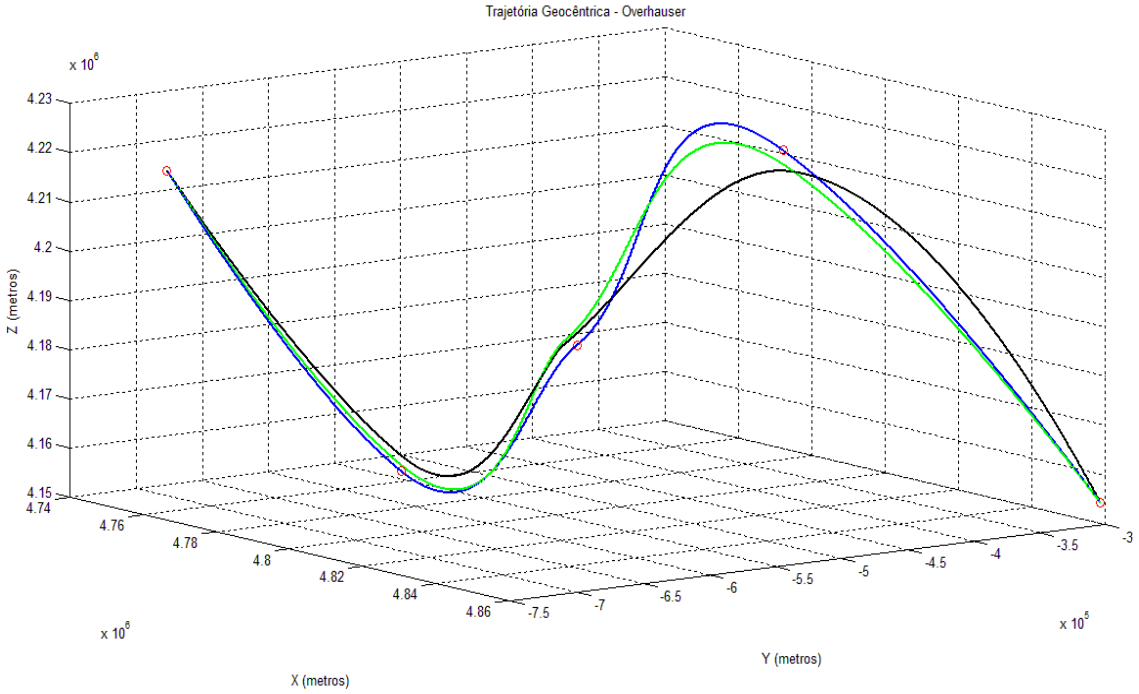

Figura 19 - Trajetória Geocêntrica - Overhauser com respeito à velocidade mínima e máxima.

Como podemos ver na figura, a cor **preta** está a trajetória que otimiza o comprimento mínimo, mas desta feita respeitando a velocidade mínima e máxima da aeronave. Olhando para a figura vê-se que esta nova trajetória tem um comprimento ainda mais pequeno comparativamente às anteriores, tal como seria de esperar, uma vez que a minimização teve mais condições, permitindo assim um maior rigor nos coeficientes polinomiais obtidos que não só minimizaram o comprimento da trajetória, como também diminuíram o tempo em que esta foi realizada. Foi permitido ao programa correr até um máximo de 5000 iterações, mas após verificar o número utilizado vê-se que este ficou pelas 2659 iterações. No **Anexo B** estão, disponíveis para consulta, as tabelas com os valores dos coeficientes antes e após a minimização (coeficientes da interpolação e da aproximação, respetivamente), bem como 3 vistas da trajetória.

Apresenta-se agora os comprimentos das trajetórias obtidas, bem como uma tabela dos novos valores dos tempos de passagem por cada *waypoint*, e das respetivas velocidades de passagem:

Tabela 10 - Comparação de comprimentos de trajetória interpoladora e da trajetória que respeita a velocidade - Método Overhauser (raio permissível de 5 km).

Comprimento da Trajetória Interpoladora (km)	**Comprimento da Trajetória Aproximadora (km)**	**Redução (%)**
525,00	478,89	8,78

Tabela 11 - Tempos de passagem em cada waypoint, com respeito aos limites de velocidade.

Tempo "waypoint" A	0 segundos
Tempo "waypoint" B	3504 segundos
Tempo "waypoint" C	8097 segundos
Tempo "waypoint" D	14332 segundos
Tempo "waypoint" E	21853 segundos

Tabela 12 - Velocidades de passagem em cada waypoint.

Velocidade "waypoint" A	29 m/s
Velocidade "waypoint" B	18 m/s
Velocidade "waypoint" C	27,38 m/s
Velocidade "waypoint" D	21,22 m/s
Velocidade "waypoint" E	19 m/s

Conforme se pode observar pelos dados recolhidos, não só a redução do comprimento de trajetória melhorou significativamente face à simulação anterior, onde não se tem em conta os limites de velocidade, (a redução tinha sido de 2,2%) como também o tempo de passagem em cada *waypoint* reduziu, e o percurso demoraria agora cerca de 21853 segundos a ser percorrido, o que corresponde a uma redução de 44,32% do tempo calculado inicialmente considerando voo nivelado, sem respeitar os limites da velocidade, e sem minimizar o comprimento da trajetória. É também possível verificar que as velocidades de passagem em cada *waypoint* estão agora superiores/iguais à velocidade mínima e inferiores à máxima.

- Método da *Spline Cúbica*:

Conforme descrito na simulação anterior (**secção 3.1.2**) o método da *Spline* Cúbica enfrentava alguns problemas na redução do comprimento de trajetória, apresentando resultados aquém do esperado, quando comparados ao método de *Overhauser*. Na simulação proposta nesta secção, com o introduzir das condições extra das velocidades mínimas e máximas, o *MATLAB*® começou a terminar a minimização (*fmincon*) mais cedo que o esperado, não conseguindo encontrar resultados satisfatórios que conseguissem cumprir com as condições impostas. O *EXITFLAG* – que dá a condição de saída do *fmincon* – tinha valores que variavam entre *-2* e *0*, o que significa *"no feasible point"* e *"too many function evaluations or iterations"* [19], tornando assim claro que da forma como o método está estruturado seria impossível a minimização com as condições atuais.

Ora, sendo impossível uma minimização bem-sucedida com as condições atuais, é também impossível com o adicionar de mais condições, devido à existência de um obstáculo estático na trajetória, pelo que este método torna-se inviável. Possíveis alterações à forma como ele está concebido neste estudo podem levar a uma solução eficaz, nomeadamente o reformular do método para acrescentar um *"function blending"* à semelhança do que é feito no método de *Overhauser* e que está descrito no **capítulo 2**. Esta é uma sugestão que fica deste estudo, para referência de trabalhos futuros.

Capítulo 4 - Deteção e Prevenção de Colisão

Até agora a trajetória era gerada partindo do pressuposto de que não há qualquer obstáculo estático ou zonas interditas entre os *waypoints* escolhidos para este estudo. De forma a compreender se os métodos em análise – *Overhauser* e *Spline Cúbica* – conseguem realizar uma correção na sua trajetória ao encontrarem um obstáculo estático ou uma zona interdita, é aqui neste capítulo proposto um algoritmo de deteção e prevenção de colisão. Este algoritmo concentra-se apenas no caso de uma zona interdita, ou seja, aquando o traçar da trajetória, deteta de imediato se esta interceta ou não uma zona na qual está proibido o voo, por forma a fazer a correção necessária de forma a respeitar essa restrição.

4.1. Deteção de Conflito com Zonas Interditas

Caso haja alguma zona interdita, é necessário que as coordenadas da mesma sejam introduzidas no programa, para que este possa detetar se existe uma interceção da trajetória que gerou segundo os critérios do **capítulo 3**.

Portanto a deteção do conflito procede-se da seguinte forma:

1. O programa tem a trajetória traçada (consoante o que foi visto anteriormente);
2. As coordenadas de uma zona interdita são comunicadas ao programa: $(X_{obst}, Y_{obst}, Z_{obst}, radius_{obst})$ – assumindo que a zona interdita é uma esfera, para fins de simplificação;
3. Procede-se à verificação da existência de um possível conflito:
    - Para cada segmento cria-se um vetor que testa se as coordenadas de cada ponto do segmento estão dentro da esfera:

$$(x - x_{obst})^2 + (y - y_{obst})^2 + (z - z_{obst})^2 \leq totalradius \quad \textbf{(90)}$$

O $totalradius$ tem em conta uma certa distância de segurança em relação ao obstáculo, com base no conceito *Alert Zone*/*Protected Zone* [1]. Desta forma, o que se considera é $totalradius = radius_{obst} + radius_{alert\ zone}$, onde a esfera em torno do obstáculo é assumida como sendo a *Protected Zone*, a zona que não pode ser invadida, cujo raio é dado pelo $radius_{obst}$ e que é somado a um raio de segurança ($radius_{alert\ zone}$), dando assim o raio utilizado nas simulações, o $totalradius$. A figura abaixo ilustra esta explicação.

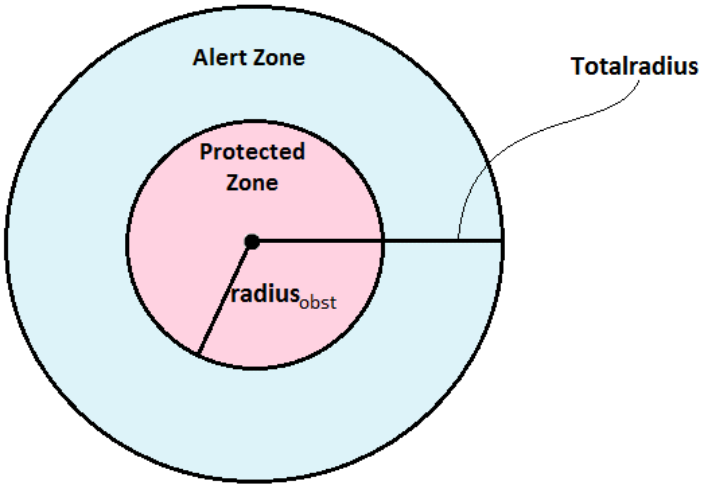

Figura 20 - Esquema do raio de segurança em torno do obstáculo.

- Caso seja encontrado algum/alguns pontos que estejam dentro da esfera interdita, o programa emite um alerta que tem conflito.

4.2. Resolução de Conflito com Zonas Interditas

Após deteção do conflito, é necessário proceder a uma nova minimização do comprimento de trajetória, desta feita com algumas condições extra. Terá que se juntar às condições descritas nas secções **3.1.1.** e **3.2.** para os constrangimentos, o seguinte:

- Primeiramente, criar um vetor (supondo "*inter*"), que a cada valor do tempo *i* (entre o tempo inicial de um segmento e o final, com um *step* aceitável) registe o valor de:

$$(x_i - x_{obst})^2 + (y_i - y_{obst})^2 + (z_i - z_{obst})^2 - radius_{obst}^2 \quad \textbf{(91)}$$

- De seguida, verificar se algum desses valores é menor ou igual a zero:

$$check = any(inter \leq 0) \quad \textbf{(92)}$$

- Esta variável *"check"* assumirá o valor de 1 em caso de conflito, e o valor de 0 em caso de conflito evitado/inexistente, assumindo assim o papel crucial nesta nova minimização, pois passa a ser condição que:

$$check = 0 \quad \textbf{(93)}$$

- A nova minimização deverá ser iniciada com os últimos valores de coeficientes obtidos, isto é, com os valores correspondentes à simulação imediatamente anterior à deteção de conflito.

4.3. Simulação Final de Deteção e Prevenção de invasão de zonas interditas

- Método de *Overhauser*:

Para esta simulação criaram-se duas zonas interditas em dois segmentos distintos de forma a verificar o comportamento do método de *Overhauser* na prevenção. Estas zonas estão aqui representadas como esferas, por uma questão de simplificação nas restrições colocadas. Naturalmente que ao moldar uma zona interdita em torno de um possível obstáculo achatado, seria mais eficaz um cilindro, por exemplo, no entanto isso implicaria mais inequações distintas de restrição para o algoritmo ($(x - x_{cil})^2 + (y - y_{cil})^2 \leq r^2 \wedge z_i \leq z \leq z_s$ - Exemplo de inequações necessárias para um cilindro), e uma vez que interessa verificar o comportamento do método face à prevenção de uma colisão/invasão de zona interdita, optou-se aqui por colocar então uma esfera.

Tabela 13 - Coordenadas das esferas interditas para esta simulação.

