MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-9878-TDI/874

AERONAVES CONFIGURADAS POR CONTROLE DO TIPO PREDITIVO NEURAL

Nei Cardoso Cardenuto

Tese de Doutorado em Computação Aplicada, orientada pelo Dr. Atair Rios Neto, aprovada em 14 de março de 2003.

INPE
São José dos Campos
2003

681.3.019

CARDENUTO, N. C.
Aeronaves configuradas por controle do tipo preditivo neural / N. C. Cardenuto. – São José dos Campos: INPE, 2003.
144p. – (INPE-9878-TDI/874).

1.Neural. 2.Controle de aeronave. 3.Predição. 4.Controle ótimo. 5.Controle de inclinação. I.Título.

Aprovado pela Banca Examinadora em cumprimento a requisito exigido para a obtenção do Título de **Doutor** em **Computação Aplicada.**

Dr. Fernando Manuel Ramos
Presidente/INPE-SJC/SP

Dr. Atair Rios Neto
Orientador/EMBRAER-SJC/SP

Dr. Reinaldo Roberto Rosa
Membro da Banca/INPE-SJC/SP

Dr. Valdemir Carrara
Membro da Banca/INPE-SJC/SP

Dr. Marcelo Curvo
Membro da Banca
Convidado EMBRAER, SJC/SP

Dr. Takashi Yoneyama
Membro da Banca
Convidado ITA/CTA, SJC/SP

Candidato (a): Nei Cardoso Cardenuto

São José dos Campos, 14 de março de 2003.

“Quando a Terra é avistada da Lua, não são visíveis, nela, as divisões em nações ou estados. Isso pode ser o símbolo da mitologia futura. Essa é a nação que iremos celebrar, essas são as pessoas às quais nos uniremos”

Joseph Campbell

A todos meus familiares, presentes, e que partiram
A nós piscianos, pela última passagem

AGRADECIMENTOS

Ao orientador Prof. Dr. Atair Rios Neto pelo conhecimento transmitido, e pela orientação, apoio e dedicação na realização deste trabalho.

Ao Instituto Nacional de Pesquisas Espaciais - INPE pela oportunidade de estudos e utilização de suas instalações.

Aos professores do LAC pelo conhecimento compartilhado, apoio e cooperação.

À profa. Dra. Sandra Sandri pelo apoio durante o curso.

À UNIVAP e à EMBRAER pelo apoio durante o programa.

Finalmente aos amigos do CVS e do “debôcho” que sempre me incentivaram.

RESUMO

Este trabalho desenvolve e testa uma abordagem de controle de aeronave em manobras não típicas conhecido como “control configured vehicle”, ou veículo (aeronave) configurado a controle, onde a abordagem proposta é controle preditivo neural. As manobras não típicas normalmente são obtidas desacoplando modos normalmente acoplados em dinâmica de vôo de aeronaves. A técnica mais usual é a atribuição de auto-estrutura, que trata de alocar simultaneamente autovalores e autovetores, sendo estes determinados de acordo com uma referência pré-estabelecida para o desacoplamento desejado. Na técnica de auto-estrutura, o sistema é linear e seu modelo é conhecido, limitando sua aplicação. A proposta de se utilizar controle preditivo com redes neurais visa obter ao mesmo desacoplamento sem a necessidade do conhecimento do modelo da planta, treinando a rede neural para modelar os modos dinâmicos da aeronave. O desempenho desta abordagem é demonstrado e os resultados são apresentados comparando-se: três redes treinadas, resultados de controle preditivo convencional e neural, gradiente analítico e numérico para a otimização, e horizontes de predição e controle.

AIRCRAFT CONFIGURED BY CONTROL

OF THE PREDICTIVE NEURAL TYPE

This work develops and tests a predictive neural control technique applied in the aircraft non-typical maneuvers known as the control configured vehicle (CCV). These maneuvers are performed with flight dynamics mode decoupling. The usual approach is the eigenstucture assignment with eigenvalues and eigenvectors placement, which is restricted to a known and linear systems model. The predictive control with neural nets approaches looks at the mode decoupling without the plant model knowledge by using neural net training in order to emulate the aircraft dynamic. The performance of the adopted approach is demonstrated and the results presented by considering and comparing: three neural nets, predictive control and neural predictive control, analytic and numeric gradient in the optimizing calculations, and prediction and control horizons.

SUMÁRIO

Pág.

LISTA DE FIGURAS

Pag.

