SPOLM 2008

ISSN 2175-6295 Rio de Janeiro- Brasil, 05 e 06 de agosto de 2008.

DETERMINAÇÃO DE TRAJETÓRIA SUB-ÓTIMA PARA MÍSSIL AR-TERRA

Filipe Rodrigues de Souza Moreira
Instituto de Aeronáutica e Espaço, CTA
Praça Marechal Eduardo Gomes, 50 – Vila das Acácias,
São José Dos Campos, SP, 12228-904
filipersmoreira@yahoo.com.br

Cap. Eng. Prof. Dr. Christian Giorgio Roberto Taranti
Instituto de Aeronáutica e Espaço, CTA
Praça Marechal Eduardo Gomes, 50 – Vila das Acácias,
São José Dos Campos, SP, 12228-904
taranti@freeshell.org

RESUMO

Para que uma bomba, míssil ou outro artefato bélico seja lançado e chegue ao alvo prescrito com uma precisão adequada é necessária a introdução de uma espécie de software embarcado. Trata-se de um sistema, muitas vezes complexo e que será destruído quando o alvo for atingido, dessa forma, torna-se imprescindível descobrir se não há opções de tecnologia menos caras para que o cumprimento da missão aconteça dentro do melhor custo-benefício. Esse trabalho foca na determinação da trajetória de referência de um míssil que possui uma missão bem definida e com condições iniciais e finais conhecidas e fixas. O veículo é dotado de sensores de posição e velocidade que mostram essas informações em tempo real. O sistema em questão atuará da seguinte maneira: o sensor de posição mostra onde está o veículo em cada instante de tempo e essa posição real do veículo é comparada com a posição prevista pela trajetória de referência. Essa diferença acarretará em um erro. O sinal desse erro realimenta o sistema, de tal modo que alguma atitude de controle seja tomada, para que o navegador retorne a sua rota prescrita. Esse tipo de sistema se mostra mais viável quando a trajetória é dada como entrada, ou seja, é conhecida. Há outros sistemas mais sofisticados e complexos, porém sua utilização faz jus à situação de alvo móvel. Para esses casos, não se tem uma trajetória prescrita, logo o veículo deve, de alguma forma enxergar, ou sentir o alvo e em cada instante de tempo tomar uma atitude que configure uma perseguição. Esse realmente é um sistema bem mais complexo e com custo mais elevado de implantação, e para o caso, em que todas as condições iniciais e finais são conhecidas a implementação desse tipo de dispositivo muito mais inteligente se torna um exagero e um desperdício de tecnologia. Um sistema mais simples, como o comentado acima se mostra o suficiente para o cumprimento dessa missão.

A busca pela melhor trajetória para atingir o objetivo, dentro das condições previamente conhecidas, foi feita através da utilização de um algoritmo de busca informada chamado A* (Aestrela) e para sustentar a sua implementação promoveu-se uma discretização do espaço obtendo assim, uma malha finita de pontos. A melhor trajetória se trata de um subconjunto dessa malha gerada.

Palavras-chave: Trajetória de Veículo não Tripulado, Inteligência Artificial, Busca Informada, Simulação Computacional.

ABSTRACT

To launch a weapon, bomb or another destruction vehicle and to find, with a great accuracy, the goal point is necessary using on board software. This work is focused in the determination of the reference trajectory for missiles witch have a well defined mission, with known and fixed initial and final conditions. The vehicle carries a position and velocity sensors in a real time. This device will proceed like this: the position sensor shows when is the vehicle for each time increment and this information will be compared to the position prescript by the reference trajectory. This difference will produce an error. The signal of this error will back to the system producing a control action that will make the vehicle back to the right way. This device is a good one when the trajectory is given like entrance of the system. There are another complex and more sophisticated systems, but these using is on the case where the goal point is not static. For these one there is not a trajectory, so they have to see or to find the objective and each time increment take an attitude that configures a persecution situation. This really would be a more complex and expensive system and to this case when the initial and final conditions is known, using this sophisticated system would be an overstate and a wasting of technology because a less complex system, like this one proposed in this work, arrives in the objective, and with a good precision.

The search for the best way to reach the objective, always watching the proposed and known conditions, was done using an algorithm of informed search called by A* (A star) and to sustain its implementation was got a discrete form of the space that produced a finite grid of points. The best trajectory is a part of this big grid of points. It's really clear that there are other applications, different of the war industry, for the result of this work and a great example is automatic pilots and other systems of alone and non military navigation.

Keywords: Non Human Conduced Vehicles, Artificial Intelligence, Informed Search, Computational Simulation.

1. INTRODUÇÃO

O trabalho com navegadores autônomos abrange vários campos tecnológicos e em particular uma das partes que compõem esse assunto é a geração de trajetórias de referência. Trata-se de um item primordial, pois se põem em voga a questão do custo de operação. Dadas condições iniciais, qual o melhor trajeto para se atingir um conjunto de condições finais? O lançamento de uma bomba à distância explora bem esse conceito. Trata-se de um dispositivo que será destruído e que as condições iniciais e finais estão fixas e são conhecidas. A questão é que não se tem necessidade de se obter um sistema complexo de navegação autônoma, com capacidade de tomada de decisão em tempo real para uma atividade como essa. Para esse caso, convém a criação de um sistema de controle com realimentação que garanta a permanência do veículo nessa trajetória de referência, dentro de uma tolerância previamente definida. Nas situações em que o alvo é móvel e não inercial faz-se necessário o uso de tecnologia mais avançada para determinar a sua trajetória. A posição do alvo é constantemente realimentada no sistema através de dispositivos de identificação usando calor ou visão computacional. Esses são dispositivos que não serão abordados nesse trabalho.