Esfera 1 (metros)		Esfera 2 (metros)	
X1	4817342,166	**X2**	4807562,563
Y1	-633951,3987	**Y2**	-476023,3152
Z1	4170455,799	**Z2**	4203055,386
Raio1	1500,053538	**Raio2**	4186,437565

A esfera 1 está sensivelmente a meio da trajetória no segmento 2 (entre o *waypoint 2* e *3*) enquanto a esfera 2 está no segmento 3 (entre o *waypoint 3* e *4*). A resposta do método de *Overhauser* está demonstrada abaixo nas figuras em 3D e com *zoom* efetuado nas áreas onde se encontram as esferas permissíveis.

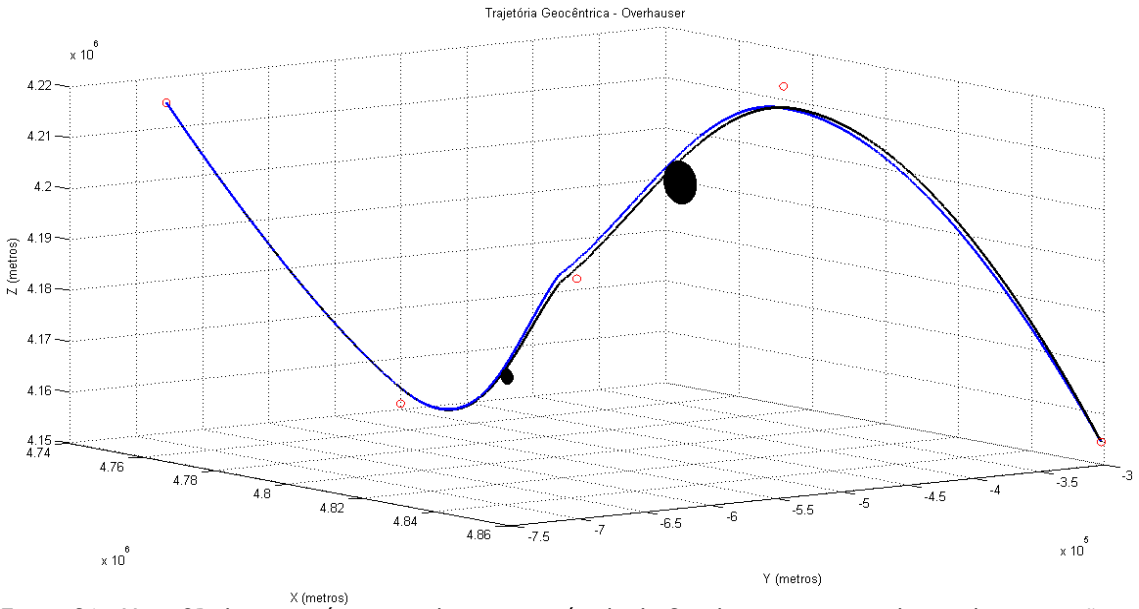

Figura 21 - Vista 3D das trajetórias geradas com o método de Overhauser, antes e depois da prevenção de colisão.

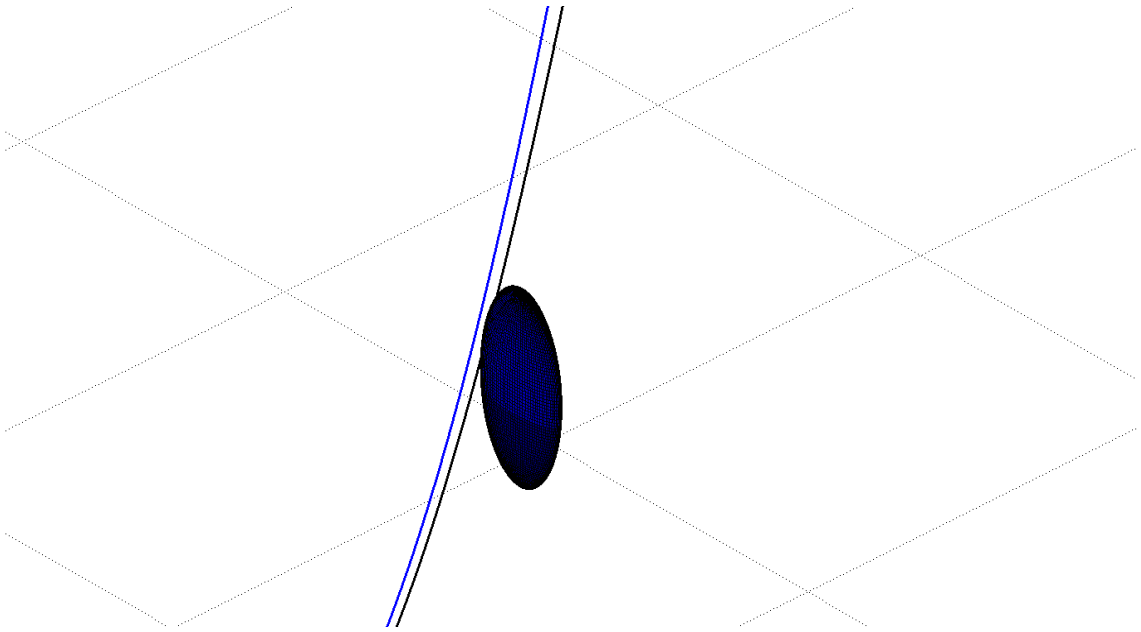

Figura 22 - Pormenor da área onde se encontra a esfera interdita nº1.

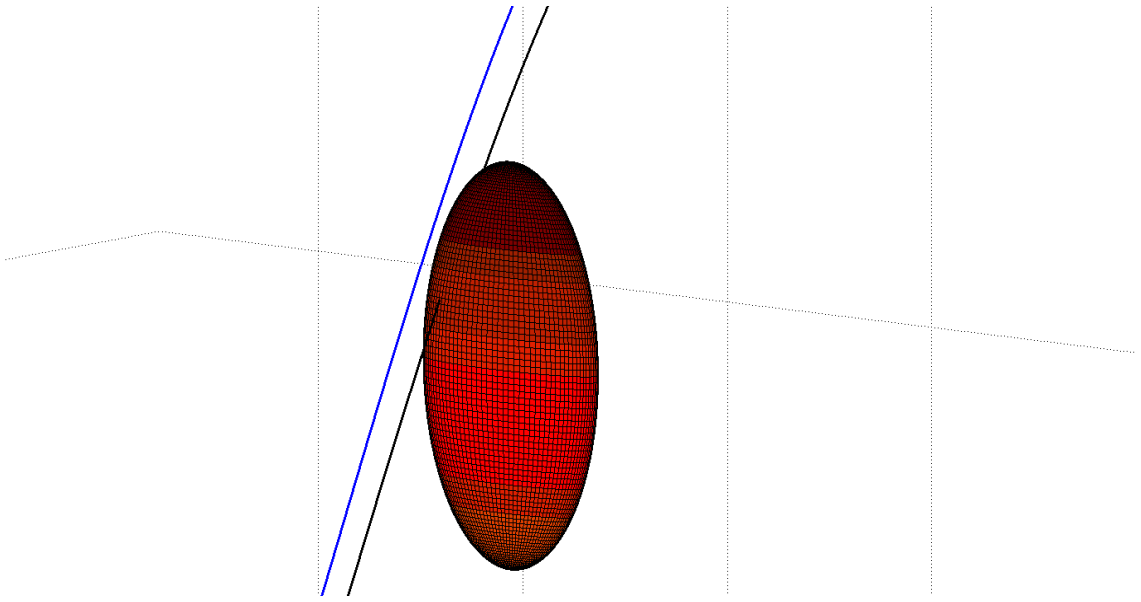

Figura 23 - Pormenor da área onde se encontra a esfera interdita nº2.

Pelas figuras aqui apresentadas, verifica-se que o método de *Overhauser* conseguiu, com sucesso, evitar a colisão com ambas as zonas interditas. Em relação à velocidade de passagem em cada um dos *waypoints*, verificou-se que todas estão a respeitar os limites de velocidade:

Tabela 14 - Velocidades de passagem por cada waypoint, no caso da simulação com obstáculos.

Velocidade "waypoint" A	29 m/s
Velocidade "waypoint" B	18 m/s
Velocidade "waypoint" C	27,12 m/s
Velocidade "waypoint" D	20,44 m/s
Velocidade "waypoint" E	19 m/s

A nível do comprimento da trajetória este tem um ligeiro aumento em relação ao comprimento obtido na **simulação 2**. Anteriormente foi registado um comprimento de 478,89 km, enquanto neste caso o comprimento é de 483,91 km, sensivelmente 1% superior, mas isso seria de esperar uma vez que existem obstáculos a contornar nesta simulação, enquanto o tempo também regista um aumento, neste caso na ordem dos 1,7% (anteriormente era de 21853 segundos, e agora é de 22225 segundos).

Foi efetuado um **segundo teste** ao método onde a única coisa alterada foi o raio da Esfera 1 (passou para o dobro: 3000 metros). Estes foram os resultados obtidos:

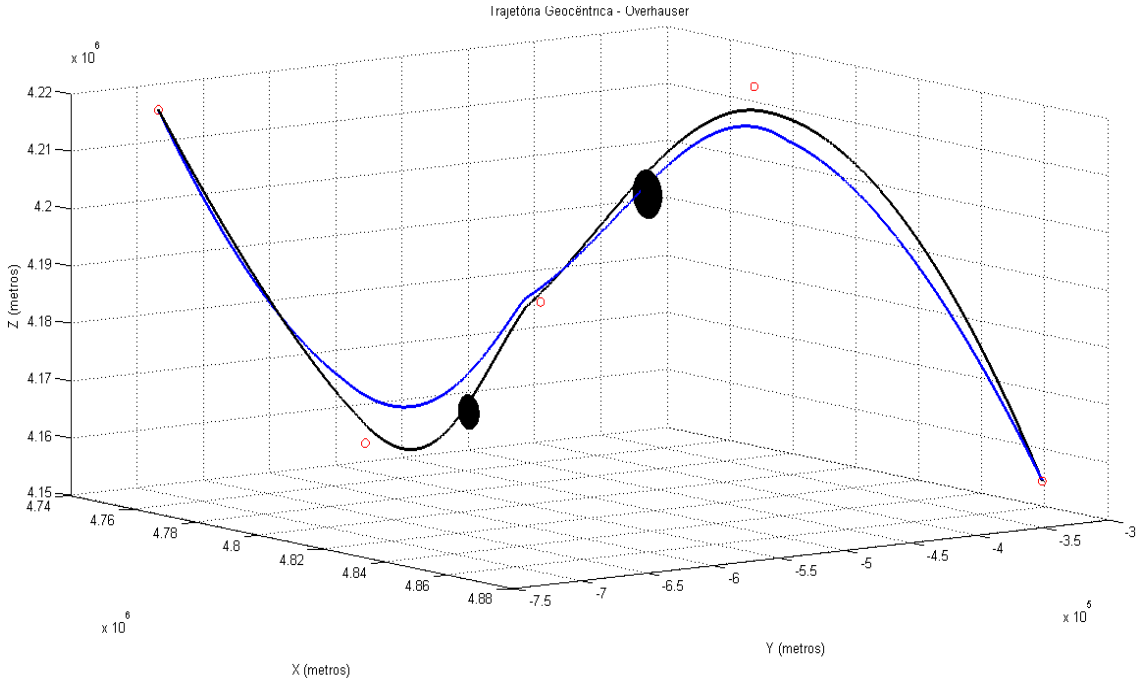

Figura 24 - Vista 3D do segundo teste de prevenção de colisão, feito com o método de Overhauser.

O aumento do raio do 1º obstáculo foi suficiente para que o método de *Overhauser* revela-se as suas limitações, pois apesar de ultrapassar com sucesso o 1º obstáculo, não o consegue fazer perante o 2º. Através do *output* do *fmincon* descobre-se que o valor máximo das iterações realizadas fica-se um pouco acima das 10000, e corresponde às iterações feitas para o 2º segmento. Quer no 1º segmento, quer no 3º segmento a minimização termina abaixo das 500 iterações, com o critério *"no feasible point found"*. Ainda assim, verificou-se que a minimização respeitou as velocidades ao longo da trajetória.

Para esclarecimento de dúvidas sobre a fiabilidade do método, mas apenas com um obstáculo, correu-se ainda um terceiro teste, onde se manteve o segundo obstáculo exatamente com as coordenadas referidas na **tabela 13**.