LISTA DE SÍMBOLOS

Latinos

a	grandeza escalar
$\mathbf{a}$	grandeza vetorial
du	dimensão do vetor de controle
dy	dimensão do vetor de saída
E	erro entre padrão de saída e saída da rede
$\hat{\mathbf{e}}(k)$	erro de predição ou saída
$f'\{.\}$	derivada da função de ativação na camada "p"
$f(net)$	função de ativação
$\hat{f}$	função predição da rede neural
$\mathbf{I}_n$	matriz identidade
$I_i^E(\mathbf{u})$	parte do índice de desempenho relativa ao erro de predição
$J(.)$	Jacobiano
$J_i(.)$	função de desempenho para otimização
$\mathbf{J_T}yu_i$	matriz de Jacobianos total da rede propagada a partir do instante "i"
Nu	horizonte de controle
N2	horizonte de predição ou saída
p	"roll rate"
q	"pitch rate"
r	"yaw rate"
r	$\mathbf{r}(k)$ referência de saída no instante "k"
R	Matriz de ponderação dos erros de saídas no índice de desempenho
Ru	Matriz de ponderação dos controles no índice de desempenho
s_i^p	entrada da função de ativação do nó "i" da camada "p" da rede
$\mathbf{u}_i$	autovetor
$\mathbf{u}_i^{\mathbf{d}}$	autovetores desejados
$u(k)$	controle no instante "k"
U, V	matrizes modais
x_7	"washout"
x_i^p	saída do nó "i" da camada "p" da rede
$\hat{y}(.)$	saída estimada
$\hat{\mathbf{y}}(k)$	saída predita da rede no instante "k"
w	pesos sinápticos
$\mathbf{W}^p$	matriz de pesos sinápticos da camada "p"

w^{p}_{1,n_p} peso sináptico da entrada “1” para o nó “n_p” da camada “p” de dimensão n_y x n_u

W^{p} matriz de pesos da camada “p”

Gregos

α_c parâmetro de ajuste para aprendizado da rede
α ângulo de ataque
β ângulo de derrapagem (“sideslip”)
γ “path angle” – ângulo do vetor velocidade
δ_a deflexão do aileron
δ_{ac} comando de aileron
δ_e deflexão do profundor (“elevator”)
δ_{ec} comando de profundor (“elevator”)
δ_f deflexão do “flaperon”
δ_{fc} comando de “flaperon”
δ_{ij} delta de Kronecker
δ_r deflexão do leme (“rudder”)
δ_{rc} comando de leme (“rudder”)
ϕ ângulo de rolamento (“bank”)
Δ_j^p derivada parcial do erro
$?\mathbf{u}$ incremento de controle
λ_i autovalor
λ_i^d autovalores desejados
Λ matriz de autovalores
ν_j autovetor
ψ ângulo de guinada (“heading”)
$\xi(t)$ variável de modo
θ ângulo de arfagem (“pitch”)
∇E gradiente do erro
$\nabla^T_{\mathbf{u}(j)} I_U(i)$ gradiente da função desempenho relativa ao controle no instante “i”

LISTA DE SIGLAS E ABREVIATURAS

CCV	-	“Control Configured Vehicle”
SVD	-	“Single Value Decomposition”
NARMA	-	“Non-linear Auto Regressive Mean Average”
RBFN	-	“Radial Basis Function Network”
TAS	-	“True Airspeed”
BFGS	-	Broyden, Fletcher, Goldfarb e Shanno
USAF	-	United States Air Force

CAPÍTULO 1

INTRODUÇÃO

1.1 Considerações Iniciais

Síntese de sistemas de controle engloba uma variedade de técnicas consagradas, onde estabilidade, performance e robustez são os objetivos a serem atingidos. As técnicas são baseadas em algum modelo assumido como sistema real, modelos estes que partem de sistemas lineares até sistemas caóticos. Em sistemas cujos modelos são de complexidade elevada, com requisitos de performance arrojados, onde critérios de síntese estabelecidos não são ainda aplicáveis, técnicas heurísticas tornam-se relevantes. Entre estas, redes neurais (Nascimento e Takashi, 2000; Zurada, 1992), baseadas em sistemas conexionistas, são uma das opções escolhida neste trabalho.

Modelos baseados em redes neurais são arquiteturas de alta conectividade entre elementos simples, cuja dinâmica é inspirada em uma das funções supostamente atribuídas ao neurônio biológico. Modelos utilizados atualmente não correspondem ao funcionamento do cérebro humano, o qual ainda é complexo e relativamente desconhecido, porém a denominação redes neurais é apenas uma alusão a um modelo de neurônio elementar.

Sistemas multivariáveis podem ser modelados por sistemas lineares ou linearizados em torno de alguma região de operação, desde que algumas hipóteses sobre a fenomenologia do sistema sejam assumidas. Devida à característica não linear variante no tempo da dinâmica de vôo de aeronaves, seus modelos em geral são obtidos para situações de operação específicas e são parametrizados para cada nova situação. Desta forma, pode-se avaliar modelos lineares locais que correspondem a situações típicas de vôo e utilizar-se de técnicas consagradas para garantir requisitos de desempenho e margens de estabilidade reguladas por normas internacionais de certificação.