O escopo desse trabalho é a determinação de trajetórias sub-ótimas para um míssil Ar-Terra. Outro problema associado é a escolha por um método para busca dessa trajetória dentro de uma malha de possibilidades. Para isso será utilizado um sistema de busca informada denominado A* ("Aestrela"). Como foi colocado acima, as trajetórias encontradas serão do

tipo sub-ótimas. De fato, parece que há uma incompatibilidade, pois, é possível provar que o A* gera uma trajetória ótima, mas a proposta central desse trabalho é garantir que a solução encontrada é muito boa, porém não necessariamente a melhor. O problema principal (lançamento de uma bomba) é de caráter dinâmico e acontece dentro de um domínio continuo e infinito. Para que o A* seja utilizado, necessariamente muitos pontos desse domínio contínuo serão excluídos até que se tenha uma discretização do espaço. O algoritmo de busca vai atuar dentro desse subconjunto de pontos e vai achar uma solução ótima, dentro desse subconjunto, dessa forma, a resposta desse programa vai ser um vetor, de cardinalidade finita de pontos. Quando a resposta ao problema principal for apresentada, para que a trajetória seja contínua será necessária uma inserção de pontos, entre os que compunham a solução ótima. Nesse momento é possível considerar que o resultado que era ótimo não tem mais essa característica, pois caso fosse possível fazer uma busca dentro do domínio infinito (que é o real) a resposta, provavelmente, seria outra. Outra justificativa para o fato da "sub-optimilidade" da trajetória é que as possibilidades, dentro de cada iteração do programa de busca também foi discretizada. Em uma situação de domínio contínuo, as possibilidades de mudança de estado, para cada ponto são infinitas e não poucas (que é o que deve ser feito para que o algoritmo de busca tenha êxito). Por esses motivos atribui-se essa idéia de que não se chega a soluções ótimas para esse problema principal e sim soluções sub-ótimas.

2. O PROBLEMA PRINCIPAL

A motivação para a realização desse trabalho se dá pelo problema do lançamento de um dispositivo bélico (bomba) que apresenta condições iniciais e objetivo bem definidos. Os pormenores do problema são apresentados a seguir:

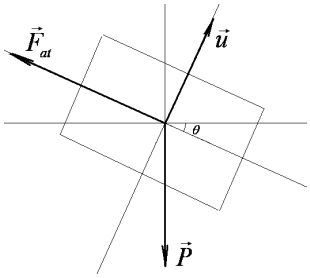


Figura 2.1 – Diagrama de corpo livre para o problema principal

Será lançada uma bomba de massa $M = 500Kg$, com uma velocidade de $v_0 = 250m/s$ na direção horizontal. Juntamente com essa bomba se encontra um dispositivo de controle capaz de receber sinais de entrada, o qual é transformado numa força de controle $\vec{u}$, responsável por controlar um atuador que vai manter o dispositivo dentro da trajetória prevista. São conhecidos os valores de velocidade e ângulo finais. O objetivo é que a bomba chegue à posição destino na direção vertical, porém sendo admitida uma tolerância de mais ou menos 2,5º.

Denotando por v_h e v_v as velocidades horizontal e vertical, respectivamente, da bomba, em um determinado instante, g a aceleração da gravidade local e kv^2 como o módulo da força de arrasto, a que a bomba estará submetida, é possível explicitar as equações do movimento. A força de controle $\vec{u}$ trabalha na direção perpendicular à velocidade e é limitada. Para esse trabalho os valores dos módulos da força de controle $\vec{u}$ estarão entre $0,6mg$ até $1,2mg$. Escrevendo essas equações do movimento, para esse dispositivo quando a direção da sua velocidade é determinada por um ângulo $0 \le \theta \le \frac{\pi}{2}$, obtêm-se o seguinte sistema:

$$\begin{cases} m\dot{v}_h = u\,\mathrm{sen}\,\theta - kv^2\cos\theta \\ m\dot{v}_v = mg - u\cos\theta - kv^2\,\mathrm{sen}\,\theta \end{cases}$$

Equação 2.1

O sistema expresso pela equação 2.1 pode ser reescrito, usando variáveis de estado. A análise sob essa ótica é importante para que se faça um estudo preliminar do movimento, usando ferramentas como “Simulink”. As seguintes definições serão feitas:

$$\begin{cases} x_1 = v_h \ @velocidade\ horizontal \\ x_2 = v_v \ @velocidade\ vertical \\ x_3 = x \ @deslocamento\ horizontal \\ x_4 = y \ @deslocamento\ vertical \end{cases}$$ **Equação 2.2**

Dessa forma as equações de estado ficam na forma:

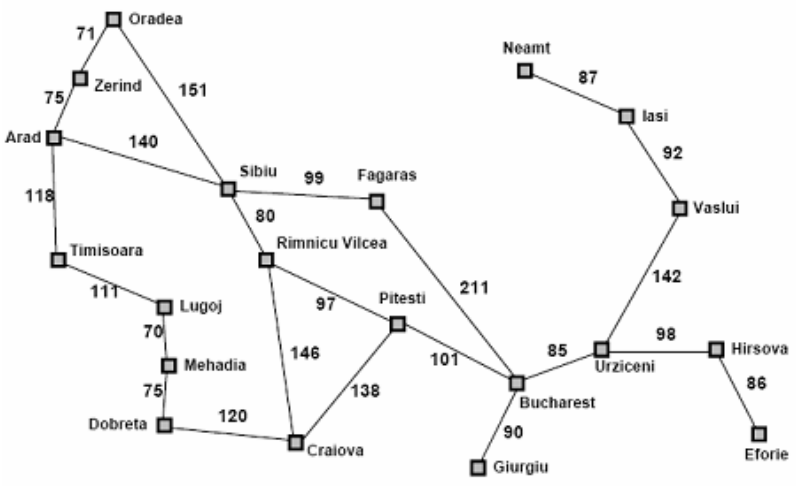
$$\begin{cases} \dot{x}_1 = \dfrac{u}{m}\dfrac{x_2}{\sqrt{x_1^2 + x_2^2}} - \dfrac{k}{m} x_1 \sqrt{x_1^2 + x_2^2} \\ \dot{x}_2 = g - \dfrac{k}{m} x_2 \sqrt{x_1^2 + x_2^2} - \dfrac{u}{m}\dfrac{x_1}{\sqrt{x_1^2 + x_2^2}} \\ \dot{x}_3 = x_1 \\ \dot{x}_4 = x_2 \end{cases}$$ **Equação 2.3**

É razoável que se considere uma situação limite em que a força de arrasto terá o mesmo módulo que a força peso. Dessa forma, a força resultante será igual a zero (o que confere equilíbrio dinâmico), assim, pode-se determinar o valor numérico do fator multiplicativo k, presente na expressão da força de arrasto. Aplicando o raciocínio acima descrito, chega-se à $k = \dfrac{mg}{v_\infty^2}$. Com esses valores bem definidos, é possível a realização de simulações para o problema, dentro do domínio contínuo, para diferentes valores da força de controle $\vec{u}$.