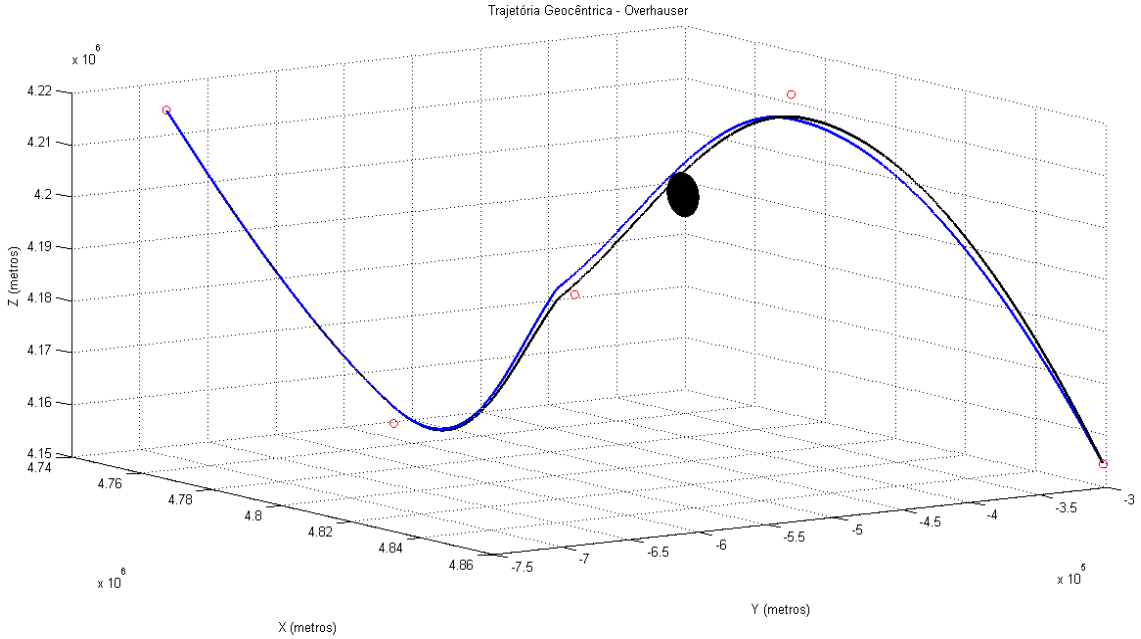

Figura 25 - Vista 3D do terceiro teste de prevenção de colisão, feito com o método de Overhauser.

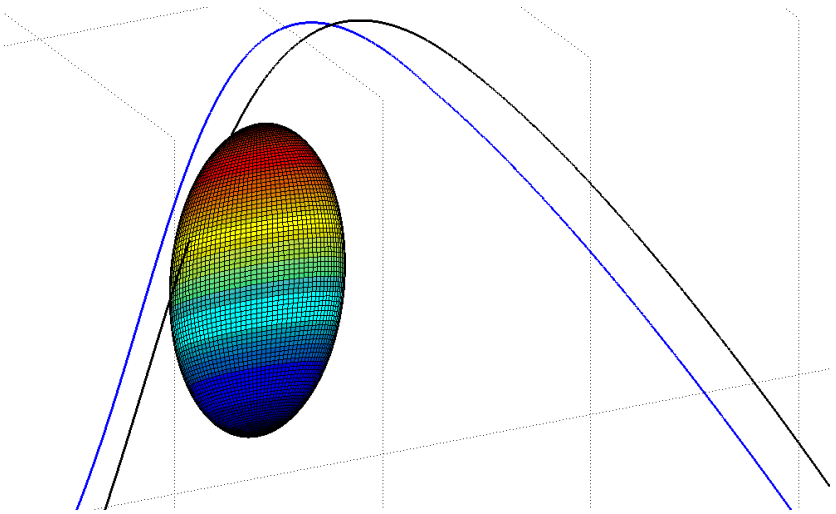
Figura 26 - Pormenor da área onde se encontra a esfera interdita.

Como esperado, a colisão com o obstáculo é evitada, e, à semelhança do que aconteceu no 1º teste desta simulação, as imposições de velocidade são cumpridas com valores extremamente semelhantes aos obtidos no primeiro teste. A diferença entre este teste e o 1º está no comprimento total da trajetória. Ora, sendo que agora só temos um obstáculo a evitar é normal que a trajetória tenha um comprimento menor (483,09 km) que face ao comprimento inicial da interpolação, resulta numa redução da ordem dos 7,9%. Comparativamente ao tempo do percurso total, ficou-se pelos 22080 segundos (pouco mais de 6 horas) apenas menos 140 segundos a menos que o 1º teste.

Este teste definitivo comprova a eficácia do método perante um obstáculo apenas, uma vez que já estava provado o seu sucesso com 2 obstáculos. No entanto, é importante destacar que também com apenas um obstáculo, há a limitação do tamanho do mesmo, isto é, se o obstáculo for muito maior que o que foi utilizado neste teste, o método irá falhar ao evitar a colisão.

No **Anexo C** encontram-se gráficos das 3 vistas, bem como tabelas dos coeficientes obtidos para esta simulação.

Capítulo 5 - Conclusão

5.1. Conclusões do estudo efetuado

Esta dissertação tinha como objetivo principal encontrar um método capaz de gerar uma trajetória em 4D de comprimento mínimo e que conseguisse evitar a colisão com obstáculos estáticos, respeitando distâncias máximas de passagem em relação aos *waypoints* intermédios, bem como as velocidades mínimas a máximas da aeronave.

Ficou claro, logo numa primeira fase do estudo, que o método de *Bézier* não seria o ideal para os objetivos traçados, devido às dificuldades relatadas aquando o seu estudo teórico. Foi também posto de parte o método das *B-splines*, que necessitava de calcular imensas funções de mistura, tornando-o pesado e obsoleto quando comparado com os métodos escolhidos para o restante estudo: *Overhauser* e *Splines Cúbicas*, métodos que recorrem a geração de trajetórias por *blending* de polinómios de segundo grau e geração de trajetórias por intermédio de polinómios de terceiro grau, respetivamente.

Ambos os métodos que restaram revelaram um bom comportamento nas respetivas secções onde apresentaram significativas reduções de comprimento da trajetória, tendo em conta distâncias permissíveis de 5000 e 10000 metros, chegando aos 4,20% no caso de *Overhauser*, e aos 3,50% no caso das *Splines Cúbicas*. O método da *Spline Cúbica* revelava-se (ao contrário do esperado) menos eficaz na geração de trajetórias de menor comprimento. Infelizmente, com o acrescentar de condições de velocidades o *MATLAB*® terminava as minimizações do método da *Spline Cúbica* com um *EXITFLAG=0* e *-2*. Este resultado negativo pode ser fruto da não existência de *"function blending"*, como presente no método de *Overhauser*.

Com o estudo a continuar apenas com o método de *Overhauser*, realizou-se a simulação final, onde dois obstáculos foram colocados em dois segmentos da trajetória. O método com sucesso ultrapassou o primeiro teste, evitando a colisão com ambos os obstáculos, e respeitando as condições de velocidade, e ainda assim apresentando uma redução no comprimento total da trajetória de 7,83%, face ao comprimento total obtido para a trajetória interpoladora, e uma redução na ordem dos 43% comparativamente com o tempo inicial.

As limitações deste método tornaram-se evidentes no segundo teste da simulação final, onde apenas se aumentou o raio do primeiro obstáculo, e o método já não foi eficaz a evitar a colisão com ambos os obstáculos. Assim sendo o estudo demonstrou que o método de *Overhauser* consegue sim evitar colisão com um obstáculo, e até mesmo dois, no entanto dependendo do tamanho dos mesmos (não deverão ser muito grandes).

5.2. Trabalhos futuros

Esta dissertação constitui uma boa base para que se possa acrescentar instrumentos numa aeronave que permitam detetar obstáculos em pleno voo, levando a alterações *in loco* da trajetória a fim de evitar as colisões. Um detetor de obstáculos com base em raios infravermelhos, por exemplo, permitia descobrir a tempo um conflito, e, uma vez que o algoritmo aqui apresentado corre numa questão de segundos, corrigir de forma eficaz a trajetória a fim de prevenir a colisão. Naturalmente que fica a ideia de que essa correção seria mais eficaz caso o método utilizado fosse o da *Spline Cúbica*, pois o método de *Overhauser* tem claras limitações, apesar de ser mais rápido a executar.

Uma sugestão apresentada e que pode ser testada num trabalho futuro é o acrescentar de um "*function blending*" também no método da *Spline Cúbica*, para que nos segmentos intermédios haja uma maior facilidade em garantir a continuidade e suavidade entre segmentos, como se faz no método de *Overhauser*.

Referências

[1] RCTA, "Report of the RTCA Board of Directors' Select Committee on Free Flight," RCTA, Incorporated, Washington D.C., 1995.

[2] K. Bousson, "Model Predictive Control Approach to Global Air Collision Avoidance," *Aircraft Engineering and Aerospace Technology, Vol. 80 Iss: 6,* pp. 605-612, 2008.

[3] R. W. Beard and W. T. McLain, "Multiple UAV Cooperative Search under Collision Avoidance and Limited Range Communication Constraints," *Decision and Control, 2003. Proceedings. 42nd IEEE Conference (Volume 1),* pp. 25-30, 2003.

[4] D. Alejo, J. A. Cobano, G. Heredia and A. Ollero, "Optimal Reciprocal Collision Avoidance with mobile and static obstacles for multi-UAV systems," *Unmanned Aircraft Systems (ICUAS), 2014 International Conference,* pp. 1259-1266, 2014.

[5] A. Richards and J. P. How, "Aircraft Trajectory Planning With Collision Avoidance Using Mixed Integer Linear Programming," *American Control Conference, 2002. Proceedings of the 2002 (vol. 3),* pp. 1936-1941, 2002.

[6] C. Carbone, U. Ciniglio, F. Corraro and S. Luongo, "A Novel 3D Geometric Algorithm for Aircraft Autonomous Collision Avoidance," *Decision and Control, 2006 45th IEEE Conference,* pp. 1580-1585, 2006.

[7] Z. He, R. V. Iyer and P. Chandler, "Vision-Based UAV Flight Control And Obstacle Avoidance," *American Control Conference,* 2006.

[8] R. V. Iyer, Z. He and P. Chandler, "On the Computacion Of The Ego-Motion And Distance To Obstacles For a Micro Air Vehicle," *American Control Conference, 2006,* 2006.

[9] A. Mcfadyen, L. Mejias, P. Corke and C. Pradalier, "Aircraft collision avoidance using spherical visual predictive control and single point features," *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference,* pp. 50-56, 2013.

[10] K. Bousson and P. Machado, "4D Trajectory Generation and Tracking for Waypoint-Based Aerial," *WSEAS Transactions on Systems & Control, July 2013 (Vol: 8, Iss: 3),* pp. 105-119, 2013.

[11] Google, "maps.google.com," 14 Janeiro 2015. [Online]. Available: https://www.google.pt/maps/@41.0160395,-6.1100517,8z?hl=en.

[12] H. Vermeille, "Direct transformation from geocentric coordinates to geodetic coordinates," *Journal of Geodesy, vol. 76,* pp. 451-454, 2002.

[13] M. Kamermans, "A Primer on Bézier Curves," Pomax, 2011. [Online]. Available: http://pomax.github.io/bezierinfo/. [Accessed 05 2015].

[14] K. Joy, "Introduction to Geometric Modeling," UC Davis, 2012. [Online]. Available: http://graphics.cs.ucdavis.edu/~joy/ecs178/index.html. [Accessed 05 2015].

[15] N. M. Patrikalakis, T. Maekawa and W. Cho, "1.3.5 Algorithms for Bézier Curves," MIT, 12 2009. [Online]. Available: http://web.mit.edu/hyperbook/Patrikalakis-Maekawa-Cho/node13.html. [Accessed 05 2015].

[16] A. W. Overhauser, "Analytic definition of curves and surfaces," Mathematical and Theoretical Sciences Department, Scientific Laboratory, Ford Motor Company, Dearborn, Michigan, 1968.

[17] M. A. Khamsi and H. Knaust, "The Mean-Value Theorem," S.O.S. Math, 2015. [Online]. Available: http://www.sosmath.com/calculus/diff/der11/der11.html. [Accessed 20 06 2015].

[18] USP, "Comprimento de Arco de uma Curva," E-Cálculo, 2012. [Online]. Available: http://ecalculo.if.usp.br/integrais/aplicacoes_integral/compr_arc_curva/compr_arc_curva.htm. [Accessed 22 05 2015].

[19] The MathWorks, Inc, "Documentation - fmincon," MathWorks, 2015. [Online]. Available: http://www.mathworks.com/help/optim/ug/fmincon.html. [Accessed 21 04 2015].

[20] H. Hadavinia, R. P. Travis and R. T. Fenner, "Continuous Generalised Parabolic Blending Elements in the Boundary Element Method," *Mathematical and Computer Modelling vol. 31,* pp. 17-34, 2000.

Anexos

Anexo A – Tabelas de coeficientes e gráficos da Simulação 1: Geração de Trajetórias de Comprimento Mínimo

Neste anexo encontram-se as tabelas com os coeficientes polinomiais obtidos para as trajetórias por interpolação e por aproximação para os dois métodos (*Overhauser* e *Spline Cúbica*), bem como os gráficos com as três diferentes vistas, para a **Simulação 1**.

