UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

Controle de veículo aéreo não-tripulado do tipo helicóptero baseado em redes neurais artificiais

Antônio Péricles Bonfim Saraiva de Oliveira

Orientador: Prof. Dr. Luiz Marcos Garcia Gonçalves

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Computação da UFRN (área de concentração: Engenharia de Computação) como parte dos requisitos para obtenção do título de Mestre em Ciências.

Número de ordem PPgEEC: M358
Natal, RN, Julho de 2012

Divisão de Serviços Técnicos

Catalogação da publicação na fonte. UFRN / Biblioteca Central Zila Mamede

Oliveira, Antônio Péricles Bonfim Saraiva de.

Controle de veículo aéreo não-tripulado do tipo helicóptero baseado em redes neurais artificiais / Antônio Péricles Bonfim Saraiva de Oliveira. - Natal, RN, 2012.

53 f.; il.

Orientador: Prof. Dr. Luiz Marcos Garcia Gonçalves.

Dissertação (Mestrado) - Universidade Federal do Rio Grande do Norte. Centro de Tecnologia. Programa de Pós-Graduação em Engenharia Elétrica Computação.

1. Aprendizagem Supervisionada - Dissertação. 2. VANT - Dissertação. 3. Redes Neurais Artificiais - Dissertação. 4. Neuro-controlador - Dissertação. 5. Flightgear - Dissertação. I. Gonçalves, Luiz Marcos. II. Universidade Federal do Rio Grande do Norte. III. Título.

RN/UF/BCZM CDU 004.032.26

Controle de veículo aéreo não-tripulado do tipo helicóptero baseado em redes neurais artificiais

Antônio Péricles Bonfim Saraiva de Oliveira

Dissertação de Mestrado aprovada em 13 de agosto de 2012 pela banca examinadora composta pelos seguintes membros:

Prof. Dr. Luiz Marcos Garcia Gonçalves (orientador) ..... DCA/UFRN

Prof. Dr. Luiz Eduardo Cunha Leite (examinador interno) ..... DCA/UFRN

Prof. Dr. Ricardo Cordeiro de Farias (examinador externo) ..... Pesc/UFRJ

Resumo

Propõe-se a criação de um controle neural (neurocontrolador) baseado na aprendizagem supervisionada, com uma rede neural artificial (RNA), sem a modelagem por espaço de estados, utilizando o Flightgear como simulador e ambientes de testes integrado com módulos de coleta de dados e controle. Diversas arquiteturas da RNA foram testadas a fim de que as mesma tivesse a eficácia pretendida nos diversos procedimentos (decolagem, pairagem, deslocamento e pouso). Testes com as RNA treinadas foram realizados até que fosse encontrada uma que pudesse gerar as respostas necessárias atendendo aos requisitos de eficiência e estabilidade necessários ao controle do VANT.

Palavras-chave: Aprendizagem supervisionada, *VANT*, rede neurais artificias, neurocontrolador, Flightgear.

Abstract

We propose the creation of a neural control (neurocontroler) based on supervised learning, with an artificial neural network (ANN), without the state space modeling using Flightgear as the simulator and test environments with integrated modules for data collection and control. Several ANN architectures were tested in order that the same had the desired efficacy in various procedures (takeoff, hover, displacement and landing). Testing with the trained ANN were performed until it was found one could generate the necessary responses meeting the requirements of efficiency and stability necessary to control the UAV.

Palavras-chave: supervised learning, *UAV*, artificial neural network, neurocontroler, Flightgear.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 O problema

Como poderíamos desenvolver um sistema de controle para diversos tipos de VANT's, sem o retrabalho ou a especificidade individual de cada um? Sabemos que um sistema, através de suas variáveis de estado, pode caracterizar completamente um sistema e desta forma podemos gerar sinais de controle com o propósito de controla-lo. Também sabemos o conjunto de variáveis de estado e sinais de controle são únicos para cada sistema. Alterações, mesmo que pequenas, em um VANT, necessitam de um novo conjunto de variáveis de estado e por consequência um novo conjunto de sinais de controle, ou seja, necessitamos de um novo controlador. Um piloto humano bem treinado em um tipo específico de VANT é capaz de conduzir, mesmo que não seja de forma ótima, outro VANT semelhante sem a necessidade de um treino. Um sistema de controle poderia controlar de forma ótima um VANT, porém estaríamos limitados a um único modelo ou a alguns poucos modelos com variações sutis. Variações mais amplas tornariam o sistema de controle ineficiente para um determinado modelo.

A principal deficiência dos pilotos automáticos existentes é a necessidade do desenvolvimento de um sistema de controle para cada modelo. Notadamente, cada modelo possui um conjunto de variáveis de estado que condensa as características de cada modelo, e por consequência, um necessita de um controlador específico.

A tarefa de construção de um sistema de controle recai na problemática da necessidade de um modelo (função de espaço de estados) que caracterize matematicamente e completamente uma planta, no nosso caso, um VANT a fim de que o mecanismo de controle tenha a eficácia necessária e requerida para manter a estabilidade durante as tarefas executadas. Para cada modelo distinto de VANT é requerido um modelo matemático diferente. Por consequência, ajustes nos sistemas de controle se fazem necessários. Grande parte destes ajustes é um processo exaustivo [Buskey et al. 2001] e que demanda tempo considerável. Como poderíamos ter um sistema de controle que não necessitasse do modelo matemático ?

A motivação para o desenvolvimento deste trabalho é possibilidade da construção de um controlador neural que tenha características de um piloto humano bem treinado e tenha capacidade de controlar diversos tipos de VANTs sem a necessidade de um novo treinamento para modelo distintos de VANT.

1.2 O que fez

A criação de um controlador para um VANT (veículo aéreo não-tripulado) passa pelo estágio inicial de modelagem e estende-se até os ajustes do próprio controlador para com a planta. O desenvolvimento de um controlador que não utiliza tais etapas necessita de uma abordagem baseada em rede neural artificial (RNA). Procedimentos de decolagem, deslocamento, pairagem e pouso necessitam de controles estáveis, robustos e eficazes. Devido à natureza do VANT de ser não-linear, MIMO (Multiple Input Multiple Output) e ter a característica de forte acoplagem cruzada através de diferentes variáveis de estado [Hashimoto et al. 2000], controles tradicionais demandam considerável tempo para sua construção e tais controles não podem ser intercambiados entre outros modelos de VANT, sendo necessário uma nova modelagem.

Testes que possam garantir que um determinado controlador esteja ajustado e atenda aos requisitos de estabilidade, robustez e eficácia, para um determinado VANT, atualmente são realizados em simuladores e posteriormente com testes em campo para ajustes mais específicos. O uso de simuladores visa a redução de custos e redução do tempo de testes. A demanda pela integração do simulador com o método de testes dos controles se faz importante na medida que o ambiente completo de simulação e testes tenha capacidade de armazenar dados para posterior análise. A possibilidade de reproduzir um mesmo ambiente para que diversos controladores possam ser comparados é uma característica obrigatória e naturalmente desejável visto que podemos realizar testes comparativos no mesmo ambiente. As variações observadas dever-se-ão à particulariedades de cada controle e não do ambiente.

1.3 Como fez

A primeira demanda neste trabalho foi a coleta de dados de um piloto real realizando diversos procedimentos (decolagem, deslocamento, pairagem e pouso), para o treino da RNA que originaria o neurocontrolador. Estes dados foram coletados com um piloto real realizando diversos procedimentos no simulador. Um módulo de coleta de dados foi criado para que os dados de telemetria (tanto do piloto real quanto do simulador) fossem armazenados para posterior análise e treino na RNA. Uma vez que tais dados foram analisados e filtrados, a RNA foi treinada. Após o treino da primeira RNA e testes no simulador com o neurocontrolador (módulo de controle), ficou evidente, através dos experimentos, que seria necessário ajustes na arquitetura da RNA pois a mesma não estava gerando as respostas necessárias e satisfatórias. Diversas arquiteturas foram testadas com diferentes quantidades de camadas ocultas e cada camada com diferentes quantidades de neurônios artificiais. Uma determinada arquitetura foi treinada e com esta rede foram realizados testes. Este ciclo de treino-teste foi realizado até que uma arquitetura para a RNA gerasse respostas estáveis e eficazes para os procedimentos (decolagem, deslocamento, pairagem e pouso) do VANT no simulador (figura 1.1).

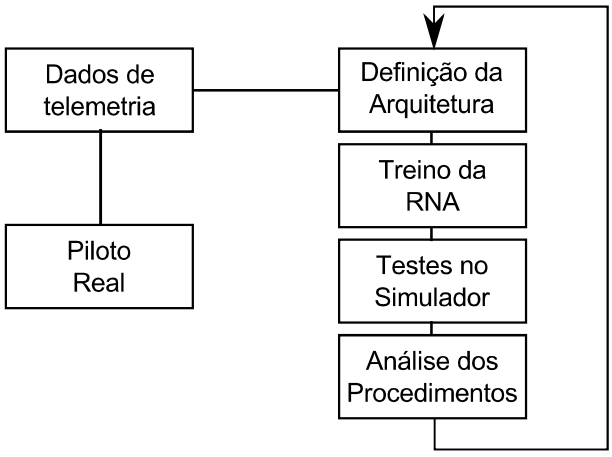

Figura 1.1: Ciclo de treinamento e teste da arquitetura.

1.4 Contribuições e objetivos

Com este trabalho geraremos subsídios para desenvolvimento de neurocontroladores através de um ambiente simulado, testes em modelos distintos de VANT e a capacidade de realizar testes comparativos (usando o mesmo ambiente controlado para todos os testes) entre controladores tradicionais e diversos outros métodos de controle relatados [Ahmed et al. 2008] que seriam : LQBA, H_2, H_∞, μ-syntesis, inversão dinâmica, linearização de entrada-saída, controle não-linear H_∞, *flatness differential, sliding mode, backstepping*, controlador *fuzzy* e controle baseado em rede neural artificial (RNA).

O objetivo deste trabalho é o desenvolvimento de um controle neural baseado em rede neural artificial (RNA) com a característica de robustez necessária para o controle não-ótimo, de VANTs com aspectos semelhantes, sem a necessidade de novo treino.

Como objetivos secundários teremos a construção de um sistema de telemetria para coleta de dados de um piloto humano, construção de um sistema de controle baseado em rede neural para gerar sinais de controle para o VANT simulado, geração de gráficos de comparação de diversas arquitetura de rede neural com o objetivo de determinar a arquitetura que torne os procedimentos de decolagem, deslocamento e pouso mais condizente com um piloto humano.

1.5 Estrutura da dissertação

No capítulo 1 desta dissertação apresentamos o embasamento teórico. Demonstramos a modelagem de um VANT e a representação de espaço de estados do mesmo assim como a dinâmica do modelo de corpo rígido do mesmo. Ainda neste capítulo fazemos referência à teoria das redes neurais artificiais e os modelos de aprendizagem.

O estado da arte, disposto no capítulo 2, baseado em um conjunto de artigos, nos provê o embasamento necessário para direcionar o foco deste trabalho na criação de um controlador neural sem a necessidade da modelagem do sistema. Também apresentamos

um comparativo com tais trabalhos.

O capítulo 3 relaciona o conjunto de problemas derivados da busca para a construção do controle neural, a construção da arquitetura da rede neural e as soluções encontradas para tais problemas.

A implementação dos mecanismos que possibilitaram os objetivos deste trabalho fossem alcançados está disposto no capítulo 4. O detalhamento da arquitetura utilizada para a integração do módulo de coleta de dados e o módulo de controle para com o simulador (Flightgear) consta neste capítulo.

Os dados gerados pelos experimentos dos módulo de coleta de dados, módulo de controle e a interação com o simulador, foram analisada de forma criteriosa para atender aos objetivos de eficiência e estabilidade. Esta análise de dados encontra-se no capítulo 5.

Capítulo 2

Embasamento Teórico

2.1 Controles de vôo

Os sistemas de controle de vôo de uma aeronave baseiam-se nos controles não-lineares dos 6 graus de liberdade (*Degree of Freedom* - DOF) e podem envolver diversos níveis de modelagem [Kinoshita & Imado 2006] [Jin et al. 2009]. Os graus de liberdade são os deslocamentos e as rotações nos eixos x, y e z conforme visto na figura 2.1.

O movimento angular de um VANT em termos aeronáuticos, nos eixos *xyz*, são denominados de rolagem, arfagem e guinada [Anton & Busby 2006].

Pela teoria de controle moderna é necessário um processo de modelagem matemática que condense as características de um objeto em um conjunto de equações com o propósito de determinar de maneira precisa o estado do sistema a qualquer tempo e assim efetivamente gerar os sinais de controles adequados. Uma abordagem é utilizar a representação de espaço de estados. Esta representação se faz necessária devido à complexidade crescente dos sistemas de controle atuais que tem a necessidade de abranger as crescentes e rigorosas exigências de desempenho de controle, aumentado assim a complexidade dos sistemas que geralmente contém múltiplas entradas e saídas e é variante no tempo [Ogata 2010].

Para a modelagem no espaço de estados são necessário estruturas que modelem matematicamente o objeto: Variáveis de estado, que constituem o menor conjunto de variáveis capaz de determinar o estado de um sistema dinâmico; vetor de estado, que agrupa as variáveis de estado para determinar univocamente o estado de um sistema para qualquer

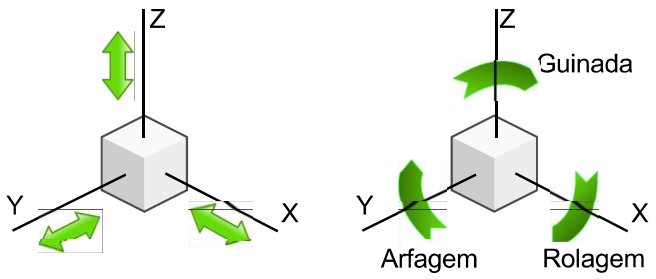

Figura 2.1: Graus de liberdade: Deslocamento e Rotação.

instante de tempo; equação de espaço de estados que agrupa em um conjunto de equações e que caracteriza completamente um sistema com suas variáveis de entrada, as variáveis de saída e as variáveis de estado [Ogata 2010].