Aeronaves militares e demonstradoras, por sua vez, operam em limites e condições adversos e técnicas de controle mais arrojadas são implementadas com o objetivo de atingir desempenho superior, o qual é o requisito primordial no cenário operacional destas aeronaves.

Aeronaves civis, por outro lado, podem exibir situações críticas devido a falhas, e um controle não típico poderia apresentar alternativas para evitar a propagação até uma situação catastrófica.

Todas estas situações comportam a utilização de redes neurais: controle adaptativo não linear em aeronaves via redes neurais (Lee e Kim, 2001); controle ativo de vibrações em aeronaves civis com redes neurais (Snyder e Tanaka, 1995); predição de instabilidade aerodinâmica e controle (Faller e Schreck, 1995); determinação de parâmetros aerodinâmicos com redes neurais (Curvo, 2001; Gosh et al., 1998); controle preditivo neural de veículos aeroespacias (Rios Neto, 2000; Silva, 2001; Silva e Rios Neto, 2000).

Neste contexto, é relevante e atual se tentar controle neural para desacoplamento em manobras não típicas de aeronaves como uma alternativa ao formalismo de atribuição de auto-estrutura.

1.2 Objetivo

O objetivo deste trabalho é utilizar metodologia de controle preditivo com redes neurais (Phan e Xing, 1995; Omatu et al., 1996) para desacoplamentos de modos, em comparação ao método utilizado em síntese de controladores por atribuição de auto-estrutura em sistemas lineares (Sobel et al., 1994). Estratégias deste tipo são opções utilizadas em CCV para manobras não típicas e a utilização de redes neurais visa estabelecer um esquema alternativo para incorporar a hipótese de desconhecimento de modelo.

A contribuição deste trabalho é desenvolver e testar uma técnica de controle que permita o desacoplamento de modos de um sistema dinâmico, estabelecendo um horizonte de resposta similar ao obtido por desacoplamento, sem considerações sobre auto-estrutura do sistema. O modelo utilizado para avaliação é linear invariante no tempo e para objetivos de comparação é o mesmo utilizado nos trabalhos encontrados na literatura (Sobel e Shapiro, 1985; Sobel et al. 1994) sobre auto-estrutura de modo a se poderem comparar resultados.

CAPÍTULO 2

REVISÃO BIBLIOGRÁFICA

2.1 CCV – “Control Configured Vehicle"

A abertura de envelope operacional de aeronaves de combate permite o engajamento de alvos de forma mais efetiva e rápida, manobras de evasão típicas podem ser sobrepujadas por uma aeronave que possua um maior grau de instabilidade dinâmica intencional , onde controles extras produzem manobras rápidas e não típicas.

Com este objetivo, em torno de 1972 o protótipo YRF-4C do F-4E Phanton foi utilizado para testes do conceito de CCV e ficou conhecido como “Precision Aircraft Control Technology” (PACT). O programa da USAF Flight Dynamic Laboratory Control Configured Vehicle (CCV) construiu protótipos do YF-16 com “canards” (aletas dianteiras), modificações no tanque de combustível para permitir variação do centro de massa da aeronave para, junto com sistema de controle permitir manobras que não poderiam ser obtidas com controle convencional, onde movimentos em um plano, normalmente são relacionados com movimentos em outro plano, para mudar-se a proa da aeronave, necessita-se um rolamento da aeronave. No YF-16-CCV estes movimentos foram “desacoplados”; a aeronave pode subir ou descer via controle direto de sustentação, deslocar-se lateralmente sem mudança de proa, inclinação ou rolamento independente da direção de trajetória de vôo.

Outros exemplos são: o “Light Combat Aircraft” (LCA) da Índia, cujo desenvolvimento de 1983 a 1990, com o o primeiro “roll-out” em 1995; o LAVI israelense, o Mirage 2000 francês, e outros, todos foram concebidos para CCV.

O controle de aeronaves instáveis foi possível com o advento da tecnologia “Fly-By-Wire”, onde os comandos são eletrônicos e os atuadores são acionados por sinais digitais. Esta arquitetura possibilita o tratamento de sinais dos sensores e sinais de

controle rápidos o suficiente para manter a aeronave em diversas situações sob controle, o que uma pilotagem convencional não seria capaz de realizar.

Atualmente, existem outras abordagens para obter-se desacoplamento de modo:

- "Post Stall Technology" (PST)
- Empuxo vetorizado ("Thrust Vectorized")
- Configurações não convencionais.

As quais não serão abordadas neste trabalho.

2.2 Síntese por Atribuição de Auto-estrutura

Um das formas de obter-se desacoplamento de modos foi por atribuição de auto-estrutura (Klein e Moore, 1977; Srinathkumar, 1978; Porter e D'Azzo, 1978), onde condições suficientes foram obtidas para assegurar a atribuição de auto-estrutura dependendo da ordem do sistema, da dimensão dos espaços de observação e controle.