2.1 O A* e seu uso

Trata-se de um viajante que precisa sair da cidade Arad e chegar à cidade Bucharest, percorrendo o menor caminho possível. Qual será esse melhor caminho?

O mapa abaixo [2] mostra a disposição das cidades, a distância entre duas cidades vizinhas e a distância em linha reta de cada cidade à Bucharest. Usando essas informações o algoritmo de busca deve ser capaz de traçar uma rota ótima, em que a grandeza associada à cada decisão e a minimização da distância total percorrida.



Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Figura 2.2 – Possibilidades de movimentação dentro da malha para cada ponto

2.1.1 Algoritmo A* (A estrela)

Essa técnica para busca informada é ainda hoje a mais utilizada, visto a eficácia do seu resultado. A grande idéia é evitar a expansão de posições que já ficaram cara. Trata-se da tentativa da combinação do algoritmo da busca gulosa, que é mais econômica, mas não se mostra completa e nem ótima, com a técnica do custo uniforme que é ineficiente, mas é ótima e completa. Para cada nó n se tem a geração de uma função *f*(*n*) que é formada por duas partes: $f(n) = g(n) + h(n)$ em que *g*(*n*) é denotada por função que expressa o custo real mínimo do ponto inicial até o nó "*n*" e *h*(*n*) que denota a função heurística para cada nó "*n*", que é o custo real mínimo desde "*n*" até um estado-objetivo. No problema do viajante, a heurística utilizada é a distância entre cada nó das possibilidades a partir de "*n*" e o ponto final, sem barreiras intermediárias. Para uma expansão arbitrária, h(n) deve ser estimado, como foi dito, usando uma heurística e num primeiro momento o custo da trajetória até o nó "*n*" não tem que ser o menor. Dessa forma define-se $\hat{f}(n) = \hat{g}(n) + \hat{h}(n)$ em que $\hat{g}(n)$ é o custo da trajetória encontrada até o momento para alcançar "*n*" e $\hat{h}(n)$ é o custo mínimo ***estimado*** desde o ponto "n" até o estado-objetivo. O algoritmo A* expande o nó que estiver associado ao menor valor de função $\hat{f}(n)$.

Como pode ser observado na figura 2.4 [2], do ponto inicial são calculados os valores da função *f*(*n*) para as cidades de ligação com Arad. Soma-se o valor do custo de operação de ir de Arad até cada uma das cidades com a respectiva heurística, que á distância euclidiana entre as cidades e Bucharest. A cidade associada ao menor valor de *f* é tomada como escolha.

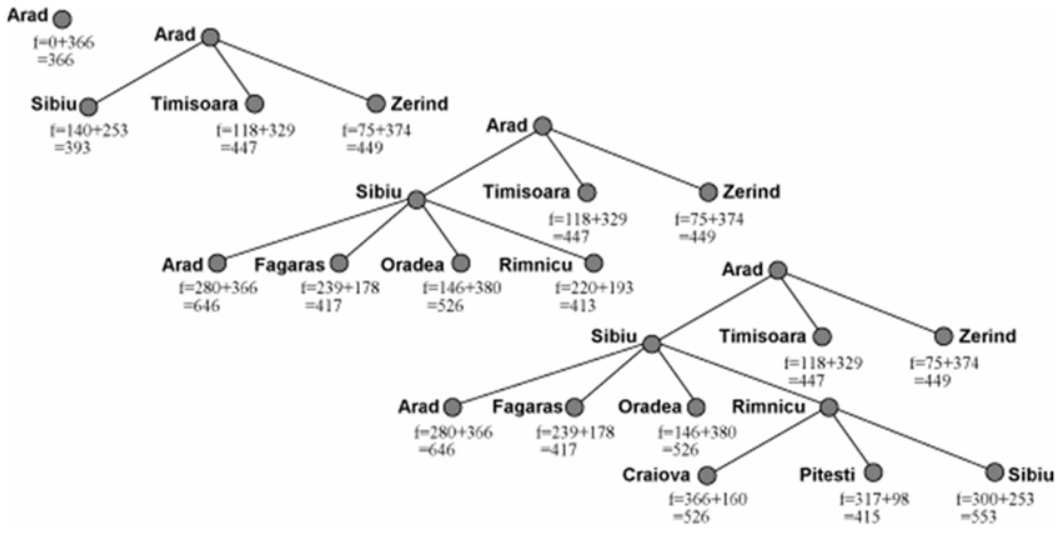


Figura 2.4 – Plano de expansão dos nós usando o algoritmo A*

Esse procedimento é feito de passo a passo. É muito importante dizer que o algoritmo A* possui memória, ou seja, os valores das funções *f* dos pontos que não foram expandidos, mas foram em algum momento tidos como possibilidade dos pontos expandidos, ficam armazenados e sempre que se calculam valores de função *f*(*n*) para novos pontos, observa-se se em algum momento não teve um ponto com valor associado de *f*, menor do que esses que foram por último calculados. Caso haja, o programa deve ser inteligente o suficiente para voltar ao ponto de menor *f* e refazer o caminho.