Variável	Definição
x	Coordenada norte no *frame* inercial
y	Coordenada leste no *frame* inercial
h	Altitude no *frame* inercial
u	Velocidade do *frame* do corpo medido ao longo de x_b
v	Velocidade do *frame* do corpo medido ao longo de y_b
w	Velocidade do *frame* do corpo medido ao longo de z_b
Φ	Ângulo de rolagem no *frame* inercial
Θ	Ângulo de arfagem no *frame* inercial
Ψ	Ângulo de guinada no *frame* inercial
p	Razão de rolagem no *frame* do corpo no tempo
q	Razão de arfagem no *frame* do corpo no tempo
r	Razão de guinada no *frame* do corpo no tempo
α	Ângulo de ataque
β	Ângulo de deslocamento lateral
V	Velocidade

Tabela 2.1: Variáveis de Estado (modelo de corpo rígido)

Como exemplo de modelagem, temos o estudo de um controle de voo ótimo para um VANT autônomo [Kinoshita & Imado 2006]. Inicialmente foi definido que algumas características do modelo de corpo rígido são necessárias. Equações de movimento são usadas para simular a dinâmica do modelo de corpo rígido. As variáveis de estado estão definidas na tabela 2.1. As forças aerodinâmicas são as funções de ângulo de ataque e ângulo de deslocamento lateral denotados por α e β respectivamente. O vetor de velocidade inercial pode ser representado por *u, v* e *w* ou α, β e a velocidade *V*, como sendo a resultante final do movimento. A relação entre $(V,\alpha,\beta)^T$ e $(u,v,w)^T$ é dador por:

$$(u \;\; v \;\; w)^T = (V cos\alpha cos\beta \;\; V sin\beta \;\; V sin\alpha cos\beta) \tag{2.1}$$

$$\begin{pmatrix} V \\ \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \sqrt{u^2+v^2+w^2} \\ tan^{-1}(w/u) \\ tan^{-1}\left(v/\sqrt{u^2+v^2}\right) \end{pmatrix} \tag{2.2}$$

Se $V=(u,v,w)^T$ e $\omega=(p,q,r)^T$ as equações de movimento podem ser escritas como:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{h} \end{pmatrix} = \begin{pmatrix} c\Theta c\Psi & s\Phi s\Theta c\Psi - c\Phi s\Psi & c\Phi s\Theta c\Psi + s\Phi s\Psi \\ c\Theta s\Psi & s\Phi s\Theta s\Psi + c\Phi c\Psi & c\Phi s\Theta s\Psi - s\Phi c\Psi \\ -s\Theta & s\Phi c\Theta & c\Phi c\Theta \end{pmatrix} \times \begin{pmatrix} u \\ v \\ -w \end{pmatrix} \tag{2.3}$$

$$\dot{V} = (-\omega \times V + F)/m \tag{2.4}$$

$$\begin{pmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{pmatrix} = \begin{pmatrix} 1 & s\Phi t\Theta & c\Phi t\Theta \\ 0 & c\Phi & -s\Phi \\ 0 & s\Phi s\Theta & c\Phi s\Theta \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} \tag{2.5}$$

$$\dot{\omega} = I^{-1}(-\omega \times I\omega + G) \tag{2.6}$$

Assumindo que as forças e torques trabalham em um VANT são primeiramente de 3 origens.

$$F = F_g + F_a + F_t \tag{2.7}$$

$$G = G_a \tag{2.8}$$

Onde F_g é a força gravitacional, F_a a força aerodinâmica, F_t o empuxo e G_a o torque aerodinâmico que são dados respectivamente por:

$$F_g = \begin{pmatrix} -mgsin\Theta \\ mgcos\Theta sin\Phi \\ mgcos\Theta cos\Phi \end{pmatrix} \tag{2.9}$$

$$F_a = 1/2\rho V^2 S(C_{Xt} \;\; C_{Yt} \;\; C_{Zt})^T \tag{2.10}$$

$$F_t = (T, 0, 0)^T \tag{2.11}$$

$$G_a = 1/2\rho V^2 S(b \times C_{Lt} \;\; \bar{c} \times C_{Mt} \;\; b \times C_{Nt})^T \tag{2.12}$$

Onde ρ é a densidade do ar, $C_{Xt}, C_{Yt}, C_{Zt}, C_{Lt}, CMt, C_{Nt}$ são os coeficientes aerodi-

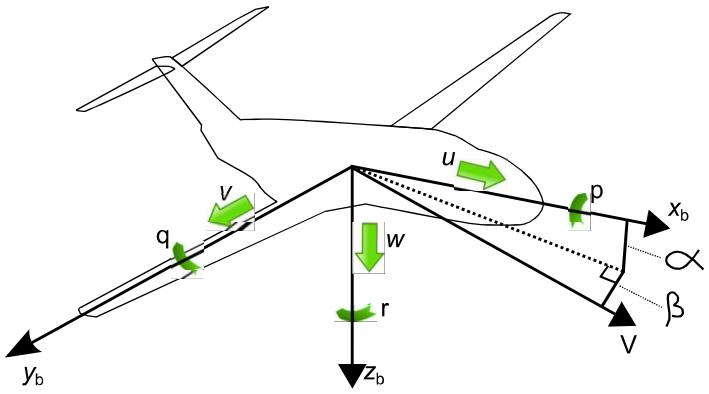

Figura 2.2: Modelo de corpo rígido.

nâmicos, *m* a massa do VANT e *g* a constante gravitacional. Desta forma Kinoshita [Kinoshita & Imado 2006] realizou a modelagem matemática para um modelo de corpo rígido [Kinoshita & Imado 2006].

É muito difícil obter um controle ótimo de 6 graus de liberdade do modelo de corpo rígido pela resolução de problemas de valores limitantes de dois pontos por causa das 4 variáveis de controle (defletor do estabilizador horizontal, defletor do *aileron*, defletor do *rudder* e empuxo total do motor). Para obter sucesso na construção de um controle ótimo, utiliza-se 3 variáveis ou menos. Um controle ótimo caracteriza-se pela obtenção de uma lei de controle para minimizar ou maximizar um funcional. Para tanto é necessário a utilização de um modelo de ponto de massa com 3 variáveis de controle (ângulo de ataque, ângulo de torção e empuxo total do motor). Este modelo também foi desenvolvido em [Kinoshita & Imado 2006].

Semelhante a outros modelos, se a sequência da parametrização não for altera, a modelagem matemática se faz efetiva e válida para estes modelos onde a parametrização não foi alterada. Um conjunto de controles, poderiam ser derivados para proceder as correções necessárias na planta observada e assim, gerar efetivamente as ações/controles necessárias. Se ocorrer a mudança na parametrização será necessário uma nova modelagem matemática. Se existir um conjunto de *n* plantas idênticas a serem controladas e cada uma destas planta apresentar uma parametrização diferente, o processo de modelagem deverá ser refeito para cada uma das *n* plantas.

2.2 Redes neurais artificiais

Uma rede neural artificial (RNA) é um modelo computacional que possui inspiração no sistema nervoso dos organismos vivos. As unidades de processamento de uma RNA são os neurônios artificiais que tem a capacidade de adquirir e reter a informação. Estas unidades estão interconectadas entre si simulando artificialmente as sinapse existentes em organismos vivos.

Algumas características das RNA são:

- Adaptação por experiência ajustando os pesos sinápticos (parâmetros internos) da rede, usando exemplos para gerar o aprendizado. Desta forma, a RNA pode adquirir conhecimento atravéz de um conjunto de exemplos de treinamento.
- Generalização do conhecimento adquirido visando a estimação de novas soluções.
- Aprendizado proveniente do conjunto de treinamento, estabelecendo assim relações entre as variáveis.

A aplicabilidade das RNA estende-se por diversos campos relacionados às engenharias e ciências com uma grande abrangência devido as características de:

- Aproximador universal de funções que faz o mapeamento do relacionamento funcional entre as variáveis do sistema com origem dos dados
- Controle de processos com o objetivo de identificar as ações que controlam a planta com a meta de qualidade, eficiência e estabilidade.
- Sistemas de previsão que visam estimar valores futuros com base nas observações prévias da entrada de dados.
- Otimização de sistemas que consistem em minimizar/maximizar uma função custo em um problema com restrições.

De acordo com Silva:

"Os neurônios artificiais utilizados nos modelos de RNA são não-lineares, fornecem saídas tipicamente contínuas, (...) coletar sinais existentes em suas entradas, agregá-los de acordo com sua função operacional e produzir uma resposta, levando em consideração sua função de ativação inerente"[Silva et al. 2010].

O arranjo dos neurônios artificiais em uma estrutura é comumente chamada de arquitetura. A composição estrutural de uma arquitetura, é definida como topologia. Podemos ter duas topologias pertencentes a uma mesma arquitetura, uma com função de ativação logística para 20 neurônios artificiais e outra com função de ativação tangente hiperbólica para outros 10 neurônios artificiais. Geralmente classificam-se as RNA quanto à arquitetura em três classes [Haykin 2001]:

- *Redes alimentas adiante com camada única*. Consiste da forma mais simples de uma RNA em camadas. Existe uma camada de entrada que esta totalmente conectada à camada de saída. Esta rede é estritamente do tipo *alimentada adiante* ou *acíclica*. Recebe a designação de rede de *camada única* (figura 2.3).
- *Redes alimentas adiante com múltiplas camadas*. Adicionalmente às camadas de entrada e saída, possui um conjunto denominado de *oculto* que pode possuir uma ou mais camadas com neurônios artificiais. Na figura 2.4 existem as camadas de entrada, saída e oculta. A camada oculta possui uma única camada com 3 neurônios, a camada de saída possui 2 neurônios e a camada de entrada possui 5 nós de entrada.
- *Redes recorrentes*. Caracterizam-se por possuir um laço de realimentação, podendo possuir camadas ocultas. A figura 2.5 nos mostra uma rede recorrente sem camada de neurônios ocultos. Os elementos z^{-1} são denominados de *elementos de atraso unitário* e tem como função fornecer uma realimentação para uma nova entrada na RNA.

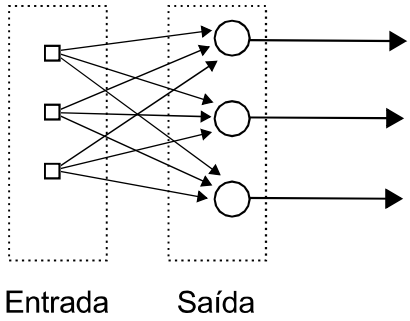

Figura 2.3: Arquitetura de RNA - Rede alimentada adiante.

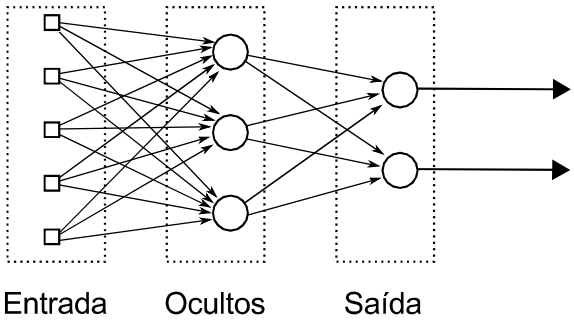

Figura 2.4: Arquitetura de RNA - Rede alimentada adiante com camada oculta.

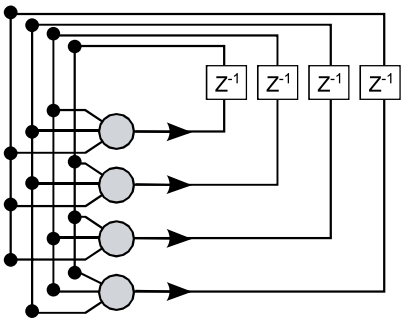

Figura 2.5: Arquitetura de RNA - Rede recorrente.

Uma RNA possui a camada de entrada, camadas escondidas e a camada de saída. A camada de entrada tem função de proceder o recebimento de informações do ambiente. Esta informações podem ser dados, sinais ou medições. A camada escondida tem a responsabilidade de extrair as características associadas ao processo ou sistema a ser inferido [Silva et al. 2010]. A camada de saída é responsável pela apresentação dos resultados finais da rede proveniente das camadas anteriores. Os cálculos realizados por toda a rede neural são extraídos e apresentados na saída.

Um dos fatos mais relevantes em uma RNA se faz presente na capacidade de aprendizagem decorrente do conjunto de treinamento. Uma vez que a RNA tenha aprendido o relacionamento dos dados de entrada com os dados de saída, será possível generalizar e apresentar saídas (mesmo que aproximadas) para novos sinais apresentados em sua entrada. O treinamento de uma RNA é uma tarefa de ajuste de pesos sinápticos com finalidade de adequar a estrutura da rede de acordo com as entradas da base de treinamento e gerar as saídas adequadas. A maneira com que os neurônios artificiais de uma RNA estão estruturados esta intimamente relacionado com o algoritmo de aprendizagem utilizado para treiná-la [Haykin 2001].

Na abordagem da aprendizagem supervisionada temos um conjunto de sinais de entrada e as respectivas saídas. Estes dados de treinamento que fazem o mapeamento de entrada e saída tem que ser conhecidos *a priori*. Uma amostra de treinamento é formada pelos sinais de entrada e saída. Os pesos sinápticos e limiares são ajustados pela utilização do algoritmo de aprendizagem que realiza ajustes utilizando o erro gerado pela diferença do valor desejado e do valor calculado pela RNA.

Com a apresentação dos conjunto de treinamento, a RNA aprende e começa a gerar erros de saída cada vez menores. Quando estes erros estiverem dentro de uma faixa especificada ou aceitável, a rede será considerada treinada. A primeira estratégia de treinamento supervisionado foi proposta em 1949 por Donald Hebb, cuja inspiração é proveniente da neurofisiologia.

Na aprendizagem supervisionada, a imposição do professor envolve substituir a saída real de um neurônio, durante o treinamento da rede, pela resposta correspondente na computação subsequente do comportamento dinâmico da rede. Embora a imposição do professor seja descrita para o algoritmo ARTR (*Aprendizagem recorrente em tempo real*), o seu uso aplica-se a qualquer outro algoritmo de aprendizagem [Haykin 2001].

A unidade básica da RNA é o *perceptron* idealizado por Rosenblatt em 1958. O diagrama da figura 2.6 nos mostra o *modelo* de um neurônio. Os três elementos básicos de um neurônio artificial para um projeto de RNA são:

- O conjunto de *sinapses* ou *elos de conexão* que, individualmente, possuem um *peso* que são denotado por $w_{k1}, w_{k2}, ..., w_{km}$. Os pesos sinápticos irão ponderar os valores da entrada $(x_1, x_2, ..., x_m)$ para a junção aditiva.
- O somador agrega os valores de cada peso sináptico com sua respectiva entrada e é denominado de *combinador linear*.
- *A função de ativação* limita a amplitude do sinal proveniente do *combinador linear* e ativa ou não a saída do neurônio.

Além do conjunto sináptico da figura 2.6 inclui-se um *bias* que tem um valor fixo e é representado por b_k.

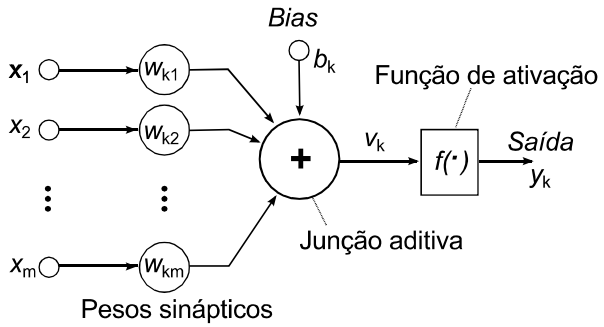

Figura 2.6: Modelo de um neurônio não-linear.

Matematicamente um neurônio k é escrito por um par de equações[Haykin 2001]:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.13}$$

e

$$y_k = \varphi(\overbrace{u_k + b_k}^{v_k}) \tag{2.14}$$

Onde $x_1, x_2, ..., x_m$ são as entradas, $w_1, w_2, ..., w_{km}$ são os pesos sinápticos do neurônio k, u_k é a *saída do combinador linear*, b_k é o bias, $\varphi(\cdot)$ é a *função de ativação* e y_k o sinal de saída do neurônio. v_k é o *campo local induzido* ou *potencial de ativação* v_k.

A saída y_k obtém o resultado diretamente da função de ativação. A saída da função de ativação gera o valor y_k. Podemos generalizar e definir 3 tipos básicos de funções de ativação.

- *Função de limiar*. Esta função (equação 2.15) gerará a saída do neurônio de acordo com a equação 2.16. Onde v_k é o campo local induzido do neurônio e é definido em 2.17.

$$\varphi(v) = \begin{cases} 1, & \text{se} \quad v \geq 0 \\ 0, & \text{se} \quad v < 0 \end{cases} \tag{2.15}$$

$$y_k = \begin{cases} 1, & \text{se} \quad v_k \geq 0 \\ 0, & \text{se} \quad v_k < 0 \end{cases} \tag{2.16}$$

$$v_k = \sum_{j=1}^{m} w_{kj} x_j + b_k \tag{2.17}$$

- *Função linear por partes*. Função descrita em 2.18. Esta função tem como característica uma *aproximação* de um amplificador não-linear [Haykin 2001].

$$\varphi(v) = \begin{cases} 1, & v \geq +\frac{1}{2} \\ v, & +\frac{1}{2} > v > -\frac{1}{2} \\ 0, & v \leq -\frac{1}{2} \end{cases} \tag{2.18}$$

- *Função sigmóide*. É a função mais utilizada para uma função de ativação em uma RNA. Um exemplo de função sigmóide é a função logística (equação 2.19), onde a é o parâmetro de inclinação da função sigmóide. No limite, quando a aproxima-se do infinito, a função torna-se uma função limiar.

$$\varphi(v) = \frac{1}{1+\exp(-av)} \tag{2.19}$$

A aprendizagem de uma RNA é o processo de aprender a partir do ambiente e melhorar seu desempenho [Haykin 2001]. Matematicamente é o ajuste do peso sináptico de cada *perceptron*. Este ajuste é realizado comparando-se a saída do neurônio com o valor desejado. Uma diferença é calculada e o peso sináptico é ajustado a cada interação. Se não existir diferença não é realizado ajuste e a saída do neurônio gerou uma informação adequada, usando os dados de entrada.