- Método de *Overhauser*:

Tabela A.1 - Valores dos coeficientes de x para cada segmento, obtidos por interpolação – Método Overhauser.

Interpolação em *x*	**Coeficiente a_1**	**Coeficiente a_2**	**Coeficiente a_3**
Segmento 1	-0,000519163	11,796903176	4757791,148243761
Segmento 2	-0,000047259	1,352651898	4804291,352960094
Segmento 3	0,000329746	-16,235691478	4988953,137683667

Tabela A.2 - Valores dos coeficientes para cada segmento, obtidos por aproximação – Método Overhauser.

Aproximação em *x*	**Coeficiente a_1**	**Coeficiente a_2**	**Coeficiente a_3**
Segmento 1	-0,000518792	11,996841309	4757791,148243801
Segmento 2	-0,000014918	0,169473025	4810363,975178574
Segmento 3	0,000302820	-14,715092456	4970749,985970632

Tabela A.3 - Valores dos coeficientes de y para cada segmento, obtidos por interpolação – Método Overhauser.

Interpolação em *y*	**Coeficiente b_1**	**Coeficiente b_2**	**Coeficiente b_3**
Segmento 1	0,0004973188	1,7559194695	-726179,5452260716
Segmento 2	0,0000080448	12,5846139385	-774391,3793013020
Segmento 3	-0,0001643115	20,6255202337	-858813,6626009392

Tabela A.4 - Valores dos coeficientes de y para cada segmento, obtidos por aproximação – Método Overhauser.

Aproximação em *y*	**Coeficiente b_1**	**Coeficiente b_2**	**Coeficiente b_3**
Segmento 1	0,0004940547	1,7321610546	-726179,5452258055
Segmento 2	0,0000010630	12,4397513455	-768295,9440372257
Segmento 3	-0,0001497291	19,6141162278	-841580,8502816526

Tabela A.5 - Valores dos coeficientes de z para cada segmento, obtidos por interpolação – Método Overhauser.

Interpolação em z	Coeficiente c_1	Coeficiente c_2	Coeficiente c_3
Segmento 1	0,000673260	-12,689980609	4218645,500933868
Segmento 2	-0,000004521	2,310765830	4151858,733572812
Segmento 3	-0,000411598	21,302073743	3952467,083178452

Tabela A.6 - Valores dos coeficientes de z para cada segmento, obtidos por aproximação – Método Overhauser.

Aproximação em z	Coeficiente c_1	Coeficiente c_2	Coeficiente c_3
Segmento 1	0,000657547	-12,466266603	4218645,500933868
Segmento 2	-0,000001563	2,136075103	4153630,500042723
Segmento 3	-0,000377222	19,356104841	3975888,903735255

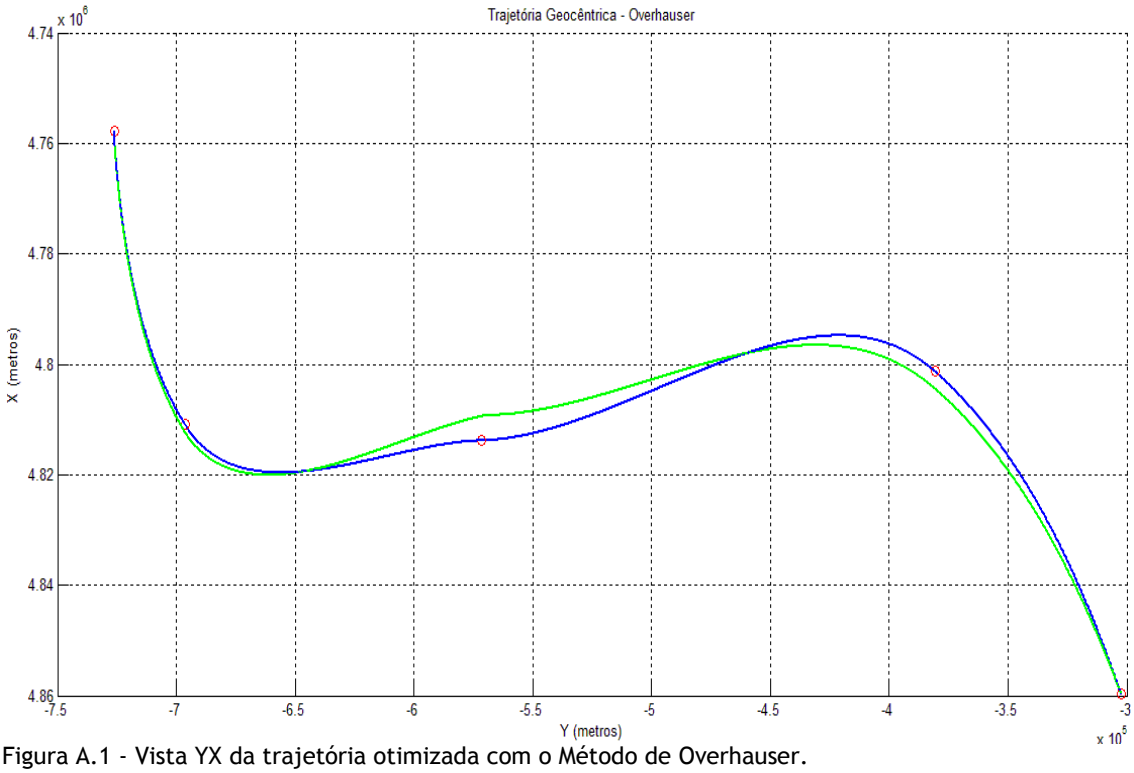

Figura A.1 - Vista YX da trajetória otimizada com o Método de Overhauser.

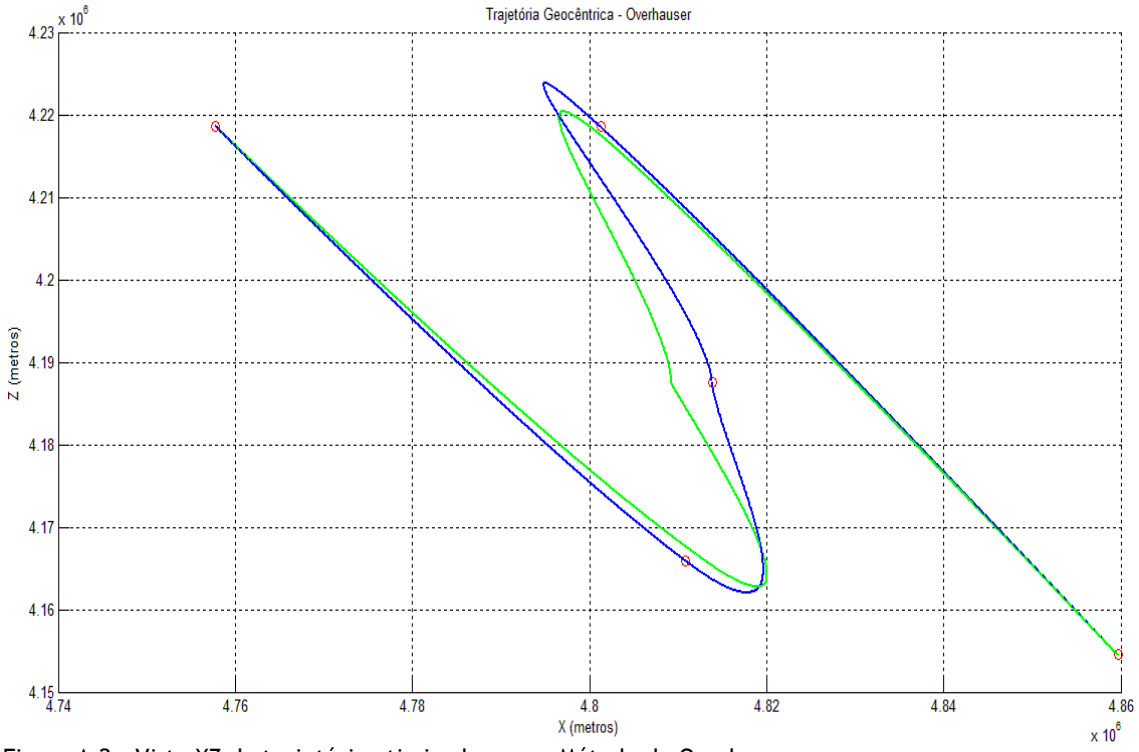

Figura A.2 - Vista XZ da trajetória otimizada com o Método de Overhauser.

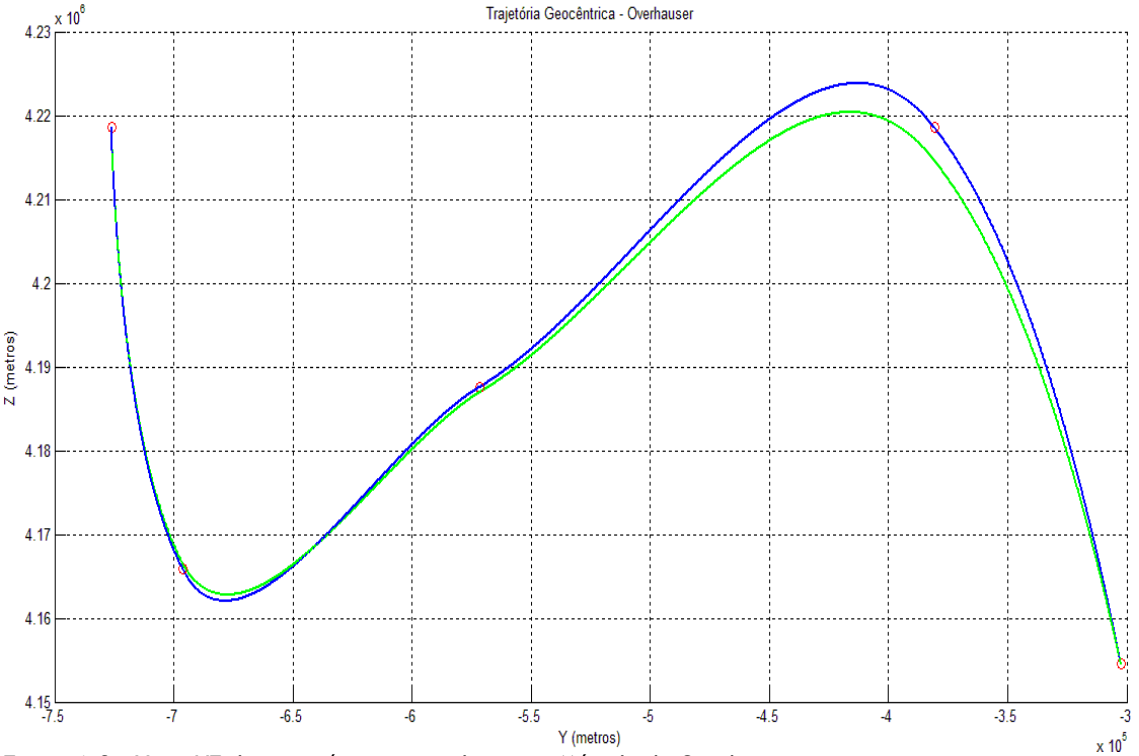

Figura A.3 - Vista YZ da trajetória otimizada com o Método de Overhauser.

- Método da *Spline Cúbica*:

Tabela A.7 - Valores dos coeficientes de x para cada segmento, obtidos por interpolação – Método Spline Cúbica.

Interpolação em x	Coeficiente a_0	Coeficiente a_1	Coeficiente a_2	Coeficiente a_3
Segmento 1	4757791,14824376	8,590967448	0,000200911	-0,000000033
Segmento 2	4810842,11274246	7,350299025	-0,000988473	0,000000027
Segmento 3	4813842,21981825	-4,111475375	0,000152412	0,000000005

Tabela A.8 - Valores dos coeficientes de x para cada segmento, obtidos por aproximação – Método Spline Cúbica.

Aproximação em *x*	Coeficiente a_0	Coeficiente a_1	Coeficiente a_2	Coeficiente a_3
Segmento 1	4757791,14824376	11,048123663	-0,000456387	0,000000001
Segmento 2	4808883,13679415	5,5432414556	-0,000526100	0,000000004
Segmento 3	4816413,06194421	-3,630920654	0,000210998	0,000000001

Tabela A.9 - Valores dos coeficientes de y para cada segmento, obtidos por interpolação – Método Spline Cúbica.

Interpolação em y	Coeficiente b_0	Coeficiente b_1	Coeficiente b_2	Coeficiente b_3
Segmento 1	-726179,545226072	4,8269638448	-0,0001924579	0,000000031
Segmento 2	-696372,065554237	6,0154305223	0,0009617329	-0,000000028
Segmento 3	-571530,731918569	16,8496844433	-0,0003269396	0,000000004

Tabela A.10 - Valores dos coeficientes de y para cada segmento, obtidos por aproximação – Método Spline Cúbica.