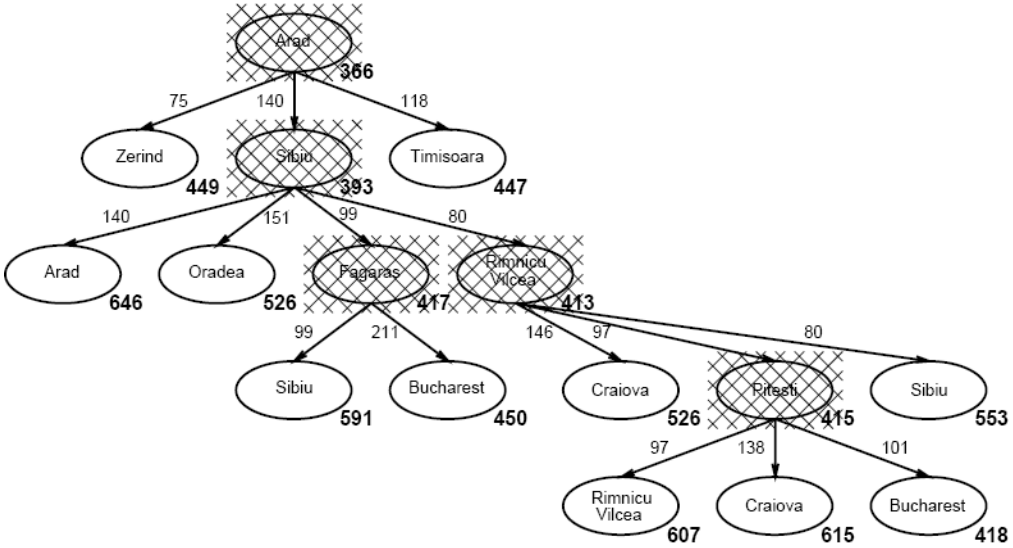


Figura 2.5 – Plano de expansão dos nós usando o algoritmo A*

Continuando os cálculos das funções *f*, a partir do ponto de chegada pela figura 2.5 [2] percebe-se que o valor de *f*, seguindo o caminho Arad. Sibiu, Rimnicu, Pitesti e por fim Bucharest produz um custo de 418 e na segunda iteração se tinha uma opção via Fagaras custando 417. Nesse momento o programa implementado deve ser capaz de voltar à segunda iteração e refazer o caminho, agora via Fagaras. Logo na próxima iteração é visto que todos os caminhos a partir de Fagaras vão produzir custo maior que 418, o qual era o que se tinha na opção anterior, logo o programa deve ser a capacidade de apresentar como solução o primeiro caminho, que tinha custo total 418.

Para esse trabalho o uso do A* se dá como uma opção de ferramenta computacional, dessa forma fica fora do escopo um estudo mais profundo sobre esse algoritmo, porém é possível se demonstrar que a solução encontrada por esse algoritmo tem caráter ótimo dentro das possibilidades dadas.

3. RESOLVENDO O PROBLEMA PRINCIPAL

Retomando o problema inicialmente colocado no capítulo 2, desse trabalho, necessita-se implementar o A*, para se encontrar uma boa trajetória, visto que se tem um ponto final e um ponto inicial bem conhecido e o veículo não tripulado deve atingir o ponto final fazendo um ângulo com a horizontal próximo de 90º.

Antes de ser desenvolvido o algoritmo propriamente dito, é necessário destrinchar as equações de estado de uma maneira mais conveniente. A equação 2.1 mostra o sistema

$$\begin{cases} m\ddot{x} = u\,\mathrm{sen}\,\theta - kv^2\cos\theta \\ m\ddot{y} = mg - u\cos\theta - kv^2\,\mathrm{sen}\,\theta \end{cases}$$ em que:

$$\ddot{x} = \frac{d(v\cos(\theta))}{dt} = \dot{v}\cos(\theta) - \dot{\theta}v\,\mathrm{sen}(\theta) \qquad \textbf{Equação 3.1}$$

$$\ddot{y} = \frac{d(v\,\mathrm{sen}(\theta))}{dt} = \dot{v}\,\mathrm{sen}(\theta) + \dot{\theta}v\cos(\theta) \qquad \textbf{Equação 3.2}$$

Substituindo (Equação 3.1) e (Equação 3.2) no sistema acima vem:

$$\begin{cases} \dot{v}\cos(\theta) - v\dot{\theta}\,\mathrm{sen}(\theta) = \dfrac{u}{m}\mathrm{sen}\,\theta - \dfrac{kv^2}{m}\cos\theta \\ \dot{v}\,\mathrm{sen}(\theta) + v\dot{\theta}\cos(\theta) = g - \dfrac{u}{m}\cos\theta - \dfrac{kv^2}{m}\mathrm{sen}\,\theta \end{cases}$$ **Equação 3.3**

A partir da equação 3.3, pode ser feita uma discretização, considerando a evolução de iteração a iteração na compilação do programa que vai resolver o problema.

Todo esse problema poderia ser evitado se o valor da força de controle u fosse conhecida. Assim, bastaria introduzi-la no simulink e esse geraria sem dificuldades a trajetória correspondente. O fato é que essa força é desconhecida. A proposta desse problema é, em sua essência, gerar a trajetória do veículo autônomo, mas juntamente com isso produzir essa força de controle u e com esse resultado fazer a simulação em questão no simulink, provando assim que o algoritmo funciona. Abaixo, se encontra a eq. 3.3, no formato discreto.

$$\begin{cases} \left(\dfrac{v_n - v_{n-1}}{\Delta t}\right)\cos(\theta_n) - \left(\dfrac{\theta_n - \theta_{n-1}}{\Delta t}\right)v_n\,\mathrm{sen}(\theta_n) = \dfrac{u_n}{m}\mathrm{sen}(\theta_n) - \dfrac{k(v_n)^2}{m}\cos(\theta_n) \quad (I) \\ \left(\dfrac{v_n - v_{n-1}}{\Delta t}\right)\mathrm{sen}(\theta_n) + \left(\dfrac{\theta_n - \theta_{n-1}}{\Delta t}\right)v_n\cos(\theta_n) = g - \dfrac{u_n}{m}\cos(\theta_n) - \dfrac{k(v_n)^2}{m}\mathrm{sen}(\theta_n) \quad (II) \end{cases}$$ **Equação 3.4**

Para calcular o valor de v_n, em função de θ_n e Δt, calcula-se $(III) = (I).\cos(\theta_n) + (II).\mathrm{sen}(\theta_n)$, assim:

$$(v_n)^2 + \frac{m}{K\Delta t}(v_n) - \frac{m}{K\Delta t}(v_{n-1} + g\Delta t\,\mathrm{sen}(\theta_n)) = 0$$ **Equação 3.5**

Logo resolvendo a equação do segundo grau em v_n chega-se que:

$$v_n = -\frac{m}{2K\Delta t} + \sqrt{\left(\frac{m}{2K\Delta t}\right)^2 + \frac{m}{K\Delta t}(v_{n-1} + g\Delta t\,\mathrm{sen}(\theta_n))}$$ **Equação 3.6**

Para calcular o valor de u_n, em função de θ_n e Δt, calcula-se $(IV) = (I).\mathrm{sen}(\theta_n) - (II).\cos(\theta_n)$, assim:

$$u_n = mg\cos(\theta_n) - \frac{m}{\Delta t}v_n(\theta_n - \theta_{n-1})$$ **Equação 3.7**

Como pode ser observado, uma vez que se tenha um valor inicial para v_n, os outros valores das outras iterações sairão de maneira recursiva. E tendo calculado v_n, obtêm-se o valor de u_n correspondente.