Como exemplo temos uma RNA que tem como finalidade classificar pontos no sistema cartesiano 2D em classe 1 ou classe 2 conforme a figura 2.7-a. Se um ponto azul for classificado pela RNA como sendo da classe 1, a rede acertou a classificação e não necessita de ajuste nos pesos sinápticos. Entretanto se um ponto azul foi classificado como classe 2, a rede errou na classificação e necessita de ajustes nos pesos sinápticos. Na figura 2.7-b existem 4 classes de pontos e duas retas separando o plano em 4 áreas. Cada uma destas retas gera uma fronteira de separação e é usada matematicamente para definir a que classe cada ponto pertence. Neste exemplo, as 4 classes são *linearmente separáveis* pois existe uma reta capaz de separar as classes. Um *perceptron* que possa ser utilizado como um classificador de padrões necessita que as classes sejam linearmente separáveis.

Em um estágio intermediário de treinamento da RNA, o ajuste dos pesos sinápticos, matematicamente é refletido no ajuste do ponto de separação de cada classe, aqui representado por uma reta (figura 2.7-a). Cada ajuste dos pesos sinápticos tenta ajustar a inclinação da reta para gerar uma reta que possa separar a classe 1 da classe 2. Na figura 2.7-c, a RNA poderia gerar a reta 1. Um novo exemplo de treinamento é dado e existe um erro na classificação, um novo ajuste é realizado e a RNA poderia gerar a reta 6. Consecutivamente a RNA irá gerá erros e acertos com o decorrer dos exemplos de treinamento. A inclinação da reta é dada pelo ajuste dos pesos sinápticos. Normalmente a reta poderá variar da reta 1 até reta 6. Quando a rede estiver com erros mínimos ou aceitáveis, a reta 3 será encontrada como melhor reta delimitadora e a mesma será a fronteira para separação das classificações futuras entre a classe 1 e a classe 2.

Quando um problema de classificação não pode encontrar uma reta que determine uma fronteira de separação entre classe, outros métodos são utilizados para realizar a

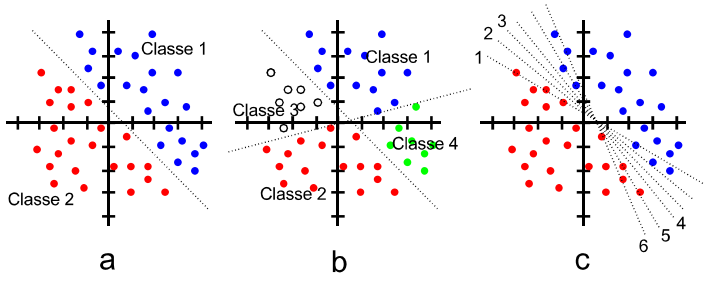

Figura 2.7: Classificação de pontos.

classificação.

Uma RNA de múltiplas camadas é comumente chamada de *Perceptron de Múltiplas Camadas* (PMC) conforme a figura 2.4.

O PMC é utilizado, também, como aproximador funcional, que consiste em mapear o comportamento de um processo baseando-se nas medições efetivas em suas entradas e saídas [Silva et al. 2010].

Devido a capacidade de gerar um conjunto de resultado proveniente de exemplos, as redes PMC torna-se uma alternativa viável em ambientes em que os únicos dados do ambiente são os dados de entrada e saída. Constata-se que as RNA têm sido extensivamente aplicadas em situações em que o processo a ser modelado é de certa forma complexo, nas quais as utilizações de métodos convencionais produzem resultados insatisfatórios ou então, naquelas situações em que o sistema já modelado se torna demasiadamente particularizado em torno de alguns pontos de operações que produzem soluções satisfatórias [Haykin 2001].

A topologia das camadas escondidas de um PMC é capaz de mapear qualquer função contínua. Em termos matemáticos tempos:

$$y(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{n_1} \overbrace{\lambda^i}^{parcela(i)} \cdot \overbrace{g_i^{(1)}(u_1^{(1)})}^{parcela(ii)} \tag{2.20}$$

$$u_i^{(1)} = \sum_{j=1}^{n} W_{ij}^{(1)} . \; x_j - \theta_i \tag{2.21}$$

As equações 2.20 e 2.21 podem gerar uma função contínua no espaço de funções reais se a função de ativação for contínua. As funções de ativação logística e tangente hiperbólica são utilizadas para gerar a saída y, na saída do PMC, utilizando uma superposição das funções de ativação. Utilizando-se de funções de ativação logística, podemos gerar uma função de aproximação para o conjunto de amostras de entrada conforme a figura 2.8.

A proposta é de gerar uma função (ou sua aproximação) tendo um conjunto de dados

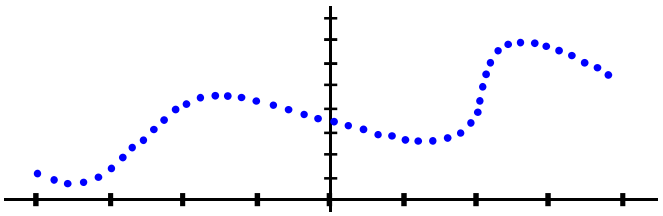

Figura 2.8: Função de aproximação - amostras de treinamento.

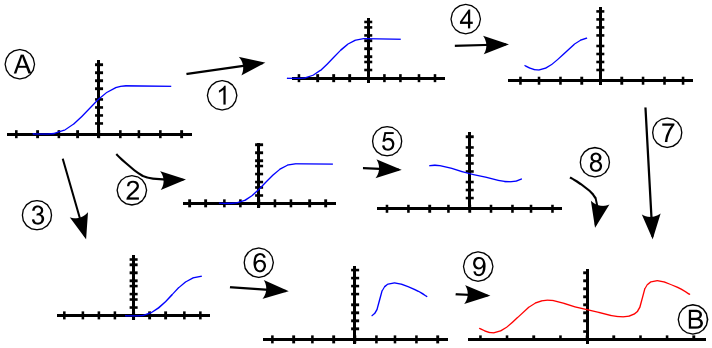

Figura 2.9: Composição de uma função de aproximação.

de entrada. Os neurônios artificiais das camadas escondidas do PMC terão seus pesos sinápticos ajustados a fim de gerar a função de aproximação. Cada neurônio artificial, juntamente com o seu peso sináptico, ajudará a descrever matematicamente uma parte da função de aproximação.

Baseando-se na figura 2.8, sendo o conjunto de pontos definidos como entrada no PMC, devemos encontrar uma função que realize o mapeamento do comportamento funcional do processo. A base para esta aproximação é a composição de uma função que agrupe parcelas de funções logísticas. O PMC, usando os dados referentes à figura 2.8 como dados de entrada deverá apresentar na saída da RNA uma função semelhante à 2.10 através da composição de funções demonstrado na figura 2.9.

Na figura 2.9 toda a base para a composição inicia-se com a função logística *A*. Alterações nas funções logística irão gerar as funções necessárias para gerar partes do funcional de aproximação. As alterações são realizado nos ajustes dos pesos sinápticos λ e na translação da função θ_i (Equações 2.21 e 2.20). Na figura 2.9 os neurônios aplicam um $\theta = 0, \theta = -6,5$ e $\theta = 5$ respectivamente em *1, 2* e *3*. Cada uma destas saídas sofre influência de um peso sináptico de valores iguais à $\lambda = -0,2, \lambda = 0,5$ e $\lambda = 0,6$ respectivamente em *4, 5* e *6*. Fazendo a composição de *7, 8* e *9* teremos a função aproximada *B*. A composição é realizada dentro do domínio dos neurônios da camada oculta.

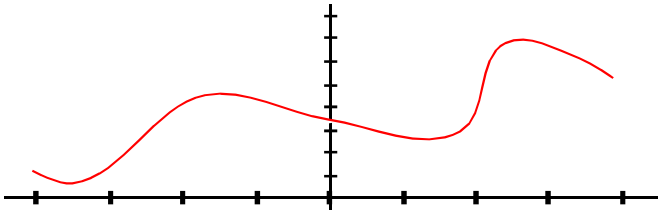

Figura 2.10: Composição final da função de aproximação.

Capítulo 3

Estado da Arte

3.1 Trabalhos relacionados

A compensação para incertezas em um VANT usando uma RN é detalhado em [Eom & Jung 2007] e nos relata que a performance de um controle de posicionamento do tipo PD é degradado quando distúrbios externos estão presentes. Uma rede neural foi utilizada para compensar estes distúrbios.

O trabalho cooperativo entre dois helicópteros no céu carregando objetos para construção de uma ponte é um trabalho que os humanos não podem realizar tão facilmente. A presença de um ambiente dinâmico torna necessário um sofisticado controle de força para o trabalho em cooperação. A adição de um controle de força para o sistema de controle de posicionamento auxilia com grande eficácia a estabilidade quando o helicóptero opera em um ambiente dinâmico. A velocidade do helicóptero é controlada pelo método LQR (*Linear Quadratic Regulator*) e a posição é controlada pelo ganho proporcional na forma de um controle de posição do tipo PD. Adicionalmente uma RN é adicionada para compensar as incertezas provenientes de distúrbios externos. Distúrbios externos forçam os controles a compensar incertezas. A compensação destas incertezas é realizada adicionando junto à estrutura de controle uma RN [Eom & Jung 2007].

Em *Neural Network Compesation for Force Tracking Control of an Autonomous Helicopter System* [Eom & Jung 2007], os resultados da simulação que faz o helicóptero desloca-se até um ponto x_e do ambiente usando o controle de força de impedância adaptativo com compensação de uma rede neural demonstra que os erros de trajetória são consideravelmente minimizados pois a rede neural compensa as incertezas e desta forma atua melhorando o controle de trajetória. Semelhante simulação é realizada porém variando-se o distúrbio externo aplicado ao sistema. Novamente o controle com rede neural obteve um melhor ajuste de trajetória. Conclui-se que a rede neural rejeita os distúrbios e assim melhorando a performance da trajetória.

O uso de um ajudante baseado em uma rede neural é foco do trabalho de Krishna-Kumar [Krishnakumar & Sawhney 1991] e cita que o treinamento de um piloto humano novato em um helicóptero é um processo de grande risco e que consome bastante tempo. Usa-se simuladores de vôo para treinar as percepções cognitivas e psico-motoras de um aluno até que o mesmo tenha capacidades básicas de coordenar diferentes controles, reconhecer a dinâmica da aeronave e a dinâmica dos controles utilizando as percepções visuais, de movimento e auditivas. A tarefa complexa de gerenciar todos os sistemas com

uma estabilidade marginal de um helicóptero é minimizada com a utilização de um instrutor. O instrutor, mesmo em um simulador, realizará as compensações adequadas para cada momento do vôo, a fim de tornar a tarefa (pouso, decolagem ou deslocamento) mais estável possível, até que o aluno tenha a capacidade de realizar estas compensações de maneira autônoma e correta. Geralmente o instrutor ajuda o aluno em certas condições de controle com o objetivo de não perder o controle da aeronave. Matematicamente, o aluno realiza um conjunto de ajustes nos controles disponíveis a fim de realizar as compensações necessárias com o intuito de manter a aeronave mais estável possível. Intrisicamente o aluno calcula o erro e realiza os ajustes necessários a fim de compensar a dinâmica da aeronave de acordo com o ambiente em operação. Caso o aluno não realize as compensações a fim de promover a estabilidade da aeronave, o instrutor intervém e faz as compensações. O instrutor tem uma capacidade, devido ao treinamento, de calcular melhor o erro e assim ser mais eficiente nas compensações e alcança uma estabilidade mais rapidamente.

Um instrutor ajuda o aluno nas compensações necessárias durante uma tarefa específica. Um ajudante virtual baseado em RN realizaria as compensações necessárias com o diferencial de ajuste de níveis de ajuda. Os níveis de ajuda tem como objetivo auxiliar o aluno a desenvolver progressivamente as capacidades de controle. Quanto maior o nível do aluno, menos será necessária a compensação provida pelo ajudante virtual. O treino progressivo ajuda o aluno a entender e adaptar-se às dinâmicas do helicóptero com um baixo nível de risco.

O desenho do treino adaptativo utilizando um neurocontrolador em [Krishnakumar & Sawhney 1991] foi dividido em seis partes.

- Definição do sistema de pairagem.
- Definição da tarefa de pairagem.
- Modelo neural para monitorar a performance do aluno na tarefa de pairagem.
- Neuromodelagem do sistema de pairagem do helicóptero.
- Desenho do neurocontrolador.
- Desenho de uma estrutura de RN para adaptar a saída do neurocontrolador para a performance do aluno.

A rede neural e o modelo do helicóptero estão alimentando o vetor de estado atual e o controle de entrada. A tarefa da rede neural é predizer precisamente o próximo estado do sistema. Este é o problema de identificação primário. Duas variações de um sistema de *backpropagation* foram implementados, o primeiro método envolve em dar para a rede o vetor de estado correto como entrada e esperar que a rede produza no próximo estado. No segundo é dado somente o vetor de estado inicial e realimentado a entrada com a saída. Esta versão gera um modelo mais robusto do sistema. A rede neural faz o papel de um ajudante para o estudante que adapta-se à curva de aprendizagem do estudante. A rede neural realiza as tarefas de monitorar a performance do estudante e adaptação de um sistema do helicóptero usando um neurocontrolador adaptativo.

A natureza não-linear dos VANT's do tipo helicóptero descrito em [Chen et al. 2007] nos relata que a dinâmica é essencialmente instável e tais características não-lineares variam com a velocidade do ar e altitude. Tais sistemas são do tipo MIMO (*multiple input,*

multiple output) e diversos controles podem ser propostos. **Controles clássicos do tipo lógica *fuzzy* podem ser utilizados e facilmente construídos pois não necessitam do modelo do sistema**. Os testes de performance de rastreio de um conjunto de desenhos de controles de vôo do helicóptero usando controle multivariado robusto linear, em [Shim et al. 1998], com controle lógico *fuzzy* com ajuste evolucionário e controle de rastreio não-linear, chegaram à conclusão que os controles robustos e *fuzzy* são capazes de manipular incertezas e distúrbios. Uma RN pode ser usada não somente como um controle inteligente o qual tem capacidade robusta e adaptativa mas também tem uma estrutura simples. Os resultados da simulação mostram que as saídas do sistema podem responder aos sinais de entrada precisamente em condições de vôo de planagem.

Ainda em [Chen et al. 2007] afirma que o helicóptero é um sistema complexo MIMO com correlação altamente cruzada acoplada. A modelagem deste sistema é semelhante ao modelo previamente modelado na seção 2.1, página 5. Os resultados da simulação em [Chen et al. 2007] relatam que para um controlador PID baseado em rede neural, o ajuste de parâmetros é um processo importante e tedioso e porque os valores iniciais dos parâmetros podem afetar a convergência da resposta do sistema. O procedimento de ajuste é dificultado devido a acoplagem entre as variáveis de estado especialmente para os parâmetros K_1 e K_2 (parâmetros de ganho da RN).

Conforme é relatado em [Buskey et al. 2001], o sistema de controle adaptativo de um VANT que tem como objetivo alcançar um estado de pairar na maioria das condições. Tentativas de usar métodos de controle tradicionais para a construção de leis de controle inversa para a planta de helicópteros podem, inevitavelmente, encontrar problemas quando encarados com a variância temporal e natureza não-linear destes sistemas. Com isto, para diferentes modelos de vôo, a dinâmica, portanto, assim deve ser o modelo de aproximação linear [Mettler et al. 2000]. A necessidade de adaptação, juntamente com a necessidade de ser capaz de lidar com estas dinâmicas não ideais, tornam este problema particularmente adequado para redes neurais e/ou controladores de lógica *fuzzy* [Mettler et al. 2000]. O conjunto de entradas e saídas são usados para treinar uma RN em exemplos de respostas viáveis para uma variedade de situações de pairagem. No tocante à arquitetura da rede, uma grande quantidade de unidades escondidas permite melhores valores de convergência entre as saídas da rede e as saídas-alvo durante o treinamento. Contudo, um número pequeno de unidades escondidas resulta em uma generalização melhor quando exposta a situações não apresentadas durante os testes. O treinamento satisfatório é considerado como sendo atribuíveis à qualidade do conjunto de treino.