Aproximação em y	Coeficiente b_0	Coeficiente b_1	Coeficiente b_2	Coeficiente b_3
Segmento 1	-726179,545226072	3,2691875011	0,0002390502	0,000000009
Segmento 2	-694782,283827458	7,2386711971	0,0003640815	-0,000000007
Segmento 3	-596080,916689005	12,232336485	0,0000392929	-0.000000010

Tabela A.11 - Valores dos coeficientes de z para cada segmento, obtidos por interpolação – Método Spline Cúbica.

Interpolação em z	Coeficiente c_0	Coeficiente c_1	Coeficiente c_2	Coeficiente c_3
Segmento 1	4218645,500933868	-8,532465243	-0,000260545	0,000000042
Segmento 2	4165955,799402993	-6,923544019	0,001308122	-0,000000038
Segmento 3	4187580,508499864	7,683504055	-0,000358105	-0,000000001

Tabela A.12 - Valores dos coeficientes de z para cada segmento, obtidos por aproximação – Método Spline Cúbica.

Aproximação em z	Coeficiente c_0	Coeficiente c_1	Coeficiente c_2	Coeficiente c_3
Segmento 1	4218645,50093387	-11,089459559	0,000101806	0,000000026
Segmento 2	4160172,79786641	-6,856611636	0,000941660	-0,000000015
Segmento 3	4169019,75857480	7,215430773	-0,000225655	-0,000000005

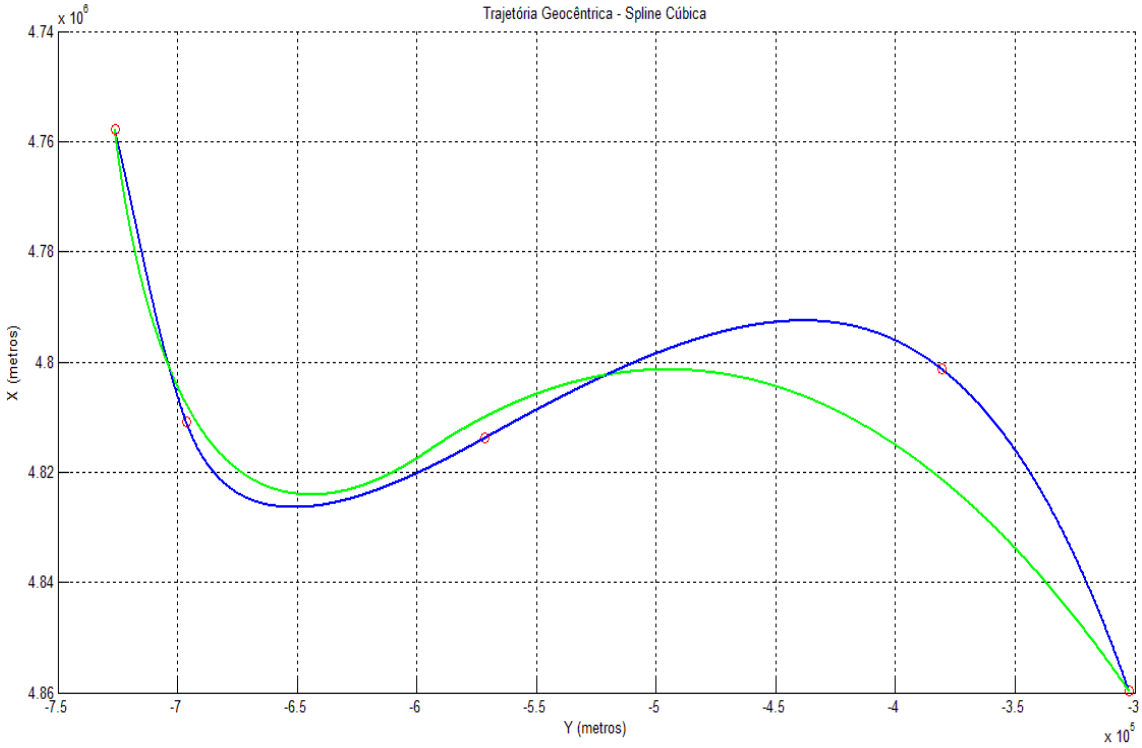

Figura A.4 - Vista YX da trajetória otimizada com o Método de Spline Cúbica.

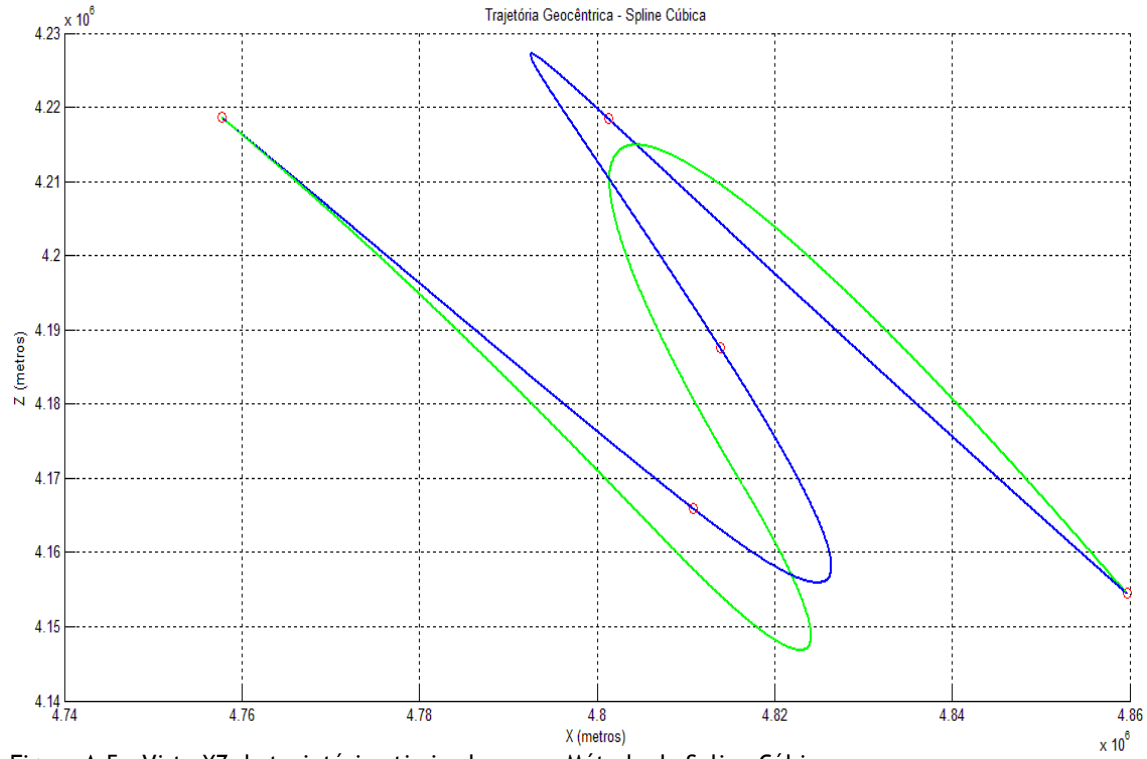

Figura A.5 - Vista XZ da trajetória otimizada com o Método de Spline Cúbica.

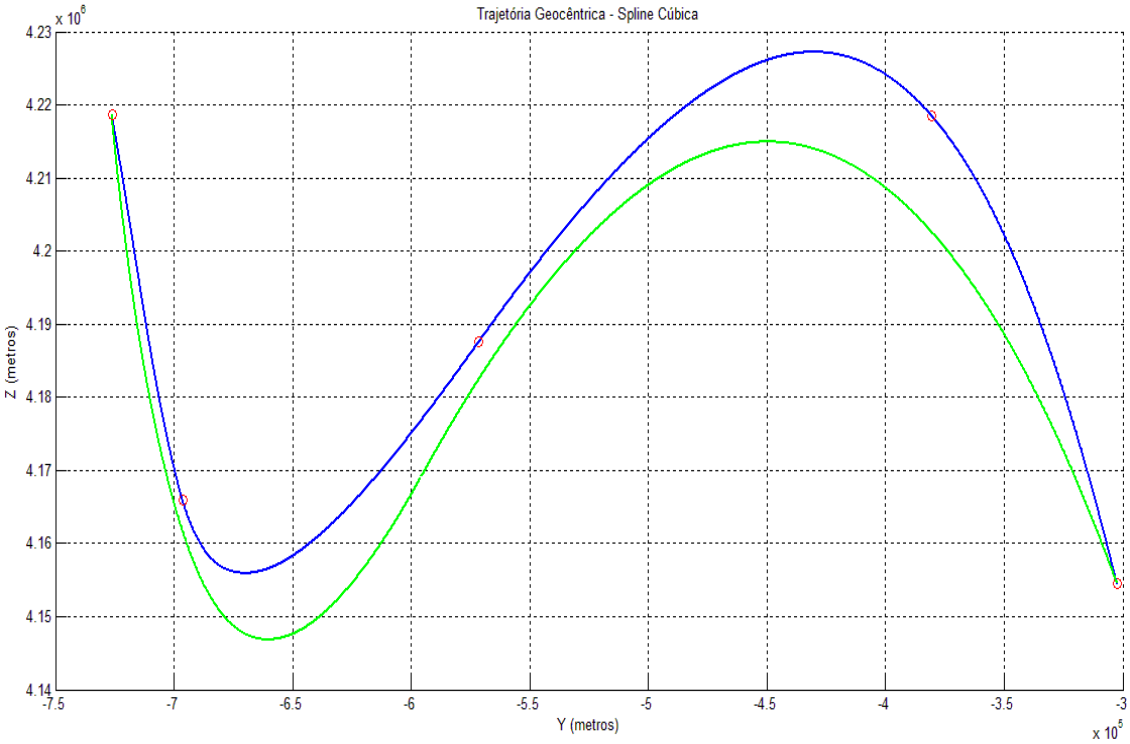

Figura A6 - Vista YZ da trajetória otimizada com o Método de Spline Cúbica.

Anexo B – Tabelas de coeficientes e gráficos da Simulação 2: Geração de Trajetórias de Comprimento Mínimo com cumprimento da velocidade mínima e máxima

Neste anexo encontram-se as tabelas com os coeficientes polinomiais obtidos para as trajetórias resultantes da minimização do comprimento com respeito à velocidade para os dois métodos (*Overhauser* e *Spline Cúbica*), bem como os gráficos com as três diferentes vistas, para a **Simulação 2**.

- Método de *Overhauser*:

Tabela B.1 - Valores dos coeficientes de x para cada segmento, para a minimização do comprimento com respeito à velocidade – Método Overhauser.

Coeficientes em *x*	**Coeficiente a_1**	**Coeficiente a_2**	**Coeficiente a_3**
Segmento 1	-0,0014069790	19,42492857	4757791,148
Segmento 2	-0,0000698993	0,437438805	4810117,115
Segmento 3	0,0006362440	-15,66023524	4898055,970

Tabela B.2 - Valores dos coeficientes de y para cada segmento, obtidos para a minimização do comprimento com respeito à velocidade – Método Overhauser.

Coeficientes em *y*	**Coeficiente b_1**	**Coeficiente b_2**	**Coeficiente b_3**
Segmento 1	0,001061783	5,587716288	-726179,5452
Segmento 2	0,000189735	23,93640406	-777645,7625
Segmento 3	-0,001403188	61,24384712	-970674,3915

Tabela B.3 - Valores dos coeficientes de z para cada segmento, obtidos para a minimização do comprimento com respeito à velocidade – Método Overhauser.