Para o exemplo mostrado acima, o A* trabalhava com a evolução de um sistema (x,y) em que para cada iteração era possível fazer uma escolha entre cinco possíveis novas posições. Nesse caso, a evolução dará num sistema (t,θ), em que *t* corresponde ao tempo de movimento do veículo e θ é o ângulo que o vetor velocidade faz com a horizontal. Para resolver esse caso foi considerado que a cada iteração há três opções de escolha:

a) $(t,\theta) \Rightarrow (t+\Delta t, \theta+\Delta\theta)$ (aumento do tempo e aumento do ângulo θ)
b) $(t,\theta) \Rightarrow (t+\Delta t, \theta)$ (aumento do tempo apenas)
c) $(t,\theta) \Rightarrow (t+\Delta t, \theta-\Delta\theta)$ (aumento do tempo e diminuição do ângulo θ)

O passo escolhido para o tempo é de 0,1 segundos e o valor do incremento angular máximo por iteração é de 0,6º. O objetivo é que o veículo autônomo alcance o alvo, fazendo um ângulo de 90º com a horizontal. Como foi comentado acima, o algoritmo A* toma decisões baseado no menor custo dentre as opções em cada iteração. Para esse problema foi proposto uma função custo da seguinte forma:

$$F(x,\theta)=\left[\alpha(x-x_F)^2+\beta\frac{(\theta-\theta_F)^2}{(x-x_F)^2}\right].10^{-6} \qquad \textbf{Equação 3.8}$$

Os valores de x_n e y_n de cada iteração são calculados pelas expressões:

$$x_n = x_{n-1} + v\cos(\theta_n) \qquad \textbf{Equação 3.9}$$

$$y_n = y_{n-1} - v\,\text{sen}(\theta_n) \qquad \textbf{Equação 3.10}$$

A motivação para a escolha dessa função como função custo é que ao passo que a diferença entre a posição horizontal da iteração "*n*" e a posição horizontal do alvo se torne muito pequena, a fração da função $F(x,\theta)$ tende a crescer e para que isso não aconteça, o ângulo deve aumentar rapidamente, afim de que a diferença $(\theta-\theta_F)$ seja bem pequena. A determinação das constantes α e β foi totalmente experimental, ou seja, já com o programa em funcionamento, fixou-se uma das constantes e procurou-se a outra que tornava o resultado final o mais próximo do objetivo. Dessa forma, coloca-se que há outros pares (α,β) que tornam a função custo uma função admissível e que atenda o objetivo final.

A seguir será pormenorizado o procedimento seguido pelo algoritmo em questão. Suponha que na iteração anterior o veículo esteja voando à t_{n-1} segundos e com um ângulo θ_{n-1} com a horizontal, a partir dessa situação, então, começa-se a n-ésima iteração.

1. Evolui-se o tempo e o ângulo positivamente. Com o valor de v_{n-1} e os novos valores de t e θ, calcula-se v_{nA} (Equação 3.6) e logo em seguida u_{nA} (Equação 3.7). Verifica-se se $u_{nA}\le u_{máx}$. Em caso afirmativo, calcula-se a função $F(x_{nA},\theta_{nA})$ e armazenam-se todos esses valores (v_{nA}, u_{nA}, θ_{nA}, t, $F(x_{nA},\theta_{nA})$) numa matriz mestre. Em caso negativo, atribui-se $u_{nA}\leftarrow u_{máx}$ e se faz o recálculo do ângulo chegando num valor $\hat{\theta}_{nA}$. Recalcula-se $\hat{v}_{nA}$, $F(x_n,\hat{\theta}_{nA})$ e se armazenam todos esses valores ($\hat{v}_{nA}$, u_{nA}, $\hat{\theta}_{nA}$, t, $F(x_n,\hat{\theta}_{nA})$) numa matriz mestre.
2. Faz-se o mesmo considerando a evolução temporal apenas.
3. Faz-se o mesmo considerando a evolução temporal e o decremento angular.
4. Percorre-se toda a coluna da matriz mestre que contem os valores de $F(x,\theta)$ e se escolhe o menor. O programa vai tomar como valores de v_n, u_n, θ_n, t, os valores pertencentes à essa linha de menor função custo.
5. Calcula-se o valor de y_n (Equação 4.10).
6. Registram-se na matriz memória todos esses valores de v_n, u_n, θ_n, t, y_n bem como a iteração correspondente.
7. Repete-se a operação até que $y_n<10\,m$.

Para se determinar as constantes α e β que figuram na expressão da função custo (Equação 3.8) foi feita uma estimativa inicial do tipo $\alpha=\beta=1$, que resultou na convergência do resultado, porém com falta de suavidade na trajetória. Fixou-se o valor de β em $\beta=100$ e variou-se o valor de α até que a trajetória fosse dada por curvas suaves e essa conduzisse a um lugar o mais próximo possível do ponto de chegada. O mais interessante é que para cada

valor de $u_{máx}$ admitido, mantendo-se constante o valor de β, α muda consideravelmente. A tabela abaixo mostra os valores de α e β, pra cada valor de $u_{máx}$ admitido.

Tabela 3.1 Valores experimentais dos parâmetros da função custo

$u_{máx}\ (N)$	$\alpha\ (x10^{-6} m^{-2})$	$\beta\ (m^2.rad^{-2})$	$v_\infty\ (m/s)$
0,6 Mg	12,000	100	60
0,7 Mg	6,900	100	60
0,8 Mg	4,750	100	60
0,9 Mg	4,600	100	60
1,0 Mg	8,100	100	60
1,1 Mg	7,755	100	60
1,2 Mg	7,396	100	60

É bom frisar que os valores acima mostrados são totalmente experimentais e que pequenas variações, da ordem de décimos de α, levam a resultados bem diferentes, quanto à convergência e precisão do resultado. Talvez, para outro foco do estudo seja interessante analisar essa sensibilidade. Para esse trabalho isso não será feito e tais valores serão tomados como dados de entrada.