Um problema comum no desenho do controle de vôo de alta performance para um VANT é obter o modelo de dinâmica com alta fidelidade[Taha et al. 2010]. Métodos de controle de inteligência artificial, como redes neurais, lógica *fuzzy* e controles *fuzzy* neurais tem possibilidades de alta performance sem alta sobrecarga computacional para identificar e controlar sistemas dinâmicos não-lineares [Taha et al. 2010]. A identificação da *black box* sem a integração do modelo físico é considerado como uma ferramenta crítica a qual é mais usada para sistemas de controle inteligente.

O processo de treinamento usado para aproximar a planta do modelo dinâmico do helicóptero foi uma RN com *backpropagation* como algoritmo de aprendizagem com a função de aproximação Levenberg-Marquardt, usado devido a sua velocidade de conver-

gência [Taha et al. 2010] em comparação ao erro médio quadrático (MSE). Chegou-se à conclusão que o uso de uma rede *nonlinear autoregressive with exogenous inputs series-parallel* (NARXSP) como um aproximador de identificação de um sistema é válido e satisfatório e a identificação do modelo **não necessita** do cálculo dos parâmetros do modelo físico.

3.2 Tabela comparativa

Característica	Autor A	Autor B	Autor C	Autor D	Autor E	Nosso
Modelo Matemático	Sim	Sim	Sim	Sim	Não	Não
Complexidade Computacional	Baixo	Baixo	Baixo	Médio	Médio	Alto
Rigidez do Modelo	Alto	Alto	Alto	Alto	Alta	Baixo
Portabilidade entre Modelos	Não	Não	Não	Não	Não	Sim
Generalização	Não	Não	Não	Não	Sim	Sim
Compensação de Distúrbios	Médio	Baixo	Baixo	Baixo	Alto	Alto
Variações Atmosféricas	Não	Não	Não	Não	Não	Sim
Reproduzibilidade dos Experimentos	Não	Não	Não	Não	Não	Sim
Treinamento Único	Não	Não	Não	Não	Não	Sim
Ajustes de PID	Sim	Sim	Sim	Não	Não	Não
Complexidade de Construção	Alta	Alta	Alta	Alta	Média	Baixa

Tabela 3.1: Tabela comparativa

Sendo autor A [Eom & Jung 2007], autor B [Krishnakumar & Sawhney 1991], autor c [Chen et al. 2007], autor D [Buskey et al. 2001] e autor E [Taha et al. 2010].

Características analisadas para a comparação dos trabalhos.

- *Modelo matemático*. É necessário o desenvolvimento de um modelo matemático (funções de espaço de estados) para a realização dos experimentos.
- *Complexidade computacional*. Capacidade computacional envolvida na tarefa para resolução do problema, cálculos envolvidos ou treinamento da rede neural. Considera-se baixo, o baixo uso de cálculos (sistemas PID) e alto, uma quantidade considerável de cálculos (treino de uma RN)
- *Rigidez do modelo*. Relacionado diretamente com as funções de espaço de estado que caracterizam um modelo. A alteração nos parâmetros de um modelo, obrigatoriamente, necessita de novas funções de espaço de estado. Alta rigidez caracteriza-se por estruturação rígida no espaço de estados para um modelo específico.
- *Portabilidade entre modelos*. Igualmente relacionada com as funções de espaço de estado, uma modelagem poderia ser portada para diversos modelos de VANTs com poucas alterações. A rigidez do modelo e a portabilidade estão intrisicamente atreladas pelo fato de que cada tipo/modelo de VANT, necessita de um modelo matemático para atender às necessidades de controle. Quase sempre este modelo não pode ser portado para tipos/modelos distintos de VANTs sem grandes alterações.

- *Generalização*. A generalização tem por finalidade a capacidade de adaptação do sistema para responder a situações não apresentadas durante a fase de treino ou configuração do sistema. Novas situações não apresentadas caracterizam-se pela falta de entradas apropriadas e suas respectivas saídas para uma determinada planta.
- *Compensação de distúrbios*. As entradas em um sistema de controle podem ter variações que são compensadas. Tais entradas podem apresentar ruído e variações devido a fatores externos (rajadas de vento, variações da densidade do ar, variações da umidade do ar).
- *Variações atmosféricas*. Ambientes que possam fazer ajustes na condições atmosféricas para testar o sistema de controle a fim de apresentar situações de difícil reprodução (senão impossível) na vida real (ex: VANT orientado na direção nordeste a uma velocidade de 60 Km/h, em uma altitude de 1.000 metros, recebendo rajadas de vento de forma estocástica de 15Km/h à 30Km/h, na direção sul-sudoeste descendente).
- *Reproduzibilidade de experimentos*. Capacidade de reproduzir um mesmo ambiente de teste para realizar experimentos com ajustes distintos no sistema de controle a fim de verificar o nível de adaptação e a reposta do sistema ao ambiente.
- *Treinamento único*. Uma vez que o sistema tenha sido ajustado não existe mais a necessidade de novos ajustes. Os ajustes atendem às demandas do controle em relação a distúrbios, portabilidade e rigidez do modelo. Geralmente, ambientes distintos necessitam de ajustes distintos para compensar as variações impostas no ambiente em que o VANT é inserido.
- *Ajustes de PID*. Somente as plantas que possuem o sistema de controle baseados totalmente ou parcialmente em PID necessitam de tais ajustes. O treino de uma RN provê a abstração dos ajustes tornando assim desnecessário nos controles baseados em RN.
- *Complexidade de construção*. A construção de um sistema de controle que compreenda a união de diversos sub-sistemas é complexa. Quantos menos sub-sistemas menos complexa é a contrução e por conseqüência menor a manutenção do mesmo.

3.3 Destaques do trabalho

Temos 04 (quatro) destaques para este trabalho.

1. Uso de um simulador com grande capacidade de customização e de código aberto. Neste trabalho foi utilizado o Flightgear (FG), um simulador que tem como ponto forte a capacidade de personalização de praticamente todas as suas partes. Novos módulos podem ser adicionados utilizando uma linguagem própria ou módulos adicionais podem ser escritos em quaisquer linguagens de programação. É definido um protocolo de comunicação entre o módulo e o Flightgear. Até mesmo este protocolo pode ser personalizado.
2. Utilização de um sistema de controle desenvolvido em C++, MATLAB (ou quaisquer linguagem de programação) permutáveis em tempo real, altamente customizado e robusto. A permutação entre sistemas desenvolvidos em linguagens de pro-

gramação diferentes, nos permite uma maior abrangência no tocante ao desenvolvimento em quaisquer linguagens de programação, assim o mecanismo de controle do VANT pode ser desenvolvido em quaisquer linguagem de programação. A permutabilidade em tempo real (podemos trocar em tempo real o sistema de controle desenvolvido em uma linguagem hipotética A por outra, desde que atenda às obrigações do protocolo do FlightGear) deve-se ao fato que o Flightgear recebe pacotes de dados pela pilha TCP/IP. Se um determinado módulo de controle estiver operacional, ele estará enviando pacote de dados pela pilha TCP/IP. Assim, basta suspender o envio destes pacotes e simultaneamente ativar o envio de pacotes de outro módulo. A única obrigatoriedade é que os controladores (módulos) respeitem o protocolo de comunicação previamente estabelecido. Ainda assim, existe a possibilidade de trabalhar com mais que um protocolo simultaneamente. A robustez caracteriza-se pela possibilidade de termos dois ou mais módulos de controles trabalhando simultaneamente. Em caso de problema ou não funcionamento de um dos módulos de controle, outro módulo assumiria o controle do VANT.

3. Utilização de diversos modelos de VANTS e reproduzibilidade de situações específicas para diversos modelos, diversos sistemas de controle e diversos ambientes de teste. O intercâmbio entre as diversas partes do experimento com baixo tempo de reconfiguração é devido à capacidade de personalização do Flightgear e ao desenvolvimento das atividades modulares. Assim, um determinado experimento pode ser repetido diversas vezes sob as mesmas condições. Podemos especificar uma gama extensa de parâmetros. Os parâmetros que geram o ambiente do simulador podem ser armazenados para reproduzir o mesmo ambiente posteriormente com outros VANTS, outros modelos de controle e quaisquer outros módulos.
4. Ambiente de testes automatizado com as características de armazenamento de dados de entrada, telemetria, dados de saída e repetibilidade de situações. A automatização dos procedimentos tem por objetivo que uma sequência de testes possam ser realizados e os dados (de entrada e saída) armazenados para uma posterior análise. Diversos experimentos podem ser enfileirados e o simulador realizará os testes. Os resultados de tais testes ficam armazenados para posterior análise. É possível especificar a quantidade de dados e quais dados de telemetria serão armazenados.

Capítulo 4

Problema e Solução

A criação do modelo matemático para um VANT torna-se difícil devido às características de ser não-linear, MIMO (Multiple Input Multiple Output) e uma forte acoplagem cruzada através de diferentes variáveis de estado [Hashimoto et al. 2000]. A função de espaço de estado caracteriza um modelo de forma matemática e o torna tratável. Uma vez que temos as funções de espaço de estado podemos gerar o sistema de controle de acordo com a reposta desejada, dentro dos parâmetros aceitáveis e com alta fidelidade em relação à dinâmica do modelo [Taha et al. 2010]. Com esta abordagem, temos como contrapartida a necessidade de para cada modelo de VANT, a criação de um novo conjunto de funções de espaço de estados. Isso deve-se ao fato de que cada VANT possui características físicas distintas, dentre elas o centro de gravidade, o volume e peso do modelo, a capacidade de carga, o torque do motor e diversas outras.

Os VANTs possuem um sistema dinâmico muito complexo com características MIMO e são altamente não-lineares. Métodos de controle baseados em rede neural tem possibilidade de identificar e controlar sistemas dinâmicos não-lineares desconhecidos [Al-Hinai & Ibnkahla 2008] [Kumon et al. 2006] [Canete et al. 2008] [Sebastien et al. 2001] [Ordonez & Passino 1999]. Com a capacidade de identificar tais modelos sem a necessidade da equação de espaço de estados, temos agora a tarefa de controlar o VANT.

Como controlar um VANT usando uma rede neural? O método utilizado para realizar o controle do VANT foi o treino de uma rede neural com um piloto real que realizou sob diversas circunstâncias a mesma tarefa diversas vezes. A telemetria das entradas do piloto real no VANT foram armazenadas assim como a resposta do VANT para as entradas do piloto real. Estes dados de telemetria foram colocados em uma rede neural para que a mesma pudesse ser treinada posteriormente. Verificou-se que seria necessário testar, treinar e avaliar diversos tipos de arquitetura para a rede neural devido aos testes de controle realizados. Arquiteturas distintas, com o mesmo conjunto de treinamento, geram respostas distintas que fazem o VANT comporta-se de forma distinta em um mesmo ambiente com mesmas características. Surge a necessidade de testar arquiteturas distintas para que encontre-se a mais adequada objetivando-se a eficácia e a estabilidade.

Com o conjunto de dados do treinamento, a rede neural *aprendeu* a controlar o VANT e realizar as tarefas de decolar, pairar e deslocar até um ponto específico de maneira estável e satisfatória (sem que ocorra um desvio do percurso ou queda do VANT).As verificações foram realizadas durante vôos utilizando a rede neural já treinada com o sistema

de controle neural em um ambiente simulado. Semelhante à coleta de dados durante a fase de treinamento com um piloto real, foi realizado também a coleta de dados da telemetria para diversas arquiteturas quando realizado uma das tarefa específicas (decolar, pairar ou deslocar-se).

Analisando uma das resposta da rede neural com diversas arquiteturas (figura 4.1), comparando com a resposta do piloto real e realizando vôos experimentais no simulador, encontramos uma boa arquitetura que atendesse as necessidades de eficácia e estabilidade. A comparação de diversas arquiteturas nos permite distinguir qual a que mais assemelha-se a um piloto real.

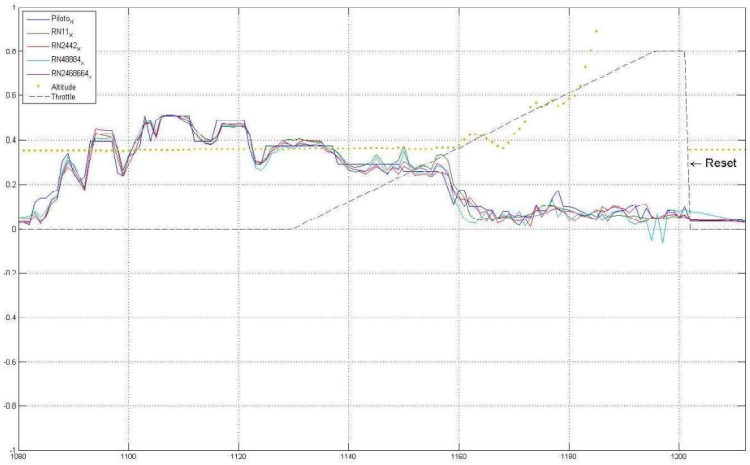

Figura 4.1: Comparação das respostas da rede neural. Eixo X.

4.1 Formulação matemática do aprendizado do controle de vôo

Nossa proposta é a criação de um sistema de controle sem a necessidade de uma formulação matemática. A caracterização matemática será abstraída pela rede neural.

O controle de vôo tem por finalidade apresentar uma resposta adequada para entradas específicas (a caracterização das relações de entrada e de saída dos componentes ou de sistemas podem ser descritos por equações diferenciais lineares invariantes no tempo). Tradicionalmente se faz necessário a modelagem do sistema. Um modelo matemático da planta ou do objeto a ser controlado é o primeiro passo em um projeto de um sistema de controle. O modelo matemático de um sistema dinâmico é definido como um conjunto de equações que representa a dinâmica do sistema com precisão ou, pelo menos, razoavelmente bem [Ogata 2010].

É possível o desenvolvimento de um sistema de controle sem a modelagem do sistema, conforme relatado anteriormente [Al-Hinai & Ibnkahla 2008] [Kumon et al. 2006]

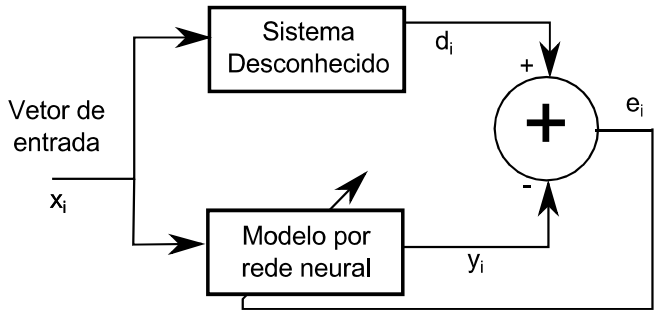

Figura 4.2: Diagrama de blocos de identificação de sistema.

[Canete et al. 2008] [Sebastien et al. 2001] [Ordonez & Passino 1999] para tanto necessitamos de uma rede neural treinada para que a mesma realize a tarefa de controlar a planta, no nosso caso, o VANT.

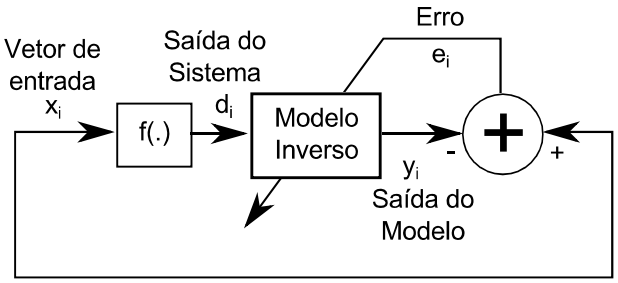

Figura 4.3: Diagrama de blocos de identificação de sistema inverso.

Desta forma, a formulação matemática do aprendizado do controle de vôo recai no problema de aprendizagem de uma rede neural com origem em um conjunto de dados de treinamento.

A habilidade de uma rede neural de aproximar um mapeamento de entrada e saída desconhecido pode ser explorado de duas formas [Haykin 2001]:

- *Identificação de sistemas*. A utilização de um conjunto de dados de exemplo pode treinar a rede neural como modelo do sistema. A diferença entre o d_i (saída do sistema) e a saída da rede y_i fornece o vetor de sinal de erro e_i. O sinal de erro é usado para ajustar os parâmetros livres da rede de forma a minimizar a diferença entre as saídas do sistema desconhecido (figura 4.2).
- *Sistema inverso*. O objetivo neste caso é construir um sistema inverso que produza um vetor x em reposta a um vetor d. Um vetor e (erro) representa a diferença entre x e a saída real da y rede neural. Este vetor de sinal de erro é utilizado para ajustar os parâmetros livres da rede neural de modo a minimizar a diferença entre as saídas do sistema inverso desconhecido e a da rede neural [Haykin 2001] (figura 4.3).