Aplicações surgem nos trabalhos de Sobel e Shapiro (Sobel e Shapiro, 1985) em sistemas de controle configurado (CCV) de aeronaves: modo de apontamento de inclinação ("pitch pointing"), o qual consta de comandar o ângulo de inclinação da aeronave sem que haja uma variação de altitude; e a contrapartida variação de altitude constante ("altitude rate") que é comandar uma variação de altitude mantendo o ângulo de inclinação constante.

Outras referências abordam este assunto: um procedimento para controle CCV (configured control vehicle) em "pitch pointing" utiliza atribuição de auto-estrutura (Sobel e Shapiro, 1985); supressão de "flutter" é abordado com esta metodologia (Liebst et al., 1986); um trabalho mais detalhado sobre a área (Sobel et al., 1994) estende aplicações envolvendo restrições de observabilidade e sistemas discretos; um mesmo problema ("pitch pointing") é tratado (Siouris et al., 1995) com ênfase em robustez.

2.3 Redes Neurais

Redes Neurais, cujos princípios já foram estabelecidos há décadas, recentemente emergiram como alternativa a controle de sistemas complexos preferencialmente não lineares e sistemas onde existe pouco conhecimento sobre o modelo (Hunt et al., 1992). Apesar da característica heurística de suas aplicações, avanços no formalismo têm produzido uma melhor compreensão dos princípios teóricos envolvidos.

Vale ressaltar que a base teórica tem sido a principal causa dos obstáculos encontrados nos trabalhos com redes neurais artificiais. A hierarquia de abordagem para sistemas dinâmicos define (Narendra, 1996) redes neurais como uma técnica a ser utilizada quando já não há possibilidade de aplicação das técnicas de análise e projeto consagradas.

Redes neurais, como outros sistemas baseados em conhecimento, possuem a capacidade de mapear relações funcionais complexas quando submetidas a um treinamento sobre o conjunto de entrada/saída desejado (Hunt et al., 1992). Sistemas caóticos têm sido reproduzidos e vários trabalhos neste sentido estão presentes atualmente (Otawara e Fan, 1996). A simplicidade celular de sua implementação torna atraente o uso de redes neurais, quando o aspecto de paralelismo e velocidade de processamento é observado. Outrossim, a implementação em “firmware” é um atrativo de grande relevância.

Aplicações aeroespaciais normalmente introduzem redes neurais para controle adaptativo (Lee e Kim, 2001), redes neurais para controle e emulador (Gili e Battipede, 2001), onde a aplicação é típica e um tratamento para estabilidade local é desenvolvido; em resumo, os trabalhos na área aeroespacial mantêm um vínculo com o tratamento formal, sempre com o objetivo de assegurar um grau de estabilidade e robustez consistente, porém, em se tratando de redes neurais, atualmente, somente se pode assegurar comportamento local.

CAPÍTULO 3

FUNDAMENTOS: REDES NEURAIS E CONTROLE NEURAL

3.1 Redes Neurais Artificiais

Redes Neurais podem ser definidas como a interconexão de neurônios, onde a arquitetura de conexões e a função de ativação podem conter uma variedade voltada às características do sistema que se pretende representar, desde neurônios binários simples para classificadores como hipercubos de "Hamming" (Zurada, 1992) até sistemas dinâmicos não lineares com comportamento caótico.

As redes neurais, apesar de suas variações, podem ser classificadas de forma geral em duas arquiteturas: direta ("feedforward") , onde as entradas da rede propagam-se até a saída, e recorrentes ("feedback"), onde as saídas da rede são realimentadas na entrada. Desde que o processamento discreto introduz um elemento de atraso, obtém-se assim um sistema de primeira ordem e na medida que se multiplicam as conexões, esses atrasos em cascata produzem sistemas dinâmicos de ordem superior, aumentando a capacidade de representação ao mesmo tempo em que a não linearidade e estabilidade tornam-se críticas, surgindo assim uma variedade de pontos de equilíbrio locais, dificultando uma formalização teórica consistente. As redes recorrentes ("feedback net") são utilizadas para memória associativas bem como para solução de problemas de otimização (Narendra e Parthasarathy, 1990).

Do ponto de vista de sistemas, as redes multicamadas diretas representam mapeamentos estáticos, enquanto as recorrentes representam sistemas dinâmicos. Ainda, as redes recorrentes, possuindo dinâmica interna devido à retroação, normalmente utilizam poucos neurônios e camada única com a finalidade de limitar a ordem do sistema.

Para um modelo mais completo, a tendência é a unificação dos conceitos para obter maior capacidade de representação de sistemas complexos (Narendra e Parthasarathy, 1990).

3.2 Regras de Aprendizado de Redes Neurais

O neurônio é considerado um elemento adaptativo, seus pesos são passíveis de alteração como resultado do par "entrada / saída desejada", se a informação do erro de saída não é utilizada, o aprendizado é denominado não-supervisionado, caso contrário, conseqüentemente, supervisionado.