4. RESULTADOS

O algoritmo foi implementado e nesse ponto do trabalho será colocado à prova se a utilização de métodos de busca (Inteligência Artificial), que é em geral utilizado para problemas discretos apresenta, de fato, eficiência e precisão no resultado do problema principal que se trata de um problema dinâmico. Abaixo será mostrado o resultado da aplicação do algoritmo em questão. Uma das respostas do programa é exatamente a evolução da altura do veículo em função da posição horizontal e outra saída é a evolução da força de controle u, em função do tempo. As condições iniciais podem ser a mais diversificadas possíveis, porém, apenas para fim de ilustração está mostrado um exemplo com as seguintes condições iniciais:

$v_0 = 250\ m/s,\ H_0 = 3000\,m,\ x_F = 3500\,m,\ \theta_F = 90^o,\ \theta_0 = 0^o$ e $u_{\max} = 0.9P$

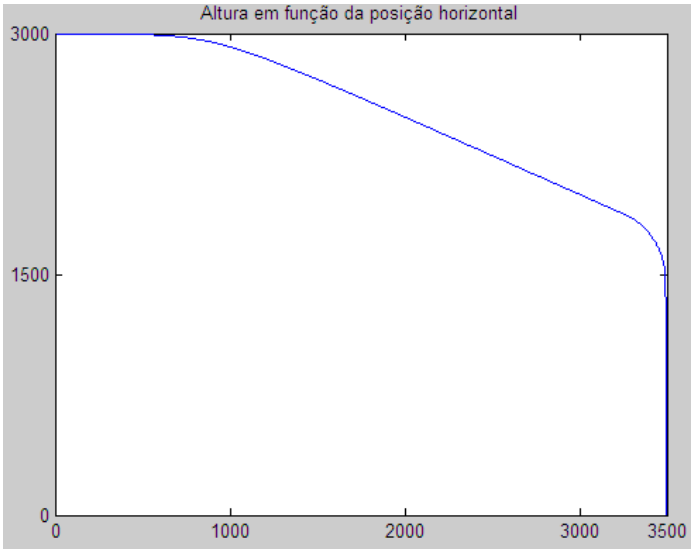


Figura 4.1 – Trajetória do veículo autônomo gerada pelo algoritmo de busca A*

Como pode ser observado o algoritmo de busca mostrou uma trajetória bem suave e que obteve uma precisão excelente, com erro inferior a 0,17%, visto que o ponto de chegada pelo programa implementado foi de 3494 m. Esse mesmo programa gerou outro gráfico que

confirmam as hipóteses feitas a *anteriori* o qual é o do módulo do vetor velocidade em função do tempo, que se encontra abaixo.

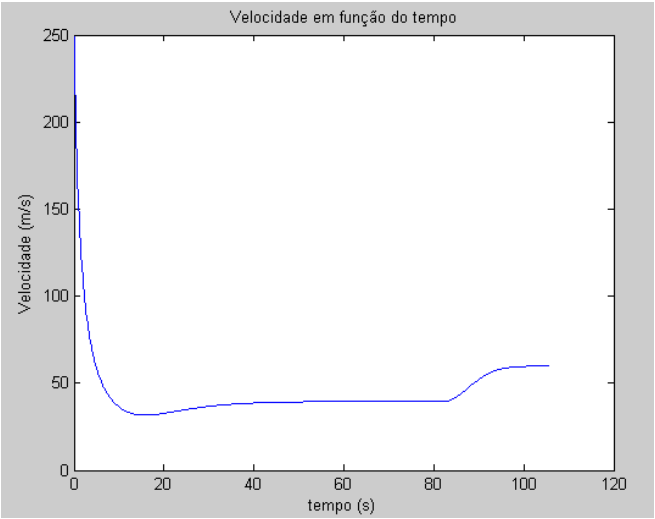


Figura 4.2 – Gráfico da velocidade em função do tempo

Logo no início foi feita a hipóteses de que o sistema, no final do movimento estaria numa situação de equilíbrio dinâmico, em que o peso se igualaria ao arrasto. Isso pode ser observado na figura acima, em que a velocidade tende a ficar constante e igual a 60 m/s.

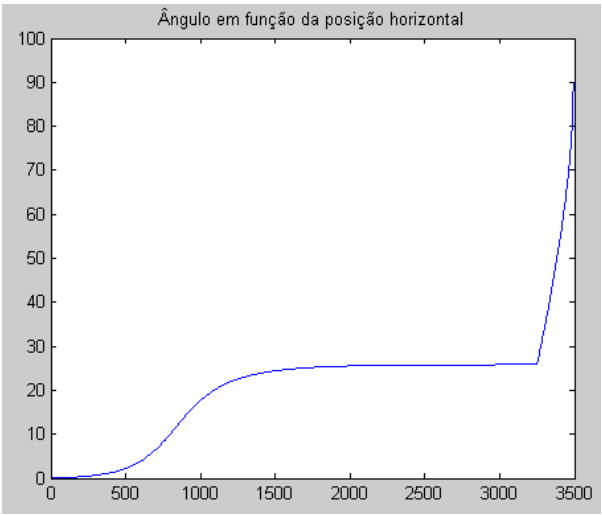


Figura 4.3 – Gráfico da do ângulo em função da posição horizontal

A figura 4.3 mostra a evolução do ângulo com a horizontal em função do tempo. Como pode ser observado trata-se de um resultado de excelente precisão. O valor do ângulo final deveria ser de 90º e o programa gerou um ângulo de 89,96º. Esse resultado proporciona um erro relativo de menos de 0,05 %. Em termos práticos obteve-se a exatidão.