O aprendizado do controle de vôo dar-se-á pelo treino da rede neural de acordo com

os dados de telemetria do piloto real que foram coletados em sessões de decolagem, deslocamento e pouso.

O funcionamento da RNA (rede neural artificial) esta detalhado na seção 2.2, página 8.

Capítulo 5

Implementação

5.1 Apresentação do simulador

O *Flightgear* (FG) é um simulador de vôo de código aberto, desenvolvido em linguagem *C++*, que foi criado em 1996 e trabalha com o modelo de licenciamento *GNU General Public License v2*, sendo assim, livre para usar, modificar e distribuir. Suporta modelos de aeronaves, terreno, aeroportos e estruturas desenvolvidos por terceiros. Sua configuração é realizada através de arquivos com formato XML. O Flightgear pode ser executado no Windows, Linux, Mac OS-X, FreeBSD, Solaris e IRIX tanto em 32 bits e 64 bits com suporte a múltiplos processadores e múltiplos adaptadores gráficos *(Graphic Process Unit - GPU)* [*Flightgear Multi-GPU* 2011].

A capacidade de grande personalização do FG reside na sua modularidade. Cada um dos sub-sistemas do FG pode ser personalizado e substituído por um mais simples ou mais avançado, dependendo da necessidade do estudo ou experimento. A permutação entre os módulos é feita por uma simples reconfiguração dos arquivos de configuração do próprio FG que é baseado em XML. Esta capacidade modular permite o desenvolvimento de módulos mais refinados e precisos para cada aspecto do FG. Os módulos podem ser desenvolvidos em linguagem nativa do próprio FG, que é muito semelhante ao C++, ou em qualquer linguagem de programação.

5.2 Diagrama de módulos

Um modelo de dinâmica de vôo (Flight Dynamics Models - FDM) é um módulo externo ao FG que permite o uso de equações matemáticas para calcular e representar a ação de forças físicas em uma aeronave simulada e os mecanismos de controle. Cada módulo trabalha de forma independente e pode ser reescrito com a finalidade de refiná-lo ou até mesmo de adicionar uma característica não implementada.

Cada aeronave possui seu próprio modelo devido a características intrísecas que a torna diferente das demais. Entre estas diferenças temos os elementos aerodinâmicos e físicos (arrasto, tamanho das asas, capacidade de empuxo, coeficiente de fricção, área das asas, peso, altura, centro de gravidade, capacidade do tanque, capacidade de carga) [*Flightgear Flight Dynamics Model* 2011]. Simulações computadorizadas são usadas para modelar uma variedade ampla de sistemas físicos, incluindo aerodinâmicas, estru-

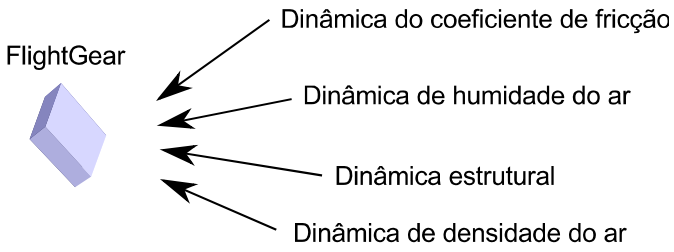

Figura 5.1: Módulos de dinâmica do FlightGear

turas, mecanismos de vôo, sistemas de controle, etc. e mais recentemente e significante, processos de manufatura [Hameed et al. 2009]. Cada um destes sistemas físicos e aerodinâmicos pode ser calculado em um módulo dedicado exclusivamente para tal finalidade.

O FG, como simulador, pode simular um ambiente com as características de vento, rajadas de vento, neve, temperatura e densidades do ar. Quando se faz necessário, podemos optar por um FDM para realizar tais funções. Neste escopo, o FG funcionará como um ambiente gráfico de renderização virtual para a aeronave simulada, representando os efeitos físicos-ambientais gerados matematicamente pelo FDM. É possível a utilização de diversos FDM em conjunto para prover uma simulação mais realística e com maior redundância. Um FDM poderia controlar as rajadas de vento e outro FDM poderia controlar a temperatura ambiente ou até mesmo o nível de precipitação de chuva ou neve ou densidade do ar [*Flightgear Aerodynamics* 2011] [*Flightgear YASim flight dynamics model* 2011] [*JB Simulator* 2011]. Cabe frisar que semelhante a cada característica física e aerodinâmica da aeronave que pode ter seus dados calculados em módulos independentes, as condições ambientais também podem ser calculadas e apresentadas no simulador utilizando módulos específicos que sejam destinados a cada aspecto. Outra vantagem é a utilização de dados reais para simular condições específicas ambientais e a utilização de dados ambientais em tempo real.

5.3 Detalhamento do módulo de controle

Um FDM pode, também, realizar o procedimento de controle de uma aeronave simulada. Usando diversos tipos de controles tais como proporcional-diferencial-integral (PDI), *backstepping, fuzzy*, baseados em Rede Neural (RN) entre outros. O controle baseado em rede neural foi desenvolvido e utilizado neste trabalho.

O acesso aos controles do FDM do FG é realizado lendo e setando valores dispostos em forma de uma árvore ou utilizado uma sequência de *bytecodes* encapsulados dentro de um pacote UDP/IP. A vantagem da utilização da arquitetura UDP é a possibilidade de distribuir o processamento do FDM e os mecanismos de controle inclusive com redundância. A utilização da estrutura, em forma de árvore, é permitir a interoperabilidade e padronização entre os diversos FDM. Poderíamos ter um FDM realizando os cálculo no subsistema de motores e outro no subsistema hidráulico responsável pelos atuadores dos airelons, profundores ou leme.

O FG pode ser controlado utilizando diversos mecanismos de controle físicos (mouse,

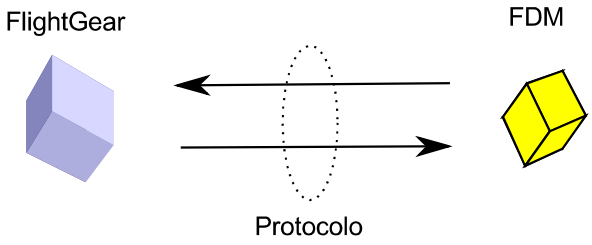

Figura 5.2: Integração do FlightGear com um FDM

teclado, joystick e manche). A aeronave simulada pode ser controlada por um conjunto de comandos enviados para o FG simulando os controles físicos. Foi desenvolvido um programa, em linguagem *C++* e posteriormente em MATLAB, que recebe e envia mensagens para o FG através do uso do UDP. Um *protocolo*, no FG, define o formato e a sequência do *bytecode* que será recebido e enviado para o FG. O protocolo é definido em um arquivo XML (figura 5.5).

A função do protocolo no FG é estabelecer o formato do mecanismo de troca de mensagens que o FG receberá e enviará para o modelo de dinâmica de vôo. A estrutura do protocolo é definida de acordo com a necessidade do experimento. Pode-se utilizar algumas variáveis contidas na extensa lista de variáveis pré-definidas. Algumas destas variáveis são do tipo *somente leitura* (latitude, longitude, data e hora, etc). Outras devem ser utilizadas para manipular os controles efetivos da aeronave simulada (figura 5.5).

A grande maioria das variáveis que serão utilizadas para controle da aeronave simulada estão no formato *float* e tem uma faixa operacional de -1 até $+1$. Se formos atuar o leme para o valor de 5^o para a esquerda, enviamos para o FG o valor aproximado de $-0,056$. O FG representa para o controle do leme 0^o como valor de 0, 90^o para a direita como valor de $+1$ e 90^o para a esquerda com o valor de -1. Estes valores são semelhantes para os diversos controles. Alguns controles possuem o valor operacional, do tipo *float*, de 0 até $+1$, como, por exemplo, o *throttle*, que no valor 0.65, representa 65% da potência máxima (não computado o uso de pós-queimadores se for o caso).

Vale ressaltar que cada aeronave (ou VANT) possui limitações física quanto à manobrabilidade. Se uma aeronave pode realizar uma guinada (eixo z, figura 2.1) máxima à esquerda de $5,8^o$ em velocidade de cruzeiro, se for especificado uma ação proveniente do FDM de controle com o valor de 0.70% para realizar uma guinada à esquerda, a aeronave irá realizar a guinada à esquerda de $4,06^o$.

5.4 Detalhamento do caminho de dados (entrada/saída)

O módulo de controle do VANT a ser trabalhado no ambiente simulado no FG foi inicialmente desenvolvido em C++. Em um segundo momento foi realizado a migração para o MATLAB já prevendo usos futuros e novas funcionalidades. Cabe ressaltar que as mesmas funcionalidades encontradas no MATLAB podem ser encontradas em diversas bibliotecas do C++. A utilização do MATLAB foi colocada em primeiro plano devido a grande variedade de módulo já incorporados e a facilidade de uso. Devido ao tempo e a

quantidade de mão de obra empregada percebeu-se que o desenvolvimento na linguagem C++ iria extrapolar o tempo destinado aos experimentos. Como um dos trabalhos futuros esta a implementação em C++ de todos os módulos utilizados do MATLAB.

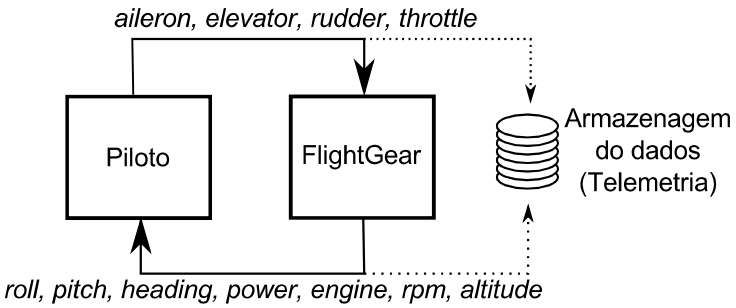

Figura 5.3: Caminho de dados - Módulo de coleta.

Foram desenvolvidos dois módulos no MATLAB. O primeiro módulo (módulo de coleta) foi destinado a proceder a coleta de dados para posteriormente realizar o treinamento da rede neural. O segundo módulo (módulo de controle) foi o controle do VANT utilizando uma rede neural já treinada. Estes dois módulos realizam uma troca de mensagens com o FG utilizando o protocolo UDP, da pilha TCP/IP. Um arquivo do tipo XML foi configurado para definir o protocolo de entrada e outro foi configurado para definir o protocolo de saída.

O primeiro módulo (figura 5.3) tem como objetivo a coleta de dados provenientes do FG em relação ao controle do UAV. Um piloto real realizou um total de 106 decolagens e 41 pairagens. Cada uma destas decolagens e pairagens gerou um conjunto de dados que foram utilizados em um estágio posterior para treinar a rede neural. Os dados coletados provenientes do FG foram:

- *latitude*. Indica a latitude em que encontra-se o VANT.
- *longitude*. Indica a longitude em que encontra-se o VANT.
- *altitude*. Indica a altitude em que encontra-se o VANT.
- *roll*. Valor nominal do ângulo de rolagem do VANT (figura 2.1).
- *pitch*. Valor nominal do ângulo de arfagem do VANT (figura 2.1).
- *yaw*. Valor nominal do ângulo de guinada do VANT (figura 2.1).
- *heading*. Direção em que a parte frontal do VANT esta direcionada. 0^o indica o norte, 180^o indica o sul.
- *throttle_in*. Valor indicativo da potência. Este valor neste módulo indica a potência colocada pelo piloto real.
- *power*. Indica a potência aplicada ao motor.
- *engine*. Valor binário que indica se o motor esta em funcionamento ou não
- *aileron*. Valor nominal definido pelo piloto real usando um manche. Usado para compensar a rolagem do VANT.
- *elevator*. Valor nominal definido pelo piloto real usando um manche. Usado para compensar a arfagem do VANT.
- *rudder*. Valor nominal definido pelo piloto real usando um manche. Usado para compensar a guinada do VANT.

- *RPM*. Valor que define as rotações por minuto do motor.

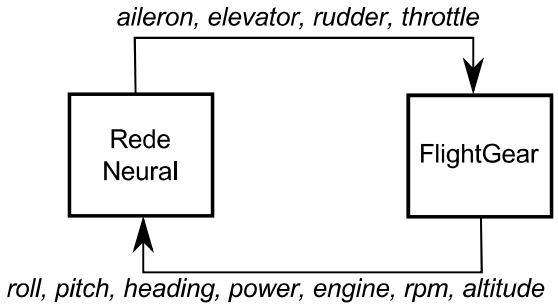

Figura 5.4: Caminho de dados - Módulo de Controle.

O segundo módulo tem como objetivo o controle efetivo do VANT no FG. Este módulo tem saída direta da rede neural e como entrada os valores produzidos pelo simulador. Os dados de saída do simulador, *roll, pitch, heading, power, engine, RPM* e *altitude* são apresentados como o vetor de entrada para a rede neural treinada com os dados do primeiro módulo. o vetor de saída (*alieron, elevator, rudder e throttle_in*) da rede neural já treinada é aplicado diretamente no FG. A definição parcial do arquivo de protocolo de entrada e saída pode ser vizualizado na figura 5.5.

Na figura 5.4, uma rede neural já treinada recebe como vetor de entrada os dados gerados pelo VANT no FG. Estes dados alimentam a RN que por sua vez gera saídas que são aplicadas ao controles do VANT no FG. A rede neural aprendeu a realizar as compensações necessárias durante a decolagem semelhantemente ao que o piloto humano realizaria.

```
<?xml version="1.0"?>
<PropertyList>
<generic>
<input>
<line_separator>newline</line_separator>
<var_separator> </var_separator>

  <chunk>
   <name>aileron</name>
   <type>float</type>
 <format>%f</format>
   <node>/controls/flight/aileron</node>
  </chunk>

  <chunk>
   <name>elevator</name>
   <type>float</type>
 <format>%f</format>
   <node>/controls/flight/elevator</node>
  </chunk>

  <chunk>
   <name>rudder</name>
   <type>float</type>
 <format>%f</format>
   <node>/controls/flight/rudder</node>
  </chunk>

  <chunk>
   <name>throttle</name>
   <type>float</type>
 <format>%f</format>
   <node>/controls/engines/engine/throttle</node>
  </chunk>

</input>
</generic>
</PropertyList>
```

```
<?xml version="1.0"?>
<PropertyList>
<generic>
<output>
<line_separator>newline</line_separator>
<var_separator> </var_separator>

  <chunk>
   <name>latitude-deg</name>
   <type>float</type>
   <format>%f</format>
   <node>/position/latitude-deg</node>
  </chunk>

  <chunk>
   <name>longitude-deg</name>
   <type>float</type>
   <format>%f</format>
   <node>/position/longitude-deg</node>
  </chunk>

  <chunk>
   <name>altitude-ft</name>
   <type>float</type>
   <format>%f</format>
   <node>/position/altitude-ft</node>
  </chunk>

  <chunk>
   <name>roll-deg</name>
   <type>float</type>
   <format>%f</format>
   <node>/orientation/roll-deg</node>
  </chunk>

</output>
</generic>
</PropertyList
```

Figura 5.5: Conteúdo parcial dos arquivos de protocolo.

Capítulo 6

Experimentos e Resultados

6.1 Coleta de dados

Para que uma rede neural seja treinada ela necessita de um conjunto de dados de treinamento. Neste trabalho, o treino da rede neural é apresentar um controle neural capaz de substituir um piloto durante os procedimentos de decolagem e pairagem. O processo de aprendizagem da rede neural é o processo de aprendizagem supervisionada.

Os dados de decolagem foram capturados durante o intervalo de 40 segundos a partir do momento que acionou-se o motor do VANT no simulador. Efetivamente somente os 20 segundos finais foram utilizados pois os 20 segundos iniciais caracterizam-se por somente aumento da velocidade do motor e aumento do torque. Todos os 40 segundos foram apresentados para o treino da rede neural.

O piloto real realizou inicialmente um total de 106 procedimentos de decolagem. Com 40 segundos armazenados de cada decolagem, foi gerado um total de 4.240 segundos (70,66 minutos ou 1,17 hora) de dados de decolagem com tamanho em disco de aproximadamente 1,23 Mb. A taxa de coleta foi de 10 (dez) amostragens por segundo.