A regra geral de aprendizado é um problema inverso, desde que, por regressão ou alguma variedade de procedimento, procura-se através das observações das saídas, obter-se uma função que mapeie o espaço de entradas no espaço de saída.

A estrutura básica está ilustrada na Figura 3.1, onde o sinal de aprendizado $r = r(w_i, x, d_i)$, e o incremento de correção dos pesos é obtido como:

$$\Delta \mathbf{w_i}(t) = cr(\mathbf{w_i}(t), \mathbf{x}(\mathbf{t}), d_i(t)) \cdot .\mathbf{x}(t) \qquad (3.1)$$

discretizando:

$$\mathbf{w}_i^{k+1} = \mathbf{w}_i^k + cr(\mathbf{w}_i^k, \mathbf{x}^k, d_i^k) . \cdot \mathbf{x}^k \qquad (3.2)$$

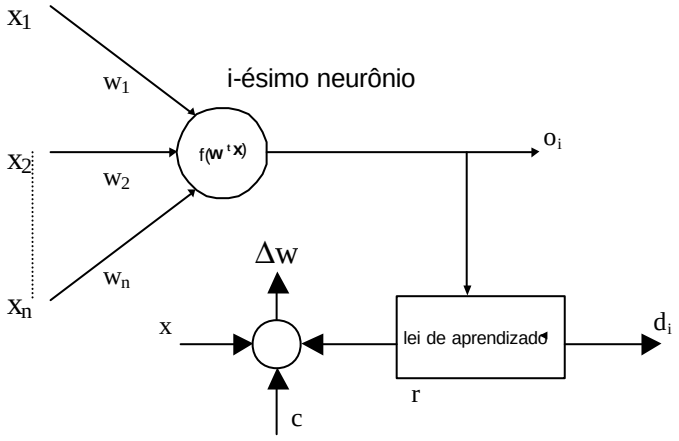


FIGURA 3.1 - Ilustração de regras de aprendizado.

3.2.1 Regra Delta Generalizada: Retropropagação em Redes Multicamadas

Este procedimento de treinamento é o mais comum para redes "feedforward" multicamadas e é o utilizado neste trabalho. Seja a Figura 3.2, representativa de uma rede típica multicamada.

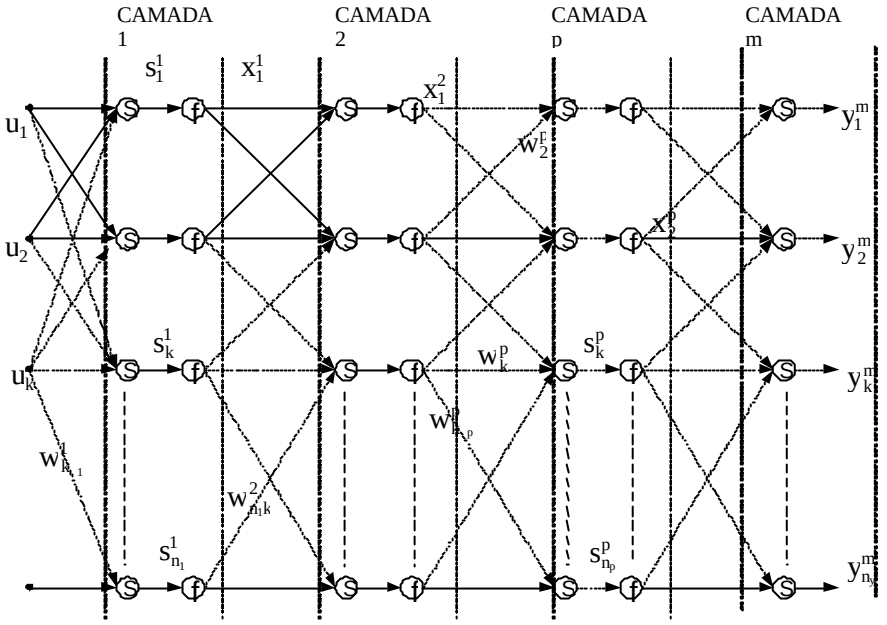


FIGURA. 3.2– Rede Multicamada "feedforward".