A discretização do sistema dinâmico foi feito de maneira bem rudimentar. A falta de rigor matemático nas aproximações, nos cálculos dos parâmetros e nas hipóteses simplificadoras pode levar a outras soluções menos refinadas. No caso do trabalho em questão foi feita uma comparação com a solução conseguida pelo algoritmo de busca e a solução tirada da simulação em Simulink. Esse estudo consiste, apenas em verificar se os cálculos foram demasiados grosseiros, pois o Simulink possui em sua implementação métodos muito mais sofisticados e finos para resolução de uma equação diferencial, do que os métodos usados na busca. Não se trata de demonstração, que só caberia sob a ótica da resolução das equações dinâmicas dentro do meio contínuo, mas sim uma verificação do resultado. O *link* entre os dois métodos de cálculo (Simulink e a busca informada) foi exatamente o vetor *u*. O valor da força de controle *u*, para cada iteração, foi armazenado em um vetor (na implementação do programa), dessa forma, aplicou-se esse vetor como entrada no Simulink (obviamente mantendo o passo de integração igual ao incremento temporal usado na busca). Essa

informação, dentro do Simulink será processada de uma maneira mais sofisticada, através de cálculo numérico enquanto que na busca foi utilizado uma abordagem puramente algébrica e recursiva.

Como foi comentado acima, a partir da equação 2.3 gerou-se um diagrama de blocos para simulação em Simulink. O diagrama abaixo expressa o sistema dinâmico proposto pelo modelo da equação citada.

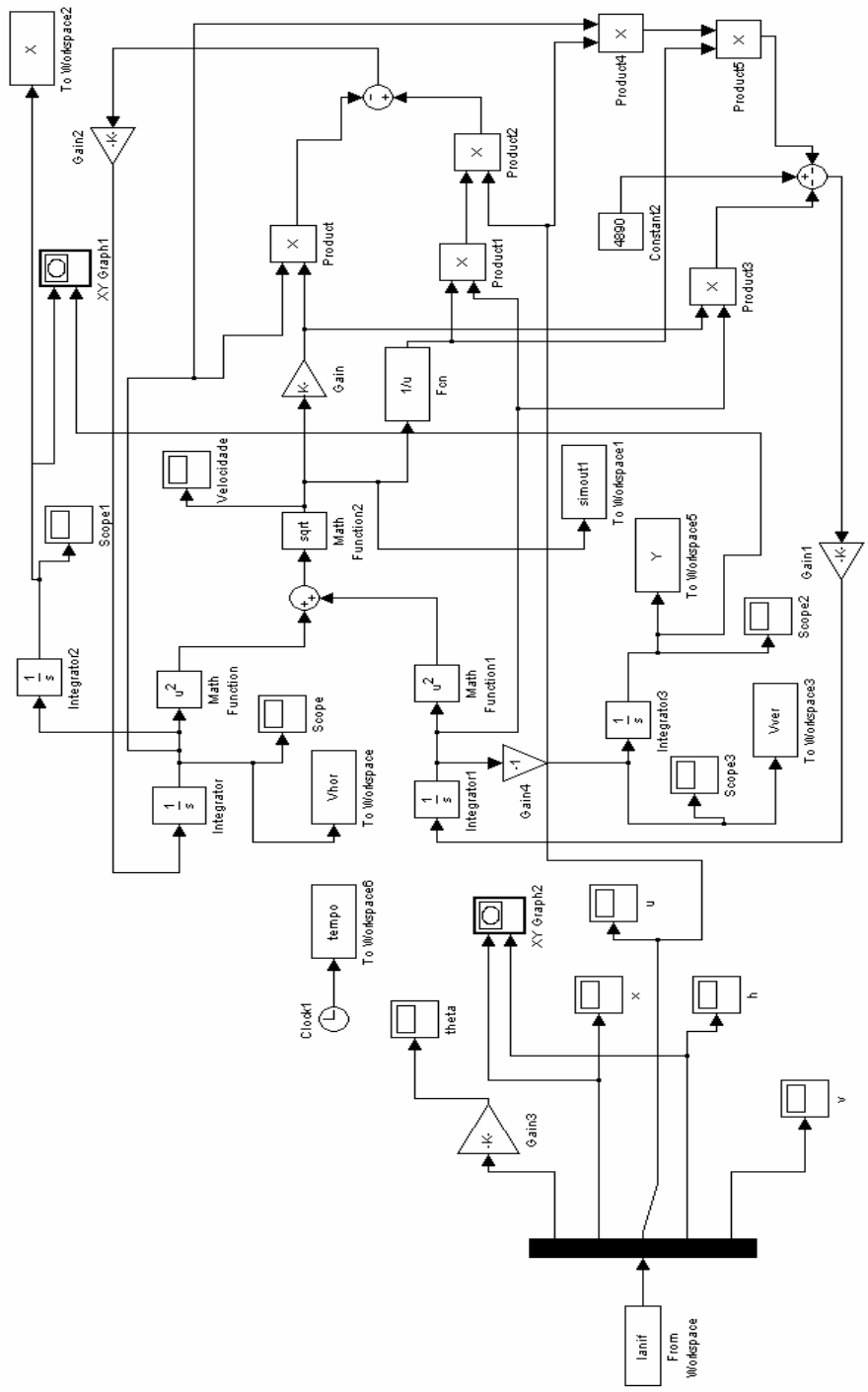


Figura 4.4 – Diagrama de blocos para a simulação do sistema da eq. 2.3

A simulação acima foi executada sob os mesmos valores de condição inicial considerados para o programa implementado usando o algoritmo A*. A função “DEMUX” busca as variáveis do programa já executado e disponibiliza como entradas do Simulink, vetores e matrizes do “workspace”. Isso foi útil, pois se utilizou a saída do vetor contendo os valores de “*u*” para cada valor de “*t*” como entrada na simulação. A figura abaixo mostra o gráfico da trajetória conseguida pela simulação usando diretamente as equações do sistema dinâmico em questão.