Realizando uma análise em um dos procedimentos de decolagem juntamente com as respostas do VANT temos que quando a potência alcança o valor de 0.4 ($t = 3770$), equivalente a 60% da potência total (nos VANTs do tipo helicóptero, quanto maior a potência, menor o torque. A potência trabalha de forma inversa), começa o efeito de decolagem e uma pequena variação na altitude. A variação negativa é devido ao bico do VANT ficar mais próximo ao solo durante o procedimento de decolagem. No t=3780, efetivamente ocorre a decolagem do VANT.

Para o procedimento de pairagem foi realizado inicialmente o procedimento de decolagem. Quando o VANT encontrava-se em estado estável após o decolagem iniciava-se a coleta dos dados de pairagem com o piloto real operando o manche e tentando deixar o VANT a uma mesma altura e mesma posição por durante 20 segundos no mínimo. Foram realizados diversos procedimentos de pairagem e deste conjunto, foram filtrados e escolhidos 41 procedimentos que apresentavam os 20 segundos de tempo mínimo e de menor variação de altura e posição para que tal conjunto fosse destinado para o vetor de entrada da rede neural no procedimento de pairagem.

Os 41 procedimentos de pairagem geraram um total de 820 segundos (13,66 minutos ou 0,227 hora) de dados com tamanho em disco de aproximadamente 492 Kb. A taxa de

coleta foi de 10 (dez) amostragens por segundo.

6.2 Entrada da rede neural

Utilizando o paradigma de aprendizagem supervisionada, utiliza um professor, o piloto real, para ensinar à rede neural (sistema de aprendizagem) as respostas necessárias apartir de um ambiente, aqui representado pelo VANT dentro do FG. Na figura 6.4, o professor e a rede neural são expostos a um vetor de treinamento (exemplo) retirado do ambiente. As variáveis coletadas para o vetor de treinamento foram rolagem, afagem, guinada, direção do VANT (usando a bússola), potência do motor, throttle do motor, RPM e altitude.

Em virtude do conhecimento prévio, o professor é capaz de fornecer à rede neural uma resposta desejada para aquele vetor de treinamento. A resposta desejada representa a ação ótima a ser realizada pela rede neural. Os parâmetros da rede neural são ajustados sob a influência combinada do vetor de treinamento e do sinal de erro. O sinal de erro é definido como a diferença entre a resposta desejada e a resposta real da rede. O objetivo é fazer a rede neural emular o professor (piloto real). Passo a passo, o conhecimento do professor é transferido para a rede neural através do treinamento [Haykin 2001]. Temos que ressaltar que a medida de desempenho (o quanto a rede neural está apta a substituir o professor) é definida pela capacidade da rede neural de realizar o procedimento de decolagem e pairagem de maneira eficiente e estável.

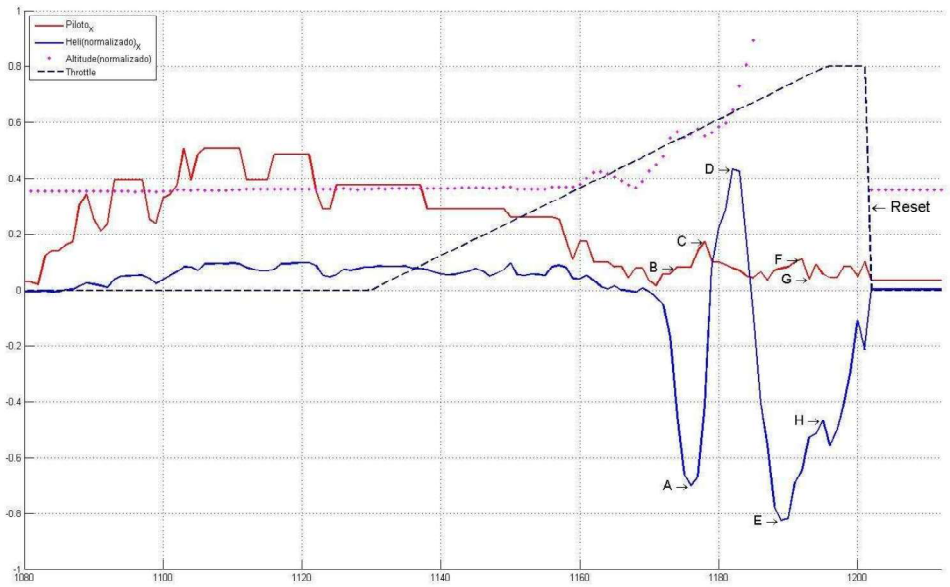

Figura 6.1: Análise da resposta do piloto. Eixo X (deslocamento linear).

A figura 6.1 apresenta no eixo Y a resposta do piloto, tendo como valor máximo (com

valor +1) o manche do piloto colocado no ponto máximo, à direita, e com o valor máximo negativo de -1, como o manche ao máximo para a esquerda. O eixo X caracteriza-se pelo tempo em segundos a partir do início do procedimento de ativação dos motores.

Antes que os dados possam ser colocados diretamente na rede neural é necessário uma análise da resposta do piloto durante o procedimento de decolagem. Esta análise é apresentada na figura 6.1. Baseando-se nesta figura, temos os seguintes fatos:

- Até o $t = 1160$ aproximadamente, o VANT não decolou e a potência está próximo de 60%.
- O VANT tenta fazer uma rolagem para a esquerda (A) e o piloto realiza uma compensação para a direita (B).
- A compensação realizada pelo piloto foi grande demais e chegou até o valor (C). Desta forma o VANT acompanhou a acão de compensação e começou a rolar para a direita (D).
- O piloto realiza diversas compensações menores e o VANT tem uma grande rolagem para a esquerda (E).
- Pequenas compensações realizados pelo piloto (F) e (G) em um pequeno intervalo de tempo fazem com que o VANT tenha o processo de rolagem desfeito (H).
- Outras compensações, próximo a $t = 1200$, fazem com que o VANT seja estabilizado onde pouco depois é aplicado o *reset* e um novo procedimento será realizado.

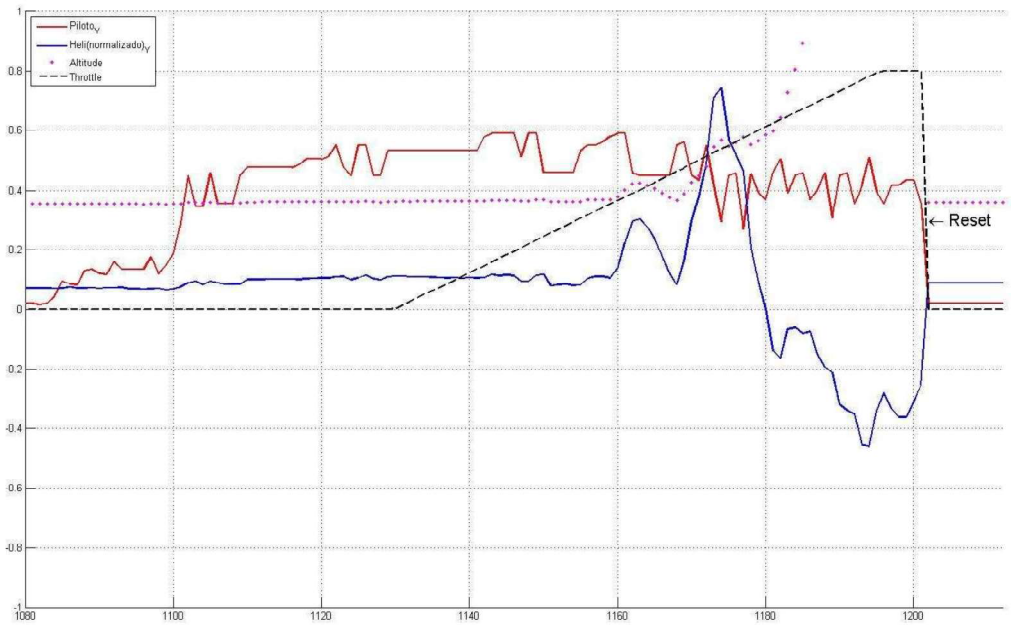

Figura 6.2: Resposta do piloto. Eixo Y.

Semelhante ao realizado com a resposta do piloto no eixo X (rolagem), uma análise

semelhante pode ser realizada no eixo Y (arfagem) e no eixo Z (guinada), respectivamente figura 6.2 e figura 6.3.

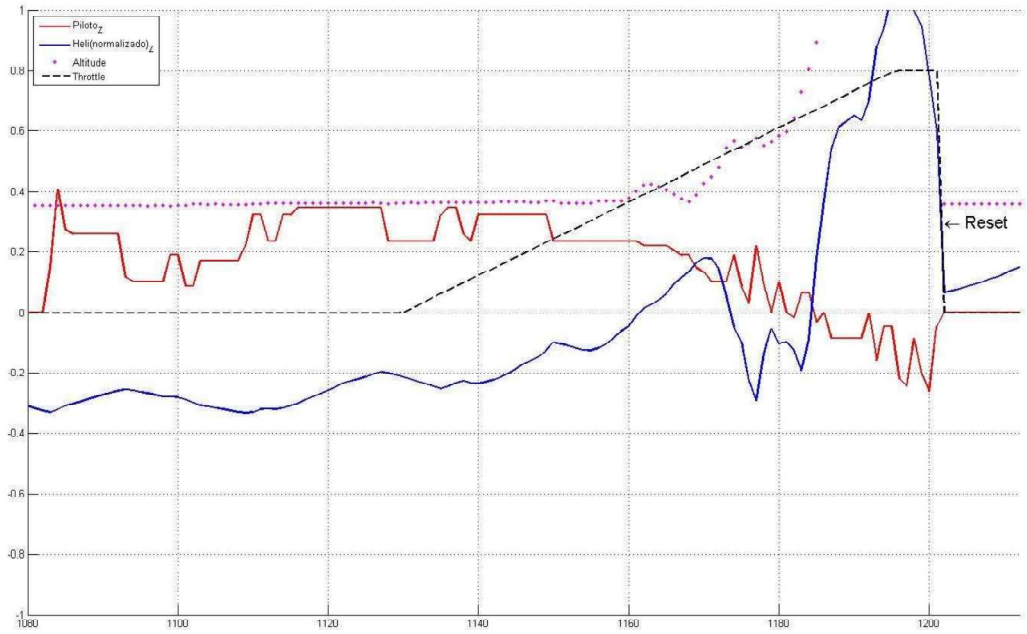

Figura 6.3: Resposta do piloto. Eixo Z.

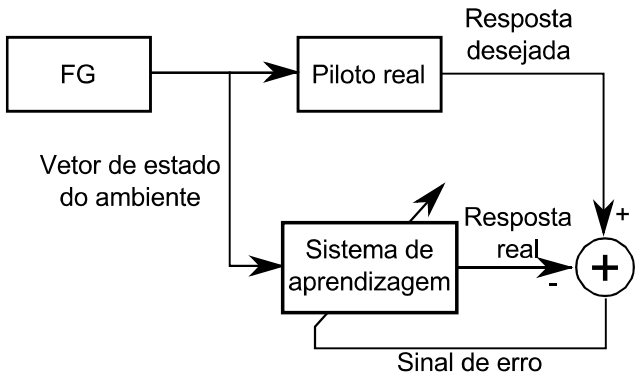

Figura 6.4: Aprendizagem com professor.

Os dados coletados durante todas as decolagens e pairagens foram inicialmente colocados em uma RN do tipo perceptrons de múltiplas camadas (MLP, *multiplayer perceptron*). A arquitetura da rede utilizada inicialmente foram de 8 neurônios na primeira camada oculta, 16 neurônios na segunda camada oculta, 8 na terceira camada oculta. Foram criadas e treinadas 03 (três) redes neurais. Cada rede neural controla um dos atuadores igualmente ao que o piloto humano faria, são eles *aileron, elevator* e *rudder*.

Posteriormente verificou-se que a arquitetura não era apropriada e buscou-se variações na arquitetura da RN.

Para efeitos de abreviação, definiremos aqui que uma rede neural com arquitetura de 8 neurônios na primeira camada oculta, 16 neurônios na segunda camada oculta, 8 na terceira camada oculta terá abreviação de RN[121]. Uma rede neural com arquitetura abreviada na forma RN[22442] possuirá 16 neurônios na primeira camada oculta, 16 neurônios na segunda camada oculta, 32 na terceira camada oculta, 32 na quarta camada oculta e 16 na quinta camada oculta. Os valores representados dentro dos colchetes serão multiplicados pelo valor 08 (oito).

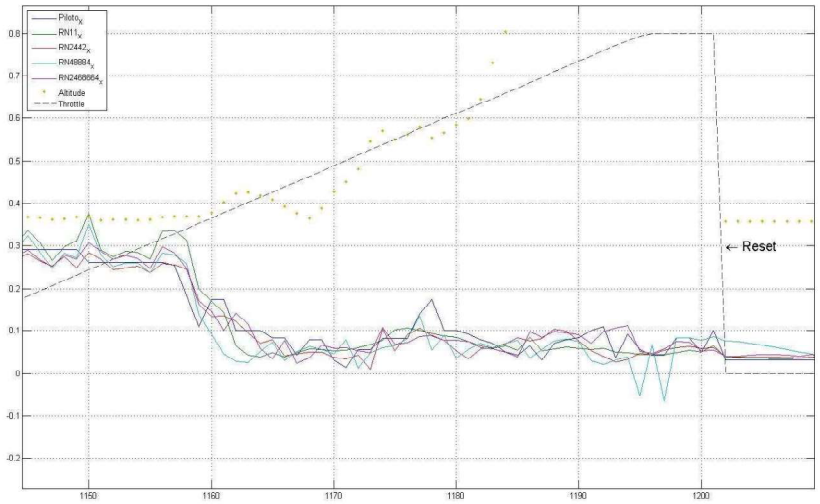

Figura 6.5: Comparativo da resposta das saídas da RN.

A primeira rede neural treinada (RN[121]) apresentou resultado insatisfatórios durante o teste de decolagem no FG. Diversas arquiteturas de RN foram testadas a fim de encontrar a que gerassem as melhores aproximações para substituir o piloto humano por um piloto baseado em RN. A figura 6.5 mostra um comparativo no momento da decolagem entre a resposta do piloto humano (obtido pelo telemetria) e algumas RN com arquiteturas diferentes.

Uma análise mais apurada da figura 6.5, e posteriores testes práticos no FG, demostraram que a RN[11] generalizou demais a resposta do piloto humano e não conseguiu realizar o procedimento de decolagem satisfatório. A RN[2442] obteve uma aproximação melhor da resposta do piloto humano sem ainda ter o sucesso necessário. A RN[2468664] gerou as respostas necessárias para o controle efetivo em substituição do piloto humano. Em uma observação mais crítica entre os controles do piloto humano e a resposta da RN[2468664] percebe-se uma maior aproximação de ambos, desta forma, a RN[2468664] aproxima-se mais da resposta do piloto humano.

Apesar de provê uma possibilidade rápida de análise, a figura 6.6 nos mostra parcialmente a diferença entre as redes neurais RN[11] e RN[2468664]. Uma análise mais

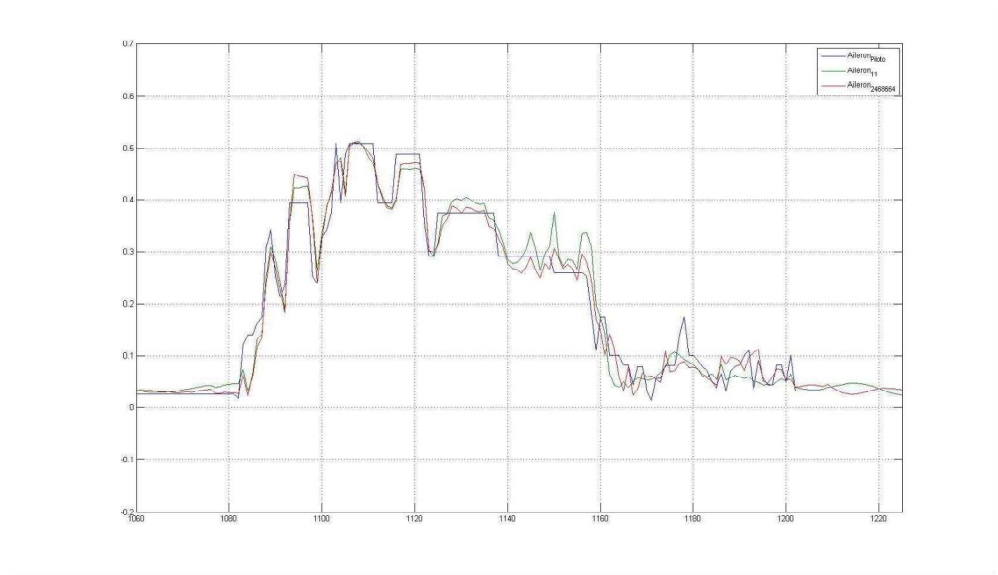

Figura 6.6: Comparativo entre a menor e maior RN.

apurada da figura 6.6 são as figuras 6.7, 6.8 e 6.9 que demostram com uma maior aproximação a resposta da RN com a resposta do piloto. A diferença entre tais respostas pode ser visto na figura 6.10. Quanto mais próximo a 0 (zero) melhor a resposta da RN em relação ao piloto.