Para uma saída qualquer x_i^p em uma camada intermediaria p qualquer:

$$x_i^p = f(s_i^p) \tag{3.3}$$

$$s_i^p = \sum_{j=1}^{n_{p-1}} x_j^{p-1} \cdot w_{ji}^p \tag{3.4}$$

Calculando as saídas para as camadas “1”, “2”, camada genérica “p” e camada de saída “m”:

Camada “1”:

$$s_i^1 = \sum_{k=1}^{n_u} u_k \cdot w_{ki}^1 \quad , \quad i = 1,2,.....,n_1 \tag{3.5}$$

$$x_i^1 = f(s_i^1), \quad i = 1,2,.....,n_1 \tag{3.6}$$

Camada “2”:

$$s_i^2 = \sum_{k=1}^{n_1} x_k^1 \cdot w_{ki}^2 \quad , \quad i = 1,2,.....,n_2 \tag{3.7}$$

$$x_i^2 = f(s_i^2) \quad , \quad i = 1,2,.....,n_2 \tag{3.8}$$

Camada “p”:

$$s_i^p = \sum_{k=1}^{n_{p-1}} x_k^{p-1} \cdot w_{ki}^p \quad , \quad i = 1,2,.....,n_p \tag{3.9}$$

$$x_i^p = f(s_i^p) \quad , \quad i = 1,2,.....,n_p \tag{3.10}$$

Camada de saída “m”:

$$y_i^m = \sum_{k=1}^{n_{m-1}} x_k^{m-1} \cdot w_{ki}^m \quad , \quad i = 1,2,.....,n_y \tag{3.11}$$

O objetivo do treinamento é, a partir de um padrão desejado de saída (Equação 3.12), calcular o valor dos pesos "w" em todas as camadas no sentido de minimizar o erro quadrático "E" (Equação 3.13):

$$d = [d_1, d_2,.....d_{n_y}] \tag{3.12}$$

$$E = \frac{1}{2}\sum_{k=1}^{n_y}(d_k - y_k^m)^2 = \frac{1}{2}\left\| d - y^m \right\|^2 \tag{3.13}$$

O procedimento é retropropagar o erro na saída ("backpropagation") ajustando os pesos em cada camada ao longo da rede até a entrada. O critério é primeiramente determinar a sensitividade do erro total em relação à rede, o que significa obter-se o gradiente "∇E"em relação aos pesos, via regra da cadeia, e corrigir os pesos seqüencialmente da saída até a entrada de acordo com o procedimento abaixo:

Partindo-se da saída, calculando-se o gradiente, a Equação 3.14 é válida em todas as camadas e o gradiente (Equação 3.16) é determinado somente na camada de saída:

$$\frac{\partial E}{\partial w_{ij}^m} = \frac{\partial E}{\partial y_j^m} * \frac{\partial y_j^m}{\partial s_j^m} * \frac{\partial s_j^m}{\partial w_{ij}^m} \tag{3.14}$$

$$\frac{\partial E}{\partial y_j^m} = y_j^m - d_j^m \tag{3.15}$$

e a partir da Equação 3.20, tem-se:

$$\frac{\partial y_j^m}{\partial s_j^m} = f'(\mathbf{s}^m) = 1 \tag{3.16}$$

conhecido a partir da função de transferência dos neurônios da camada "m", que é a identidade, e das Equações 3.3 e 3.4 para a camada "$m-1$", tem-se:

$$s_i^m = \sum_{k=1}^{n_m} x_k^{m-1} \cdot w_{ki}^m \quad , \quad i = 1,2,.....,n_m \tag{3.17}$$

$$x_i^m = f(s_i^m) = s_i^m = y_1^m, \quad i = 1,2,.....,n_m \tag{3.18}$$

Assim, o gradiente na camada de saída "m " fica:

$$\frac{\partial E}{\partial w_{ij}^m} = (y_j^m - d_j^m) * f'(s^m) * x_i^{m-1} \tag{3.19}$$

Para o cálculo do gradiente para a camada "$m-1$" em diante a alteração dá-se no cálculo de "$\frac{\partial E}{\partial y_j^{m-1}}$".

Uma entrada "x_j" da camada "m" oriunda da saída "j" da camada "m-1" entra em todos os neurônios da camada de saída "m" , portanto a sensitividade do erro à esta saída leva em conta todos os neurônios da camada "m", logo :

$$\frac{\partial E}{\partial x_j^{m-1}} = \sum_{l=1}^{n_m} \frac{\partial E}{\partial y_l^m} * \frac{\partial y_l^m}{\partial s_l^m} * \frac{\partial s_l^m}{\partial x_j^{m-1}} \tag{3.20}$$

onde, observando-se a Equação 3.4 tem-se :

$$\frac{\partial s_l^m}{\partial x_j^{m-1}} = w_{jl}^m \tag{3.21}$$

e assim pode-se resumir o algoritmo da seguinte forma:

Sendo $\Delta_j^p = \frac{\partial E}{\partial y_j^p}$, pode-se escrever então:

$$\Delta_j^{p-1} = \sum_{l=1}^{n_p} \Delta_l^p * f'(s_l^p) * w_{jl}^p \qquad (3.22)$$

e o gradiente fica:

$$\frac{\partial E}{\partial w_{ij}^p} = \Delta_j^p * f'(s_j^p) * x_i^{p-1} \qquad (3.23)$$

p - é a camada genérica

j - é a entrada "j"

n_p - número de neurônios da camada "p"

e a atualização dos pesos é realizada da forma (representação matricial):

$$\mathbf{W}^p = \mathbf{W}^p - \alpha_c \nabla E^p \qquad (3.24)$$

onde "α_c" é um parâmetro de convergência.