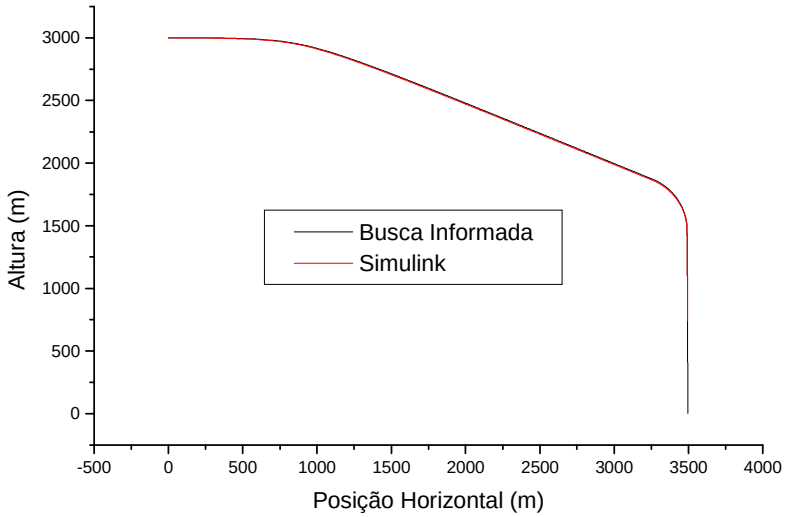


Figura 4.5 – Ambas as trajetórias sob a mesma abscissa

Como foi comentado acima, a importância da figura acima é em especial para verificar a funcionabilidade do programa de busca informada implementado. Os gráficos são idênticos não apenas em aparência, mas também o é em conteúdo, assim a veracidade da convergência do A* para a determinação de trajetórias para míssil Ar-Terra está **verificada**.

5. CONSIDERAÇÕES FINAIS E CONCLUSÃO

Esse trabalho resultou em um grande desafio, pois pesava sobre o autor, a responsabilidade de adequar técnicas, que funcionam de maneira excelente para um tipo de problema, a outra natureza de problema. Como foi comentado, algoritmos de busca informada funcionam muito bem para determinação de caminhos para robôs, resolução de jogos e outros tipos de problema, porém, de natureza discreta. O domínio finito e de baixa cardinalidade é uma condição primordial para o bom funcionamento de um algoritmo de busca informada. Embora o problema principal apresente um domínio infinito e contínuo, foi possível fazer uma discretização razoável, mas que consistiu em uma malha com mais de 50000 pontos. Se o programa fosse trabalhar em todos os pontos da malha o uso desse método seria inviável, contudo, a escolha de uma boa função heurística permitiu que durante a execução do programa muitos pontos não fossem abertos para análise, visto que para essa heurística o objetivo estava sempre sendo levado em consideração. Esse fato ocasionou em uma redução de processamento gerando assim uma convergência mais rápida.

O estudo poderia ter sido finalizado no momento em que se obteve o resultado do programa que é a trajetória de referência, mas considera-se importante que houvesse uma comprovação de que essa trajetória encontrada fosse realmente adequada. Dessa maneira fez-se uso do Simulink para que ao introduzir uma simulação do problema com suas equações de estado e respeitando as mesmas condições iniciais e de contorno que foram utilizadas no programa de busca, fosse gerada a trajetória de referência. Como foi visto as trajetórias de referência foram as mesmas, o que mostra que o uso do sistema de busca informada num problema contínuo foi uma utilização razoável e uma boa idéia, pois basta saber as condições iniciais e de contorno, além do torque máximo que o sistema pode desenvolver, para que a trajetória seja gerada. Talvez seja óbvio que não são todos os algoritmos de busca informada, que vão ser úteis num problema como esse, mas foi mostrado nesse trabalho que com uma boa heurística consegue-se obter resultados surpreendentes. Para a resolução desse problema, em especial o uso da busca gulosa, ao invés do A* também geraria uma boa solução, visto que a heurística utilizada é admissível. Com relação a obtenção de uma equação para essa trajetória

de referência pode ser feita usando interpolação polinomial, com os pontos obtidos desse método discreto utilizado.

Trata-se de uma atividade nova e, portanto, muita coisa desse estudo pode ser melhorada. Um dos itens que merecem uma revisão é o modelo aplicado, pois seria mais razoável que a força de controle também fosse dependente do modulo do vetor velocidade. Está provado que o método funciona, mas o uso de um modelo mais próximo da realidade é com certeza um fator que gera mais confiabilidade ao estudo. Outra proposta que pode ser feita a partir desse trabalho é a utilização de outros algoritmos de busca, informada, que precisem de menos processamento. Não é garantia que gerarão uma boa trajetória, mas pode ser que para o caso desse modelo, ou ainda um modelo desse problema (mais melhorado) tais algoritmos sejam suficientes e o resultado seja satisfatório. Outro bom ramo é o estudo das constantes da função $h(n)$ (equação 4.8) que aqui foram atribuídas de maneira empírica, bastava saber qual o valor de $u_{máx}$ e tomar como fixo uma das constantes, fez-se o programa funcionar várias vezes até que o resultado fosse satisfatório. Em fim é possível, a partir desse artigo, a geração de outros trabalhos, além da expansão desse próprio para um nível mais elevado e detalhado.

6. REFERÊNCIAS BIBLIOGRÁFICAS

[1] Russell, Stuart Jonathan. Inteligência Artificial: tradução da segunda edição/ Stuart Russell, Peter Norvig: Tradução da Publicare Consultoria – Rio de Janeiro: Elsevier, 2004 – 4º impressão. 1021 p.

[2] Ribeiro, Carlos Henrique Costa. Algoritmos de Busca Informada: Problemas Satisfação de Restrições. CTC-15/CT-215 aulas 5 e 6 – São José dos Campos – 2006. 45 p.

[3] Dias, Leonardo Alves. **Planejamento de trajetória para plataforma móvel em ambiente incerto .** Trabalho de graduação. Instituto Tecnológico de Aeronáutica. Divisão de Engenharia Eletrônica. Orientadores: Cairo Lúcio Nascimento Júnior e Eduardo Hisasi Yagyu. São José dos Campos : CTA/ITA, 1998 . 79 p.

[4] Nascimento Júnior, Cairo Lúcio do. **Inteligência artificial em controle e automação.** Livro – Português. São Paulo : Edgard Blücher, 2000. 218 p.

[5] Yepes , Javier Lovera. New Algorithms for Aircraft Intent Inference and Trajectory Prediction. Journal of Guidance, Control and Dynamics – Vol. 30, No. 2. Março-Abril 2007. 13 p.