6.3 Variações da arquitetura e tempo de treinamento

A arquitetura inicialmente escolhida foi a RN[121] que apresentou resultados insatisfatórios sob ponto de vista de eficiência. Desta forma foi necessário o treinamento de diversas RN com arquiteturas diferentes. As arquiteturas foram escolhidas de forma aleatória. Conforme demonstrado em [Mettler et al. 2000], no tocante à arquitetura da rede, tipicamente, uma grande quantidade de unidades escondidas permite melhores valores de convergência entre as saídas da rede e as saídas-alvo durante o treinamento. Contudo, um número pequeno de unidades escondidas resulta em uma generalização melhor quando exposta a situações não apresentadas durante os testes.

Os resultado insatisfatórios obtidos na RN[121] foram decorrentes de uma generalização da RN. Para que pudéssemos ter um controlador de melhor eficácia, teríamos que aumentar o tamanho da RN e a quantidade de camadas ocultas. Diversos tamanhos foram testados até que a RN[2468664] foi treinada e obteve um desempenho adequado aos experimentos realizados.

A figura 6.11, 6.12 e 6.13, respectivamente a média, variância e desvio padrão dos resultados de cada uma das redes treinadas para cada um dos eixo (rolagem, guinada e arfagem) em relação à resposta do piloto real.

O tempo de treinamento da RN tem o seu crescimento de forma exponencial devido a quantidade neurônios em cada camada e a quantidade de camadas ocultas em cada RN

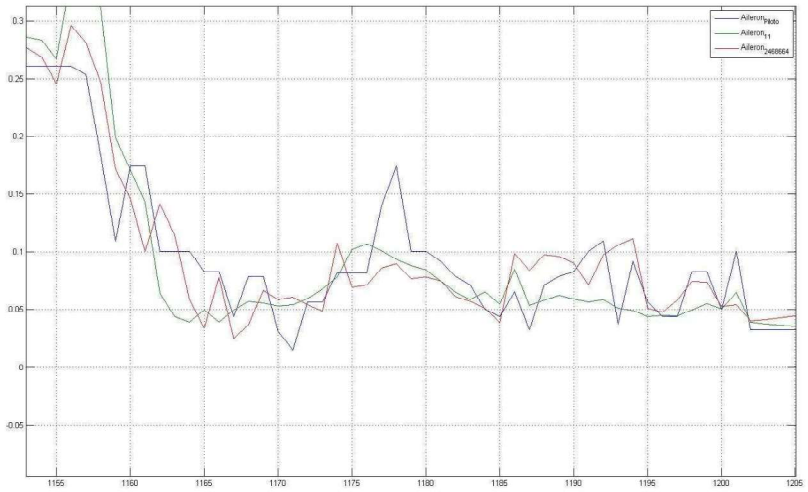

Figura 6.7: Comparativo detalhado 1.

treinada conforme a tabela 6.3 e a figura 6.14.

6.4 Imagens e vídeos

Um vídeo com os testes de cada uma das RN com as arquiteturas aqui apresentadas esta disponível em http://www.youtube.com/watch?v=aqdel3vqym0 . Todos os experimentos foram realizados nas mesmas condições e no mesmo ambiente. A diferença recai na resposta do VANT para com o módulo de controle baseado em redes neurais (neurocontrolador).

A figura 6.15 nos mostra o FDM inicial desenvolvido em C++. A janela com título *mapa* é um FDM que recupera a localização diretamente do Google Maps. A marcação utiliza os valores de latitude e longitude que o FG envia para o FDM.

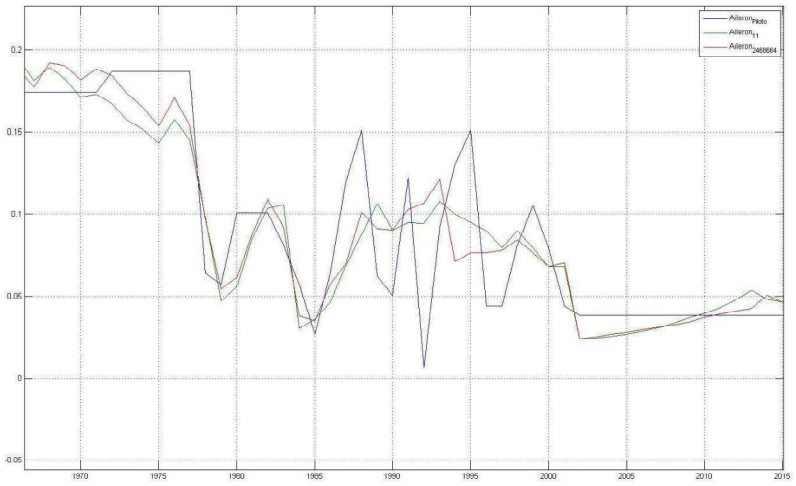

Figura 6.8: Comparativo detalhado 2.

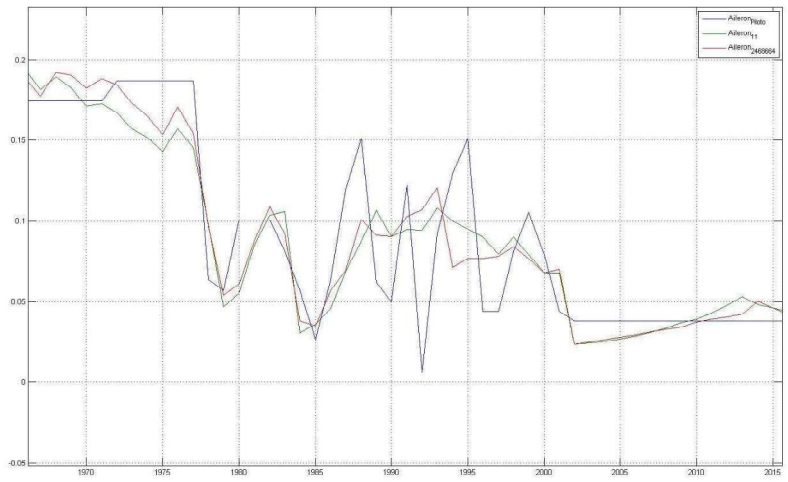

Figura 6.9: Comparativo detalhado 3.

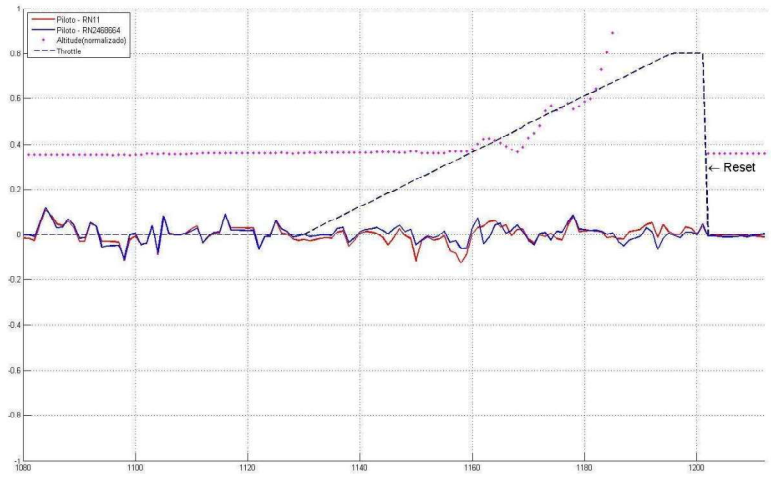

Figura 6.10: Diferença entre as respostas das RN.

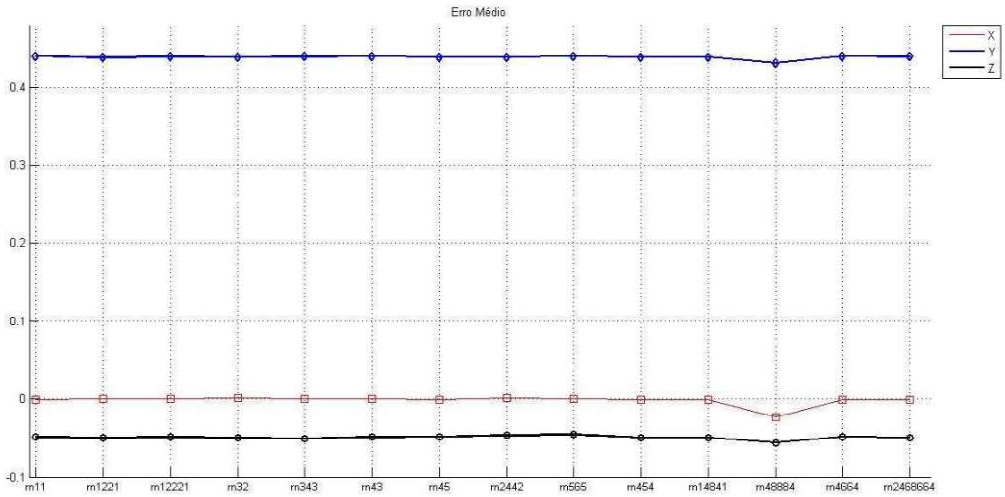

Figura 6.11: Tempo de treinamento - Média.

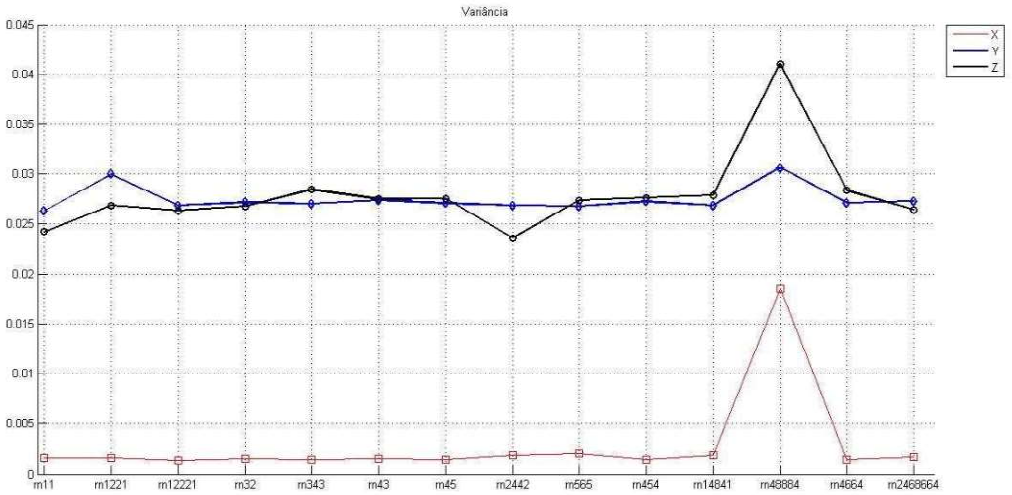

Figura 6.12: Tempo de treinamento - Variância.

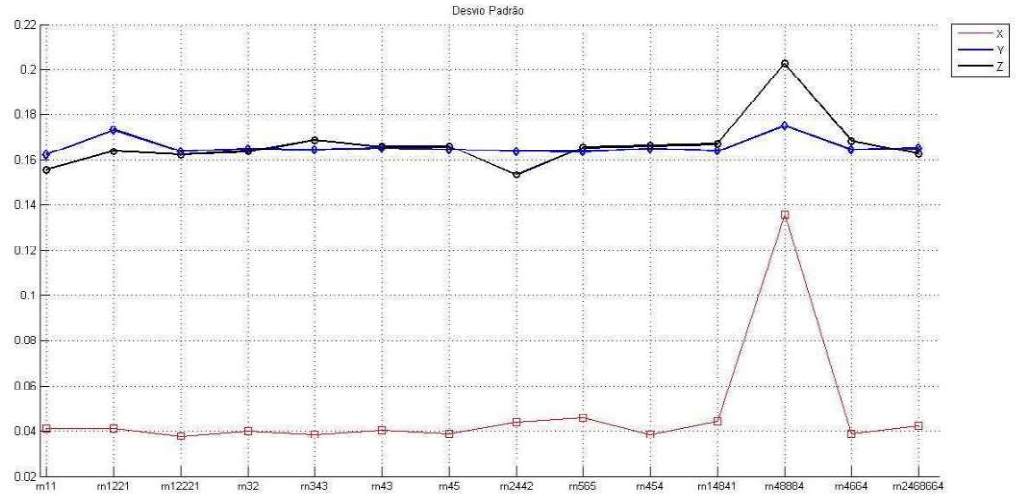

Figura 6.13: Tempo de treinamento - Desvio padrão.

Arquitetura	Tempo de treinamento (horas)
RN[11]	0,08
RN[1221]	0,15
RN[32]	0,25
RN[343]	0,44
RN[43]	0,444
RN[45]	0,48
RN[2442]	1,00
RN[564]	1,13
RN[454]	1,36
RN[14841]	2,68
RN[48884]	3,23
RN[46645]	5,88
RN[2468664]	38,22

Tabela 6.1: Tempo de treinamento de RN com arquiteturas diferentes.

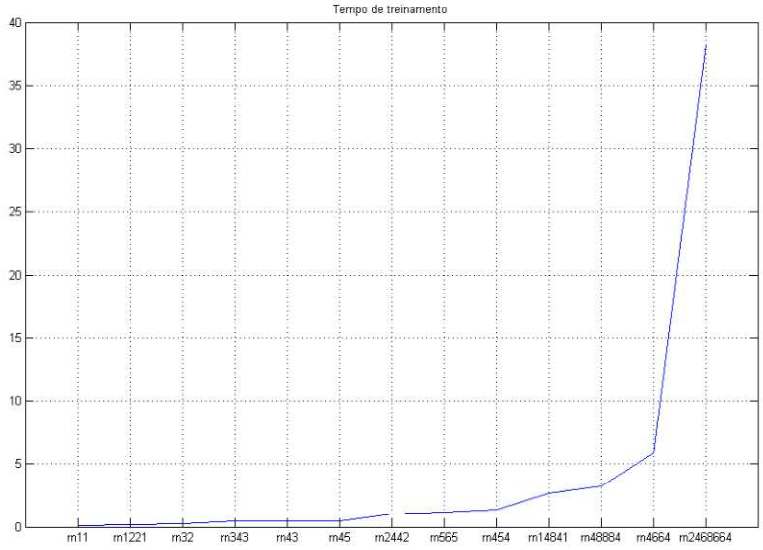

Figura 6.14: Tempo de treinamento das redes neurais.