3.3 Controle com Redes Neurais

Controle de sistemas dinâmicos não-lineares com redes neurais foi proposto primeiramente no final da década de 80 (Hunt et al., 1992). Técnicas de análise e projeto de sistemas de controle são bem consagradas quando se trata de sistemas lineares, porém para sistemas não lineares, um conjunto de conjecturas é reunido para chegar-se a algum método que forneça um embasamento teórico consistente.

Redes neurais, a princípio, foram exploradas sem base em algum critério teórico consagrado, sendo um método puramente experimental que resolvia a problemática de

modelagem de sistemas não-lineares. Resultados teóricos têm evoluído nas últimas duas décadas e um avanço incremental é notado.

Entretanto, apesar de todo o conjunto de progressos formais, os modelos matemáticos envolvidos partem de um princípio básico, ou seja, faz-se necessário um conhecimento fenomenológico do sistema a ser modelado e, outrossim, uma avaliação a priori das fontes de incertezas, seja na dinâmica do próprio fenômeno, na sua observação, ou na própria estrutura admitida como modelo e assim, as ferramentas, com base teórica bem sustentada, são plenamente aplicáveis sobre modelos bem adequados e comportados, como sistemas lineares, linearizados e não lineares bem específicos, onde funções auxiliares (funções de Liapunov, funções descritivas) já são bem exploradas. Neste aspecto, quando da presença de sistemas não lineares, por exemplo, onde a determinação de um modelo já possui uma complexidade elevada ou um grau razoável de desconhecimento, e ainda, o formalismo para análise de desempenho, otimização e robustez não são suficientes para determinar uma lei de controle aceitável, a busca de soluções caminha na direção de sistemas heurísticos, baseados em regras de inferência, em decisões baseadas em lógica nebulosa, em redes neurais, etc. Algoritmos estes, onde há necessidade de um conhecimento adquirido pela experiência anterior, ou seja, a memória histórica é a principal fonte de informação (Omatu et al., 1995).

3.3.1 Controle Baseado em Entradas e Saídas

A representação de um sistema dinâmico por variáveis de estado adequadas aos métodos teóricos utilizados, nem sempre está acessível. A observação da dinâmica dos sistemas, em geral, é realizada externamente, assim, a relação entre as entradas e saídas é a forma fenomenológica observada dos sistemas.

Sistemas lineares invariantes no tempo possuem correspondência entre representação por variáveis de estado e representação por relação entre entrada e saída, ou função de transferência, porem em sistemas não lineares, esta correspondência é mais complexa de

estabelecer. Nesta representação (I/O), a importância de neuro-controle torna-se relevante.

3.3.2 Identificação e Redes Neurais

Identificação de sistemas pode tratar de sistemas estáticos ou dinâmicos. Reconhecimento de padrões é um típico problema de identificação de parâmetros em sistemas estáticos; identificação de sistemas dinâmicos abrange a identificação de função, assumindo-se uma caixa preta ou uma aproximação funcional adequada.

Tratando-se de identificação de função, sistemas dinâmicos podem ser descritos por dois tipos de modelos: modelo de entrada-saída e modelo de espaço de estados.

O modelo de entrada-saída descreve um sistema dinâmico baseado em dados de entradas e saídas do sistema. O modelo assume que um novo conjunto de saída do sistema pode ser previsto a partir de entradas e saídas anteriores.

O modelo descrito pelo espaço de estados possui a dinâmica representada pelas Equações 3.25 a 3.32.

Na forma contínua:

$$\dot{x}(t) = f(x(t), u(t)) \qquad (3.25)$$

$$y(t) = h(x(t)) \qquad (3.26)$$

Linearizada contínua em torno da trajetória nominal:

$$\dot{x}(t) = Ax(t) + Bu(t) \qquad (3.27)$$

$$y(t) = Cx(t) \qquad (3.28)$$

Discreta não linear:

$$x(k+1) = \Phi[x(k), u(k)] \tag{3.29}$$

$$y(t) = \Psi[x(k)] \tag{3.30}$$

Discreta linear:

$$x(k+1) = Ax(k) + Bu(k) \tag{3.31}$$

$$y(k) = Cx(k) \tag{3.32}$$

O problema de identificação está associado com o desconhecimento de h e f em sistemas não lineares, discretizados nas matrizes Φ e Ψ; e nas matrizes A, B e C nos modelos linearizados.

Se a entrada u(.) é limitada ao longo do tempo e saída y(.) também, a planta é assumida estável com uma parametrização conhecida, mas com seus valores (parâmetros) desconhecidos. O objetivo é construir um modelo (Figura 3.3) de identificação o qual, quando sujeito à mesma entrada $u(k)$ da planta, produz uma saída $\hat{y}_p(k)$ que é uma aproximação de $y_p(k)$.