Coeficientes em z	**Coeficiente c_1**	**Coeficiente c_2**	**Coeficiente c_3**
Segmento 1	0,001748684	-20,17613393	4218645,501
Segmento 2	0,0000557248	3,53972529	4153760,019
Segmento 3	-0,000880096	23,85689121	4053455,892

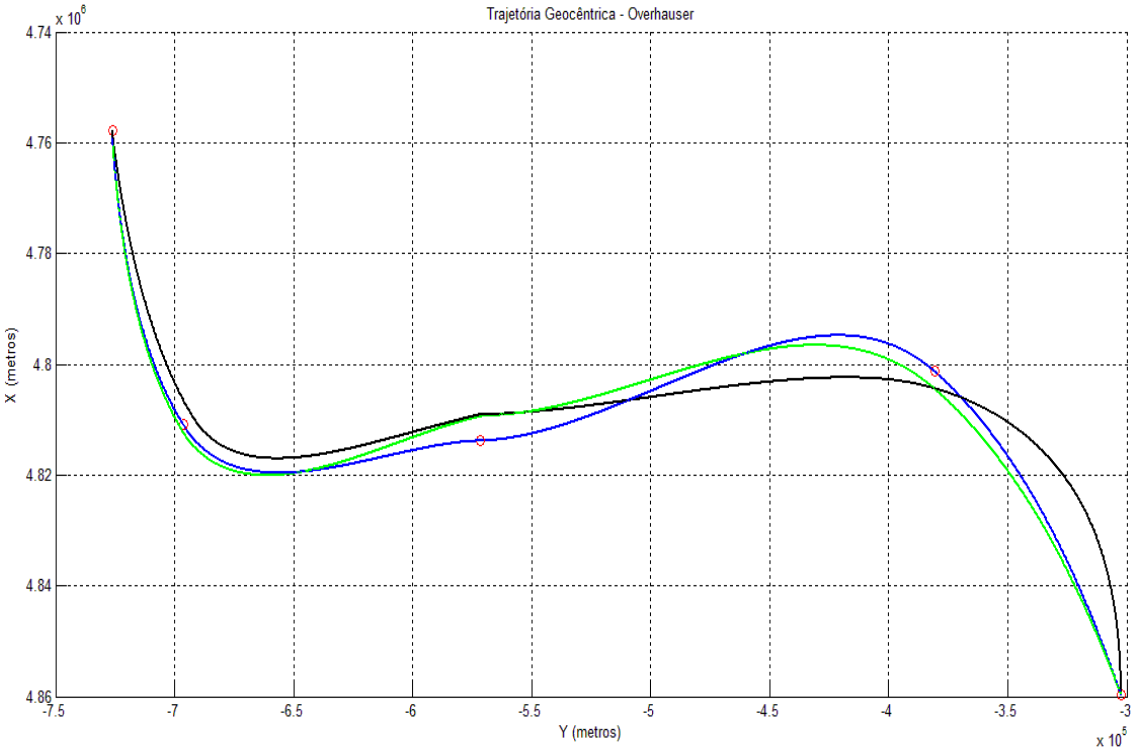

Figura B.1 - Vista YX da trajetória, que respeita a velocidade mínima e máxima, com o Método de Overhauser.

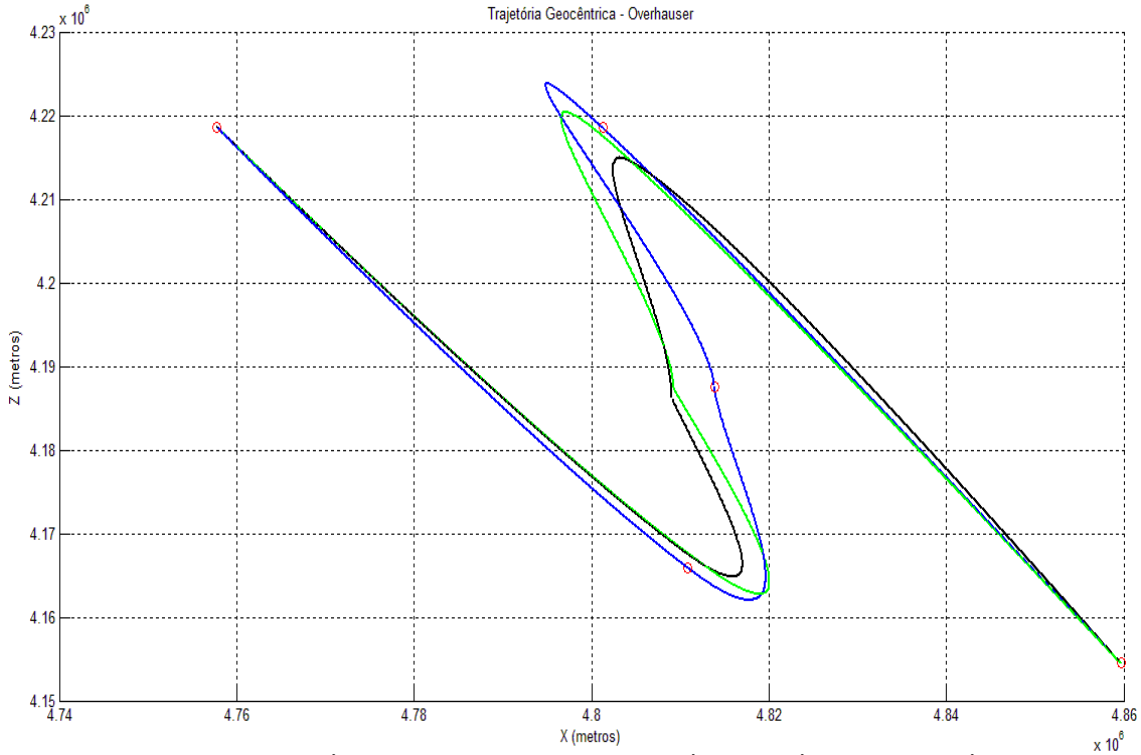

Figura B.2 - Vista XZ da trajetória, que respeita a velocidade mínima e máxima, com o Método de Overhauser.

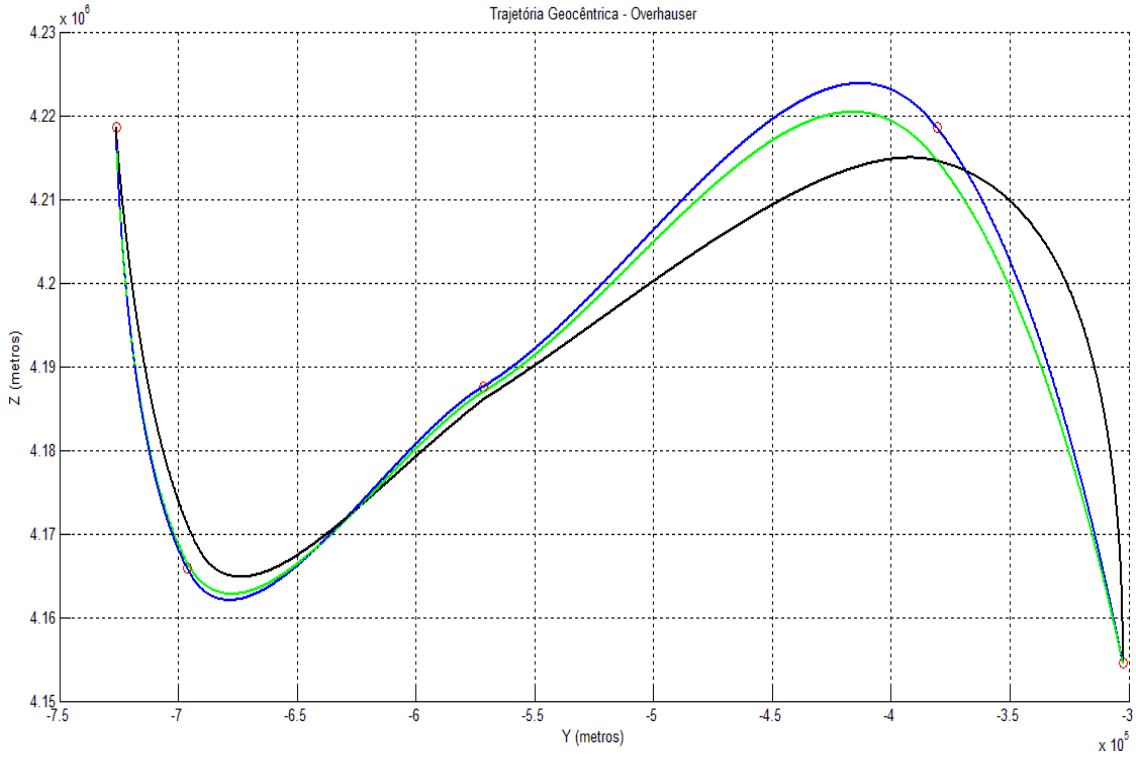

Figura B.3 - Vista YZ da trajetória, que respeita a velocidade mínima e máxima, com o Método de Overhauser.

Anexo C – Tabelas de coeficientes e gráficos da Simulação Final: com prevenção de colisão com 2 obstáculos modelados por esferas

Neste anexo encontram-se as tabelas com os coeficientes polinomiais obtidos para as trajetórias resultantes da minimização do comprimento com respeito à velocidade, e a evitar colisão com 2 obstáculos para os dois métodos (*Overhauser* e *Spline Cúbica*), bem como os gráficos com as três diferentes vistas, para ambos os ensaios da **Simulação Final** em cada método.

- Método de *Overhauser* – 1ºteste:

Tabela C.1 - Valores dos coeficientes de x para cada segmento, para a prevenção de colisão com 2 obstáculos – Método Overhauser.

Coeficientes em *x*	**Coeficiente a_1**	**Coeficiente a_2**	**Coeficiente a_3**
Segmento 1	-0,001406975	19,42488995	4757791,148
Segmento 2	-0,000061795	0,302708386	4810488,268
Segmento 3	0,000634157	-15,8652564	4899042,937

Tabela C.2 - Valores dos coeficientes de y para cada segmento, para a prevenção de colisão com 2 obstáculos – Método Overhauser.

Coeficientes em *y*	**Coeficiente b_1**	**Coeficiente b_2**	**Coeficiente b_3**
Segmento 1	0,001061781	5,587716834	-726179,5452
Segmento 2	0,000216374	23,20821086	-775410,8268
Segmento 3	-0,001265696	57,26614337	-949958,315

Tabela C.3 - Valores dos coeficientes de z para cada segmento, para a prevenção de colisão com 2 obstáculos – Método Overhauser.

Coeficientes em z	**Coeficiente c_1**	**Coeficiente c_2**	**Coeficiente c_3**
Segmento 1	0,001748688	-20,17616909	4218645,501
Segmento 2	0,000010794	4,322919711	4151507,705
Segmento 3	-0,000860449	23,81951668	4050130,374

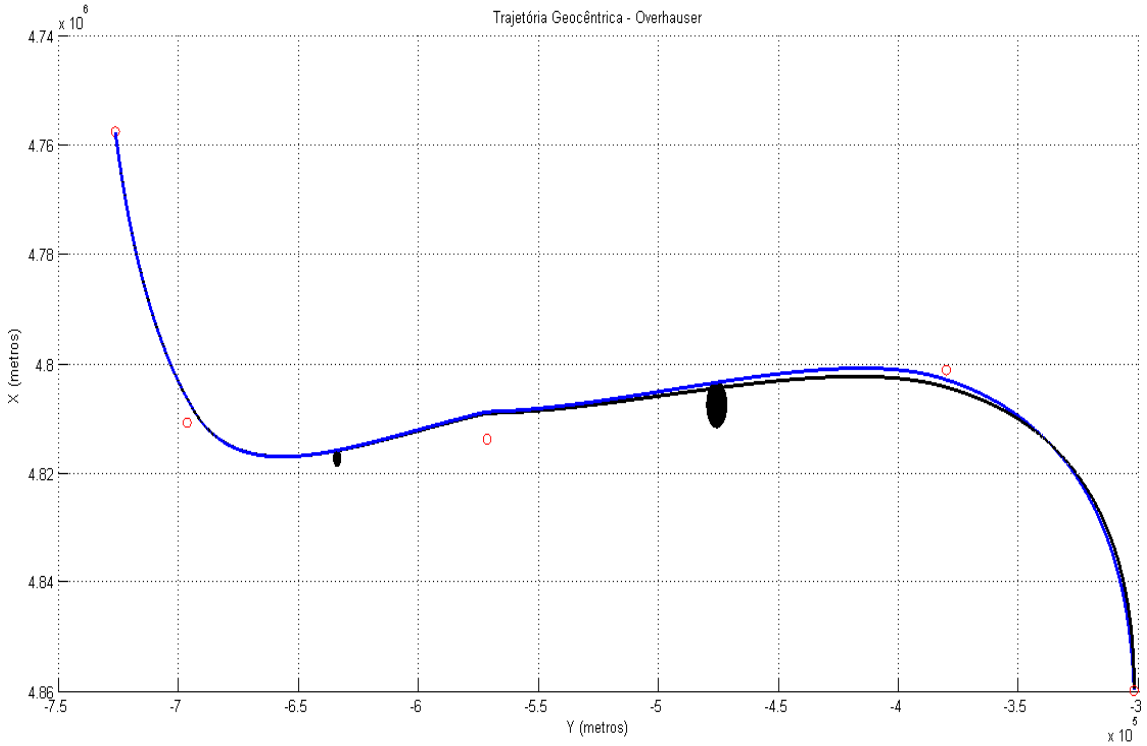

Figura C.1 - Vista YX da trajetória, que evita a colisão com 2 obstáculos, com o Método de Overhauser.