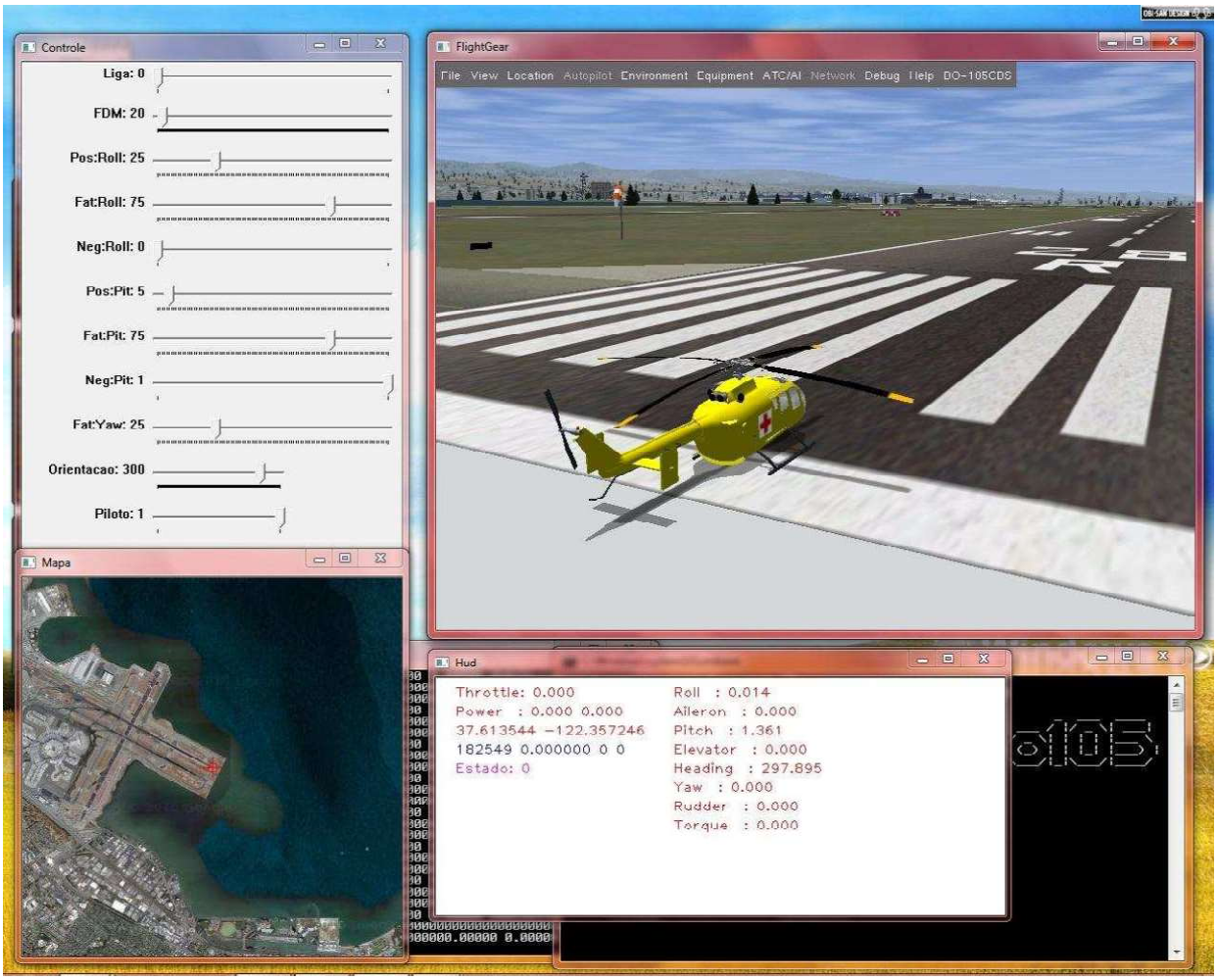

Figura 6.15: FDM baseado em C++.

Capítulo 7

Conclusão

7.1 Resultado das observações

Temos 03 (três) conclusões importantes a respeito dos resultados obtidos neste trabalho.

- *Criação do sistema de controle*. É possível a criação de um sistema de controle para VANTs do tipo helicóptero baseado em redes neurais sem a modelagem do sistema (caracterizado por funções de espaço de estado). A rede neural teve êxito em aprender a dinâmica do VANT que foi gerada pelo piloto real (professor). Em [Buskey et al. 2001] o mesmo nos relata que tentativas de usar métodos de controle tradicionais para a contrução de leis de controle inversa para a planta de helicópteros podem, inevitavelmente, encontrar problemas quando encarados com a variância temporal e natureza não-linear destes sistemas. Uma das alternativas é a utilização de uma rede neural como sistema de controle (neurocontrolador).
- *Qualidade dos dados de aprendizagem*. Quanto melhor for o procedimento realizado por um piloto real (ou o conjunto destes procedimentos), melhor será o repasse para a rede neural. Através de exemplos, uma rede neural aprende a experiência do piloto real (professor). Quando mais próximos um procedimento for considerado como quase-ótimo ou ótimo pelo professor, melhores serão os resultados da rede neural e o neurocontrolador apresentará respostas melhores tornando assim os procedimentos mais estáveis. Nos procedimentos de pairagem realizados por um piloto real, somente 41 foram utilizadas. Poderíamos ter utilizado todos os conjuntos de dados contudo observou-se que os exemplos ruins contribuíam negativamente para o aprendizado da rede tornando as respostas do neurocontrolador insatisfatórias. Esta contribuição negativa ficou perceptível nos testes utilizando todos os dados (sem filtros) de pairagem da RN.
- *Tamanho da arquitetura da rede*. Redes neurais com uma arquitetura com grande quantidade de camadas ocultas e quantidade de neurônios artificiais geram resultados melhores. Respostas melhores condizem com que [Mettler et al. 2000] nos relata a respeito da quantidade de neurônios e camadas ocultas. Percebemos que um sistema de controle (neurocontrolador) baseado em redes neurais mais generalizado, que utiliza menos camadas ocultas e menos neurônios, não suporta com um desempenho adequado e estabilidade necessária a dinâmica não linear e fortemente

acoplada de um VANT.

Uma observação de menor importância foi a recursividade do treino. Com uma rede neural já treinada, foi feito um conjunto de experimentos, gerando assim dados telemetria. Estes dados foram apresentados para treinamento para uma nova rede neural (2^a geração) e a mesma foi treinada. Com esta nova rede neural treinada realizou-se um conjunto de procedimentos de decolagem no VANT usando o Flightgear. Observou-se que o piloto real durante o procedimento de decolagem realizava um pequeno desvio para a direita. Esta característica ficou mais perceptível quando usada a 2^a geração da rede neural treinada. Um estudo mais detalhado poderia gerar um melhor entendimento desta característica/aspecto.

7.2 O que foi feito e aplicações

Foi desenvolvido um ambiente de coleta de dados de entrada e saída para com o Flightgear (figura 5.3) possibilitando o treino de diversos VANT de forma simulada e em paralelo utilizando diversos pilotos reais.

Como aplicações deste trabalho temos a possibilidade de simulação e desenvolvimento de neurocontroladores com um extensa gama de configurações, condições atmosféricas, testes de performance e comparativos entre controles tradicionais (PID) e outros neurocontroladores. A integração com o MATLAB nos permite a utilização de módulos (*toolbox*) que possam ser utilizados como parte de simulação de propriedades físicas do VANT, simulação de propriedades aerodinâmicas e atmosféricas, controladores do tipo *fuzzy*, PID, neurocontroladores e diversos outros.

Com o módulo de coleta de dados, podemos gerar dados para uma análise posterior ou uma análise distribuída. A análise distribuída utiliza um canal de comunicação TCP/IP. Um computador poderia gerar as simulações e outro computador usando um programa ou o MATLAT analisar tais dados. Geograficamente, este outro computador poderia estar situado em qualquer lugar que fosse possível uma conexão de rede ou Internet. Esta possibilidade de distribuição pode ser utilizada para uma redundância em quaisquer subsistema, pois o Flightgear é fortemente modular.

7.3 Trabalhos futuros - aprendizagem por reforço (combinação)

Demonstramos que um neurocontrolador construído utilizando o paradigma de aprendizado com professor foi gerado satisfatoriamente. O próximo possível passo é a utilização de programação dinâmica para construir um neurocontrolador sem um professor. Esta tarefa será possível com a utilização do aprendizado por reforço, notadamente a *aprendizagem Q*. O neurocontrolador terá apenas informações do ambiente e pelo método de tentativa/erro e melhor recompensa, produzirá sequências de controle que permitirá o VANT decolar, pairar, deslocar-se e pousar de forma eficiente, estável e segura. A integração com o ambiente do Flightgear é fundamental principalmente nos testes pois o erro

na decolagem produz uma destruição do VANT simulado. A realização deste método na vida real torna-se dispendioso.

A utilização do Flightgear como ambiente de testes nos possibilita um ambiente de testes altamente modular e de fácil operacionalização. Apesar destes atrativos, o Flightgear trabalha em tempo real, ou seja os tempos gastos em simulação são tempos reais. Uma possibilidade é a utilização do JSBSim (http://jsbsim.sourceforge.net/) para acelerar o tempo dedicado aos testes e gerar uma dinâmica mais apurada dos controles de vôo. O JSBSim pode ser acoplado ao Flightgear como um módulo que trabalhará a dinâmica do vôo.

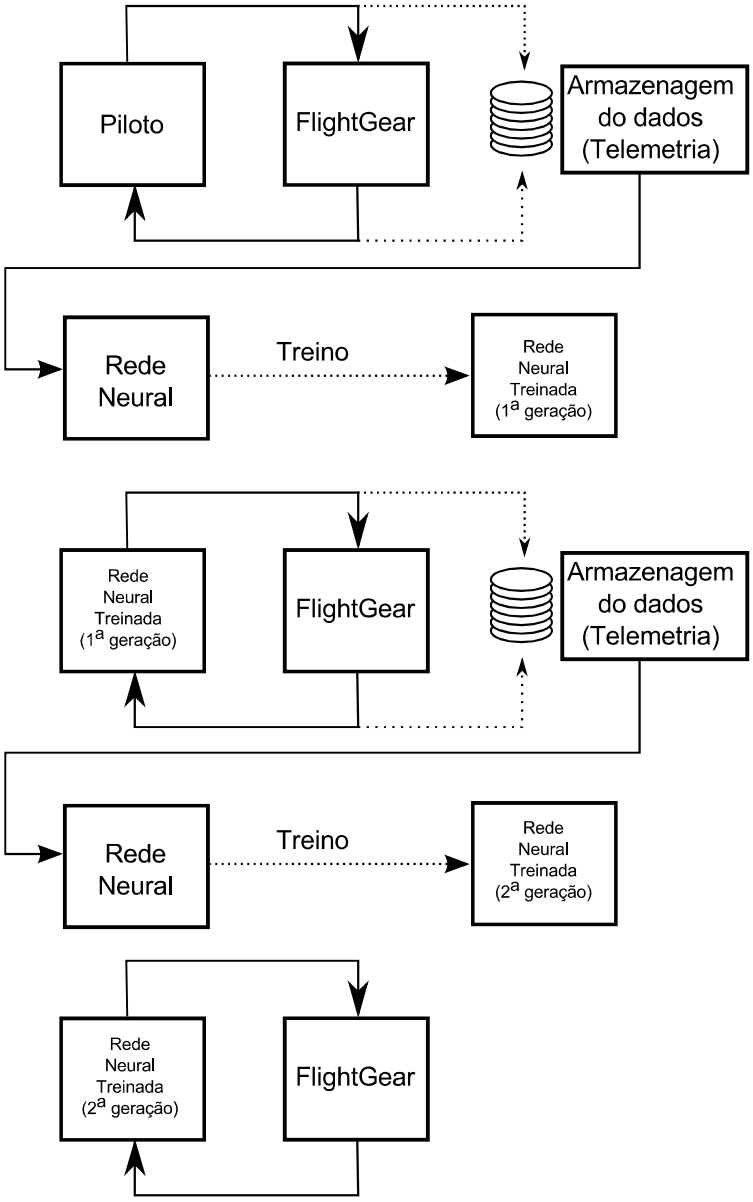

Figura 7.1: Treino recursivo.

Referências Bibliográficas

Ahmed, Bilal, Hemanshu R. Pota & Matt Garratt (2008), 'Flight control of a rotary wing uav - a pratical approach', *Proccedings of the 47th IEEE Conference on Decision and Control* pp. 5042–5047.

Al-Hinai, A. & M. Ibnkahla (2008), 'Neural network nonlinear mimo channel identification and receiver design', *ICC* pp. 835–83.

Anton, Howard & Robert C. Busby (2006), *Algebra Linear Contemporanea*, 1ª edição, Editora Bookman, Porto Alegre, Brasil.

Buskey, G., G. Wyeth & J. Roberts (2001), 'Autonomous helicopter hover using an artificial neural network', *Proceddings of the 2001 IEEE Internacional Conference on Robotics and Automation* pp. 1635–1640.

Canete, J. Fernandez, S. Gonzalez-Perez & P. del Saz-Orozco (2008), 'Software tools for system identification and control using neural networks in process enginneering', *International Journal of Intelligent Systems and Technologies* .

Chen, Wenbing, Xia Xu, Ming Liu, Yunjian Ge & Min Zhu (2007), 'Sensor based adaptive neural network control for small-scale unmanned helicopter', *Proceeding of 2007 International Conference on Information Acquisiton* pp. 187–191.

Eom, Il Yong & Seul Jung (2007), 'Neural network compesation for force tracking control of an autonomous helicopter system', *International Conference on Control, Automation and Systems* pp. 435–440.

Flightgear Aerodynamics (2011), Relatório técnico.
URL: `http://wiki.flightgear.org/JSBSim\_Aerodynamics`

Flightgear Flight Dynamics Model (2011), Relatório técnico.
URL: `http://wiki.flightgear.org/FDM\_engine\_feature\_standardization`

Flightgear Multi-GPU (2011), Relatório técnico.
URL: `http://wiki.flightgear.org/Multi\_core\_and\_Multi\_GPU\_support`

Flightgear YASim flight dynamics model (2011), Relatório técnico.
URL: `http://wiki.flightgear.org/Yasim`

Hameed, Tahir, Wang Wei & Ren Zhang (2009), 'Conceptual designing - unmanned aerial vehicle flight control system', *International Conference on Electronic Measurement e Instruments, 2009* pp. 712–716.

Hashimoto, S., T. Ogawa, S. Adachi, A. Tan & G. Miyamori (2000), 'System identification experiments on large-scale unmanned helicopter for autonomous flight', *Proceedings of the 2000 IEEE International Conference on Control Application* pp. 850–855.

Haykin, S (2001), *Redes Neurais: Principios e Pratica*, 2ª edição, Bookman, Porto Alegre, Brasil.

JB Simulator (2011), Relatório técnico.
URL: `http://jsbsim.sourceforge.net/index.html`

Jin, Guo-Dong, Liang-Xian Gu & Li-Bin Lu (2009), 'Uav simulator-based simulation of flight control system', *International Workshop on Intelligent Systems and Applications* pp. 1–4.

Kinoshita, Takuya & Fumiaki Imado (2006), 'A study on the optimal flight control for an autonomous uav', *Proccedings of the 2006 IEEE International Conference on Mechatronics and Automation* pp. 996–1001.

Krishnakumar, K. & S. Sawhney (1991), 'Simulator based adaptative helicopter training using neural networks', *International Conference on Systems, Man, and Cybernetics, 1991* pp. 1511–1516.

Kumon, Makoto, Yuya Udo, Hajime Michihira, Masanobu Nagata, Ikuro Mizumoto & Zenta Iwai (2006), 'Autopilot system for kiteplane', *IEEE/ASME Transactions on Mechatronics* pp. 615–624.

Mettler, Bernard, Takeo Kanade & Mark B. Tischler (2000), System identification modeling of a model-scale helicopter, Relatório técnico, Carnegie Mellon University, Pittsburgh.

Ogata, K. (2010), *Engenharia de Controle Moderno*, 5ª edição, Editora Pearson Prentice Hall, Sao Paulo, Brasil.

Ordonez, R. & K. M. Passino (1999), 'Stable multi-input multi-output adaptive fuzzy/neural control', *IEEE Transactions on Fuzzy Systems* pp. 345–353.

Sebastien, Lesuer, D. Massicotte & P. Sicard (2001), 'Indirect inverse adaptative control based on neural networks using dynamic back propagation for nonlinear dynamic system', *IEEE International Symposium on Circuits and Systems* pp. 600–603.

Shim, H., T. J. Koo, F. Hoffmann & S. Sastry (1998), 'A comprehensive study of control design for an automous helicopter', *Proc. of 37th IEEE Conference on Decision and Control* pp. 3653–3658.

Silva, Ivan Nunes, Danilo Hernane Spattia & Rogerio Andrade Flauzino (2010), *Redes neurais artificiais*, Editora Artliber, SÃ£o Paulo, Brasil.

Taha, Zahari., Abdelhakim Deboucha & Mahidzal Bin Dahari (2010), 'Small-scale helicopter system identificaton model using recurrent neural networks', *IEEE Region 10 Conference* pp. 1393–1397.