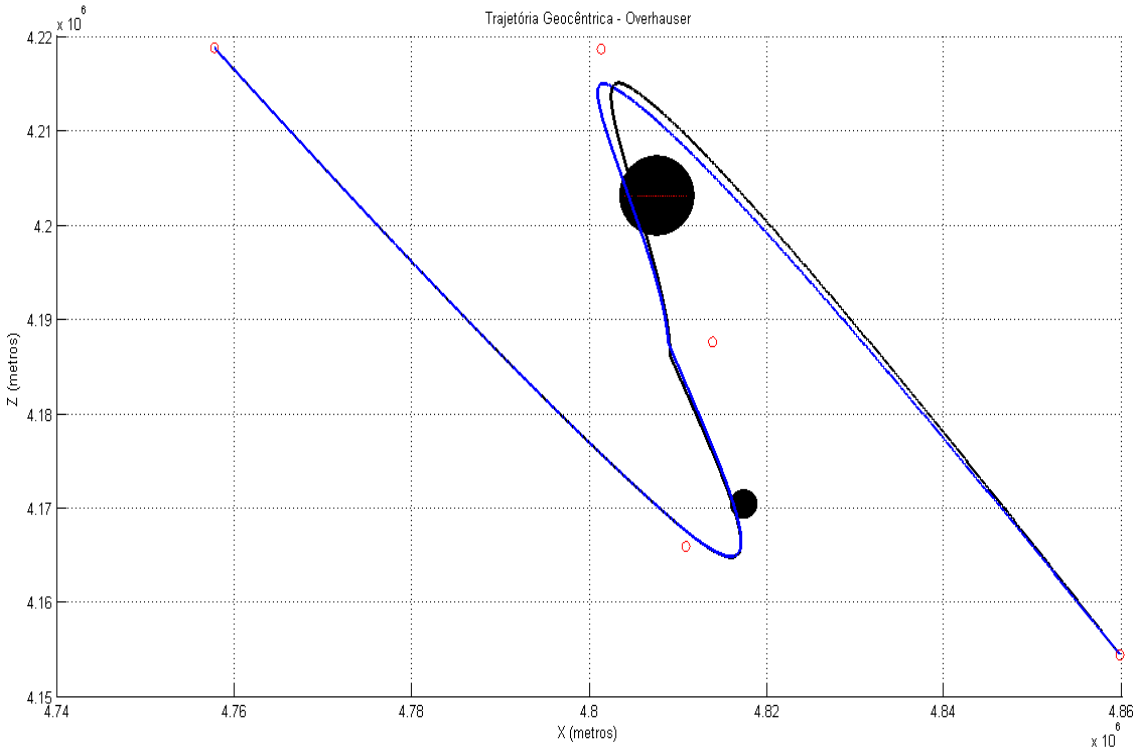

Figura C.2 - Vista XZ da trajetória, que evita a colisão com 2 obstáculos, com o Método de Overhauser.

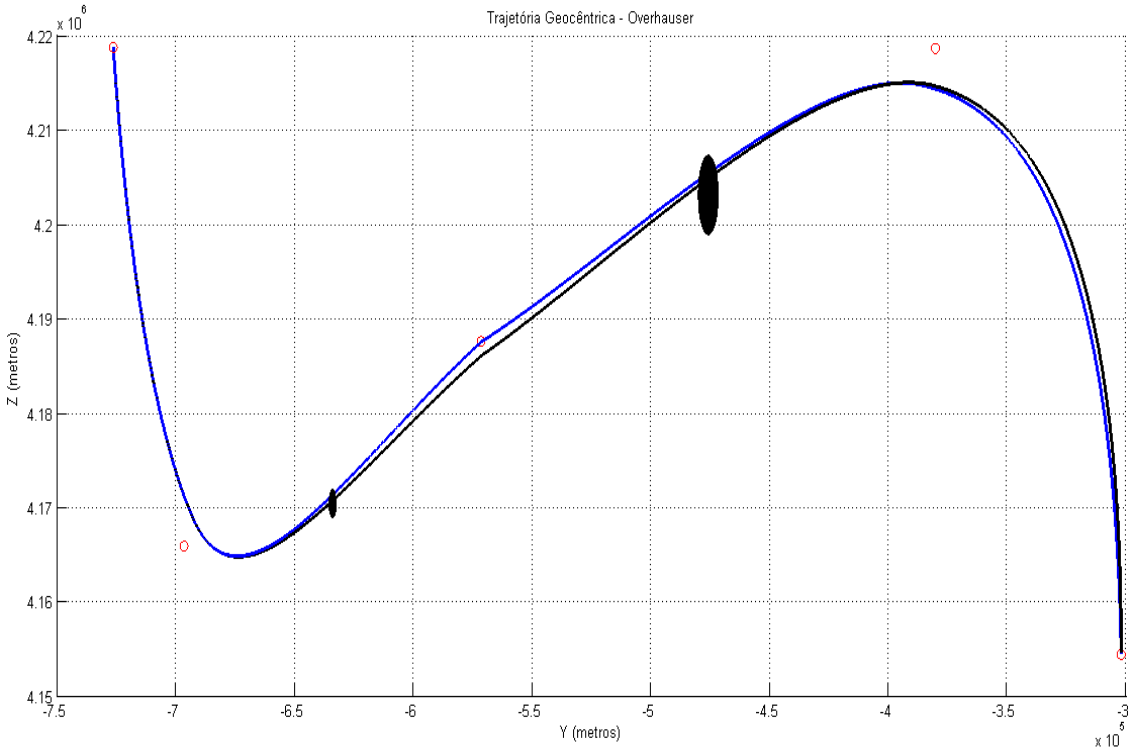

Figura C.3 - Vista YZ da trajetória, que evita a colisão com 2 obstáculos, com o Método de Overhauser.

- Método de *Overhauser* – 2ºteste:

Tabela C.4 - Valores dos coeficientes de x para cada segmento, para a prevenção de colisão apenas com o 1º obstáculo, falhando no 2º – Método Overhauser.

Coeficientes em *x*	**Coeficiente a_1**	**Coeficiente a_2**	**Coeficiente a_3**
Segmento 1	-0,000841794	14,77931215	4757791,148
Segmento 2	-0,000061357	0,314313554	4810431,987
Segmento 3	0,000655134	-17,23696115	4920107,422

Tabela C.5 - Valores dos coeficientes de y para cada segmento, para a prevenção de colisão apenas com o 1º obstáculo, falhando no 2º – Método Overhauser.

Coeficientes em *y*	**Coeficiente b_1**	**Coeficiente b_2**	**Coeficiente b_3**
Segmento 1	0,001551474	5,668618032	-726179,5452
Segmento 2	0,000228048	22,81078411	-775542,3245
Segmento 3	-0,000988733	48,96638599	-902676,3157

Tabela C.6 - Valores dos coeficientes de z para cada segmento, para a prevenção de colisão apenas com o 1º obstáculo, falhando no 2º – Método Overhauser.

Coeficientes em z	**Coeficiente c_1**	**Coeficiente c_2**	**Coeficiente c_3**
Segmento 1	0,00175171	-18,36744856	4218645,501
Segmento 2	0,00001277	4,260184864	4151466,744
Segmento 3	-0,000878263	24,70737632	4038624,33

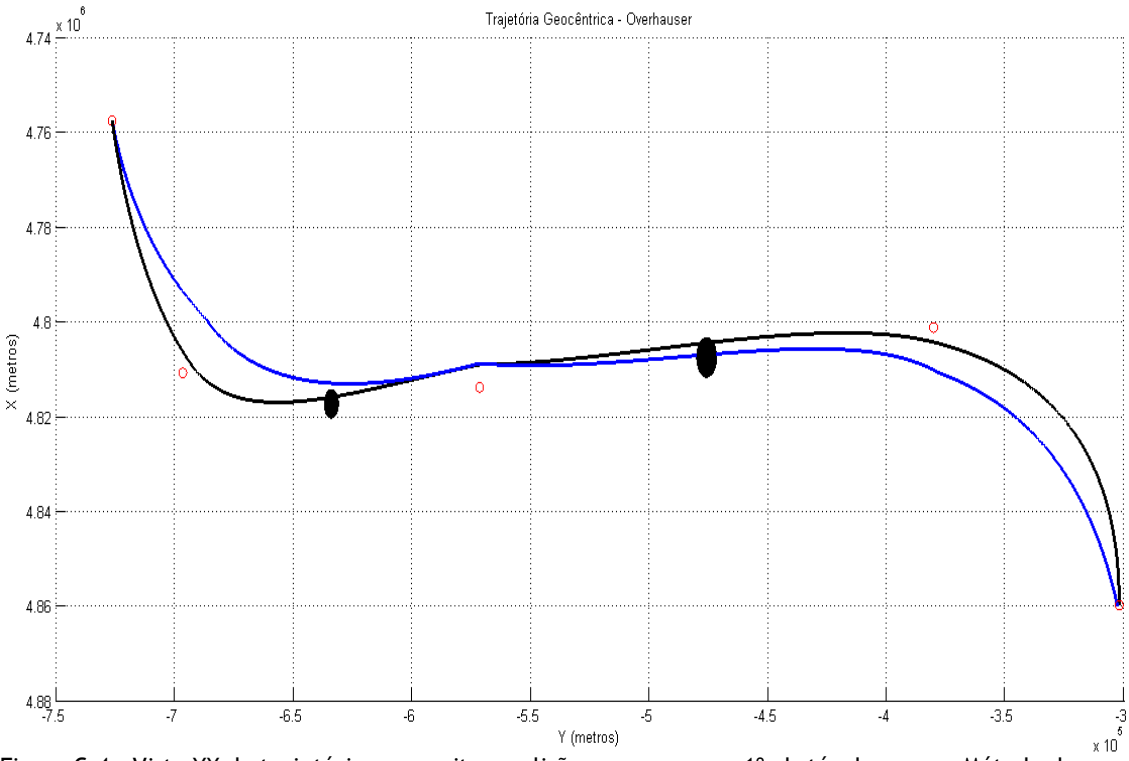

Figura C.4 - Vista YX da trajetória, que evita a colisão apenas com o 1º obstáculo, com o Método de Overhauser.

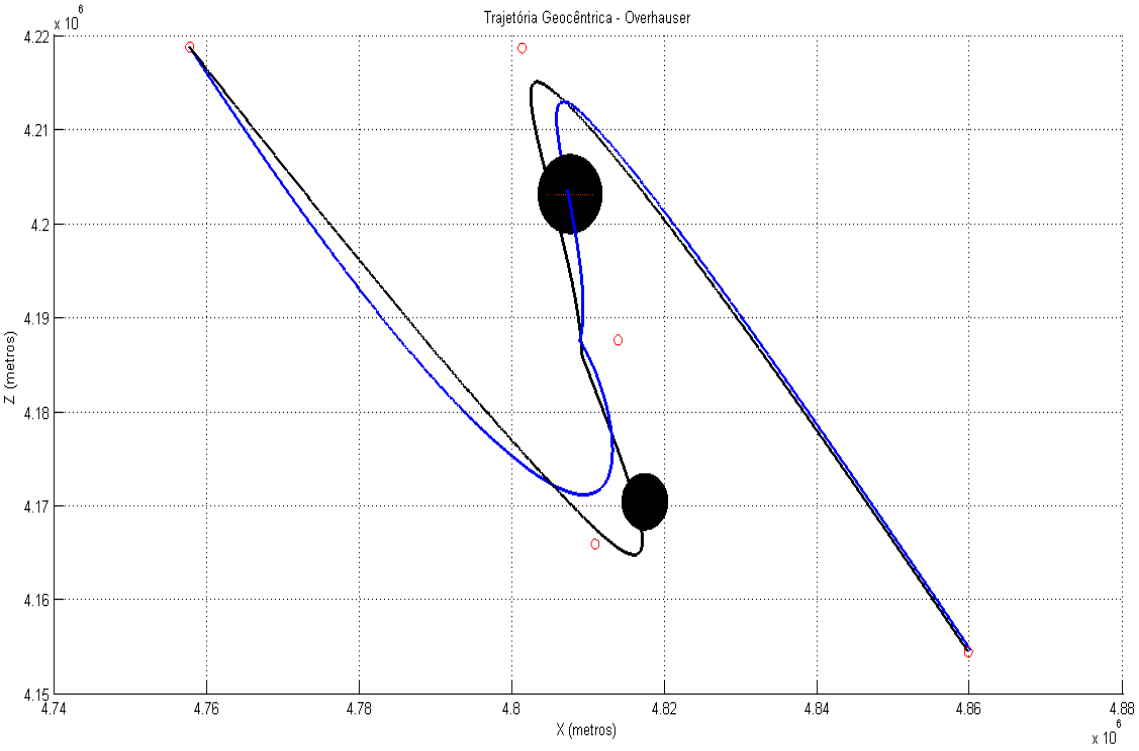

Figura C.5 - Vista XZ da trajetória, que evita a colisão apenas com o 1º obstáculo, com o Método de Overhauser.

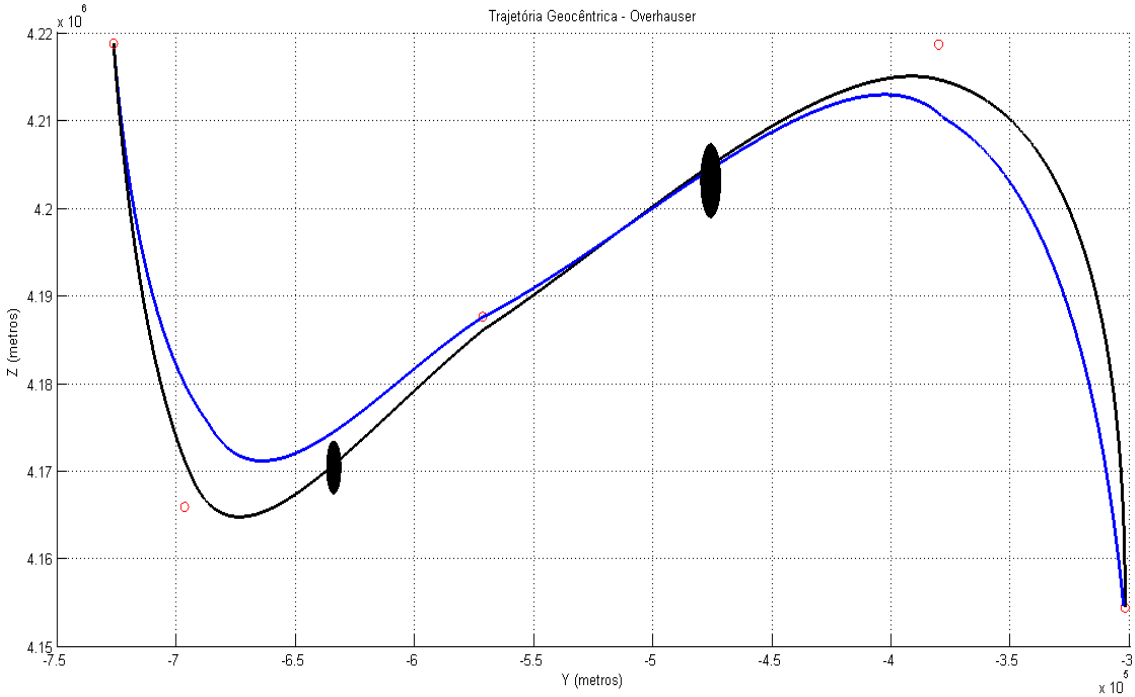

Figura C.6 - Vista YZ da trajetória, que evita a colisão apenas com o 1º obstáculo, com o Método de Overhauser.

- Método de *Overhauser* – 3ºteste:

Tabela C.7 - Valores dos coeficientes de x para cada segmento, para a prevenção de colisão apenas com um obstáculo no caminho – Método Overhauser.

Coeficientes em *x*	Coeficiente a_1	Coeficiente a_2	Coeficiente a_3
Segmento 1	-0,001406973	19,424877520	4757791,148
Segmento 2	-0,000064266	0,323671456	4810459,511
Segmento 3	0,000658120	-16,554383772	4904345,124

Tabela C.8 - Valores dos coeficientes de y para cada segmento, para a prevenção de colisão apenas com um obstáculo no caminho – Método Overhauser.

Coeficientes em *y*	Coeficiente b_1	Coeficiente b_2	Coeficiente b_3
Segmento 1	0,001061783	5,587714972	-726179,5452
Segmento 2	0,000186338	23,84573954	-777313,4874
Segmento 3	-0,001268863	57,23702738	-947599,2935

Tabela C.9 - Valores dos coeficientes de z para cada segmento, para a prevenção de colisão apenas com um obstáculo no caminho – Método Overhauser.

Coeficientes em z	Coeficiente c_1	Coeficiente c_2	Coeficiente c_3
Segmento 1	0,00174869	-20,17616596	4218645,501
Segmento 2	-0,000002772	4,58869386	4150944,521
Segmento 3	-0,000841399	22,90533817	4058959,647

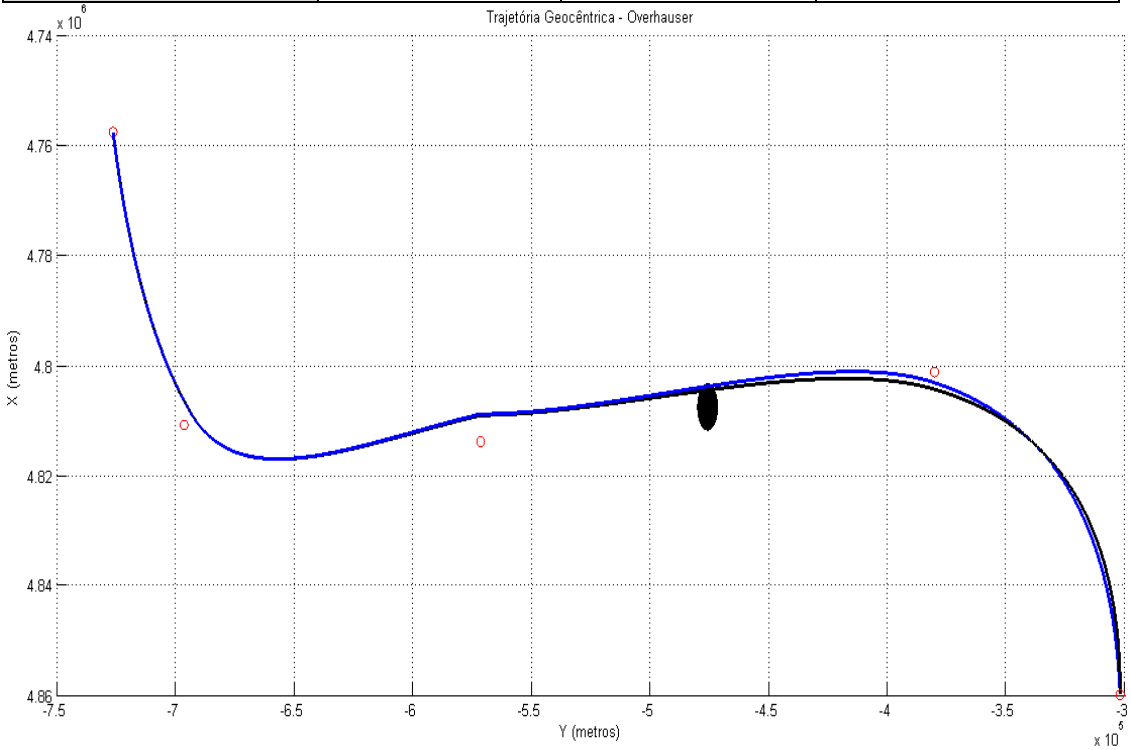

Figura C.7 - Vista YX da trajetória, apenas com o um obstáculo no caminho, com o Método de Overhauser.

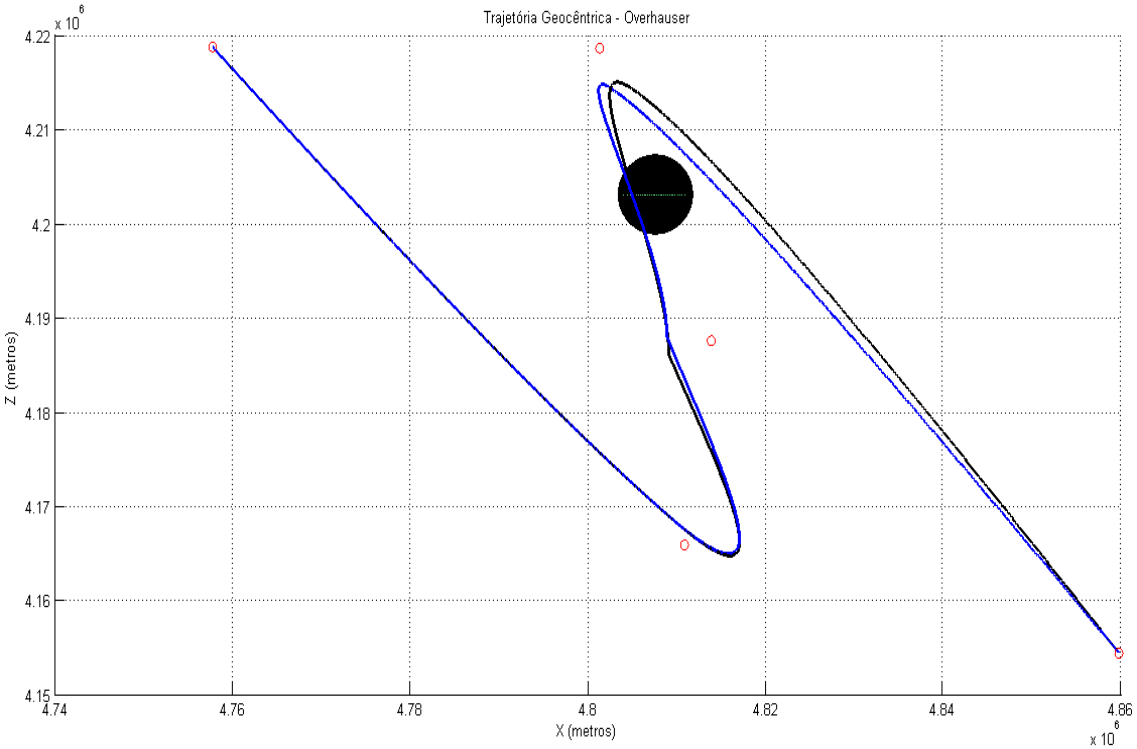

Figura C.8 - Vista XZ da trajetória, apenas com um obstáculo, com o Método de Overhauser.

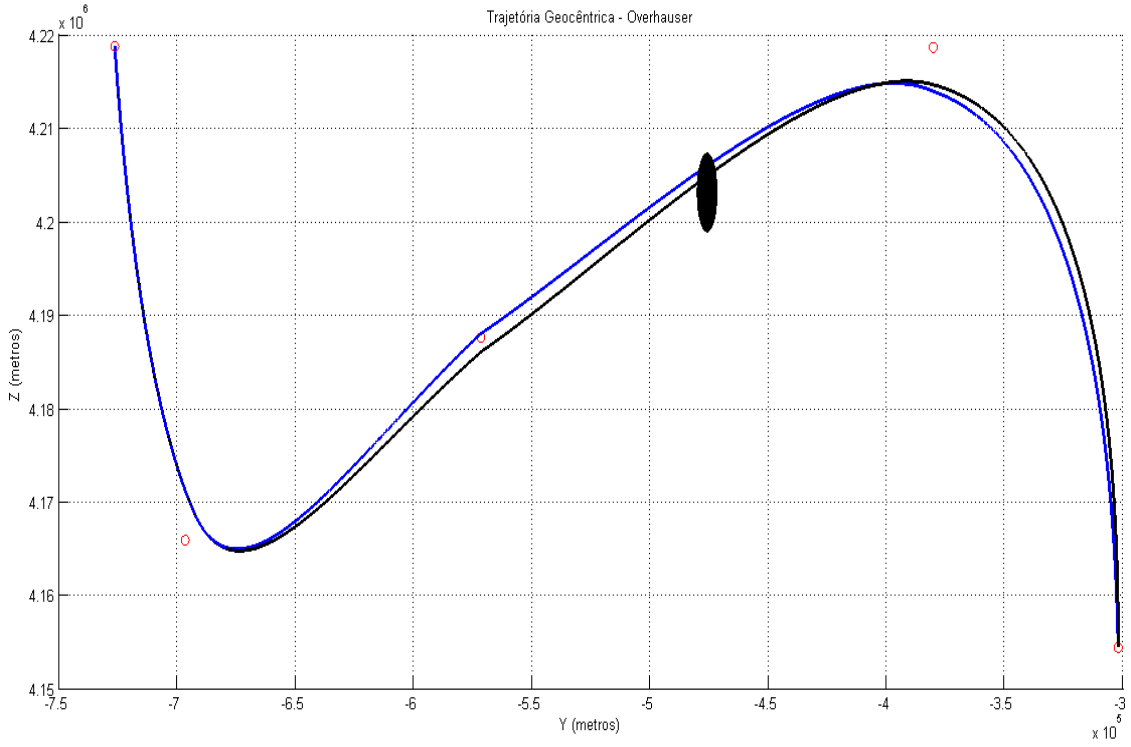

Figura C.9 - Vista YZ da trajetória, apenas com um obstáculo no caminho, com o Método de Overhauser.