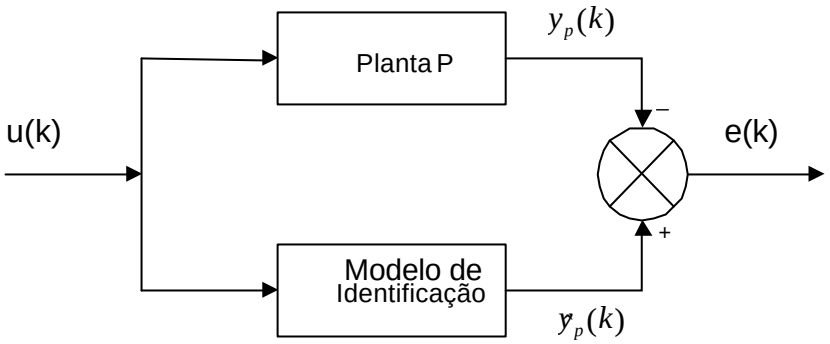


FIGURA. 3.3- Identificação.

Descrito de outra forma pode-se dizer sobre identificação (Narendra, 1996): para identificar uma planta não-linear, deve-se assumir que o sistema é BIBO estável ("bounded-input bounded-output").

A premissa anterior permite que a identificação seja realizada "off-line" ou "on-line". Se a planta for instável, identificação e controle devem ser realizados simultaneamente "on-line", com controle adaptativo. Provar teoricamente a estabilidade em sistemas adaptativos é uma tarefa de difícil solução e em casos bem particulares (Narendra, 1996). Considere-se o modelo entrada-saída de um sistema dinâmico caracterizado como:

$$\begin{aligned} y_p(k+1) = f[&y_p(k), y_p(k-1), ..., y_p(k-n+1); \\ &u(k), u(k-1), ..., u(k-m+1)] \end{aligned} \tag{3.33}$$

onde $f : R^{dy*n+du*m} \rightarrow R^{dy}$ é assumida função diferenciável de seus argumentos, onde a caracterização do modelo é dada por:

$$\begin{aligned} \hat{y}_p(k+1) = \hat{f}[&\hat{y}_p(k), \hat{y}_p(k-1), ..., \hat{y}_p(k-n+1); \\ &u(k), u(k-1), ..., u(k-m+1)] \end{aligned} \tag{3.34}$$

o qual é denominado estrutura de identificação paralelo (Narendra e Parthasarathy, 1990), onde os valores anteriores da saída do próprio modelo são utilizados, normalmente quando em regime permanente presume-se que a saída do sistema seja aproximadamente a saída do modelo ($y_p \cong \hat{y}_p$), tornando-o assim, independente da planta.

A esta estrutura de modelo, denomina-se NARMA ("Non-linear autoregressive moving average"). Quando as próprias saídas do modelo alimentam a rede denomina-se estrutura paralela e quando as saídas do sistema alimentam o modelo denomina-se série-paralela.

3.3.3 Arquiteturas de Controle com Redes Neurais

Há várias situações onde a ação do operador humano se faz necessária devido à dificuldade de sintetizar um controlador dentro do formalismo que garanta os requisitos desejados. No sentido de imitar o operador humano, controladores são projetados, daí a denominação “supervisionados”. Dentre os métodos utilizados, sistemas baseados em conhecimento, lógica generalizada, e redes neurais pertencem a esta categoria.

Vários esquemas de neuro-controle (Omatu et al., 1996) baseados em algoritmos de retropropagação (“backpropagation”) são consagrados e em geral os esquemas são baseados nas seguintes abordagens:

- Controle série: a rede é treinada para mapear o modelo inverso, ou seja, do sinal desejado (entrada) para o controle (saída).

- Controle paralelo: a rede é utilizada em paralelo a um esquema de controle convencional para modelar não linearidades ou desconhecimento do modelo.

- Controle auto-sintonizado: a rede sintoniza os parâmetros de controle de um controlador convencional (parâmetros de PID por exemplo).

- Emulador-controlador: da classe de modelo interno, a retropropagação do erro de saída é realizada através do emulador (identificador da planta) e do controlador. Também utilizado para fornecer o Jacobiano da planta em um esquema de controle convencional adaptativo ótimo.

3.3.3.1 Controle Direto (“feedforward”) Inverso

A consagração do algoritmo de retropropagação (“backpropagation”) para treinamento de redes fez surgir diversas arquiteturas para neuro controle, e este esquema é o mais popular devido à sua simplicidade (Omatu et al.,1996). A rede é treinada no modelo inverso da planta e utilizada no caminho direto (“feedforward control”) para gerar o

sinal de controle para a planta. Há dois esquemas básicos nesta arquitetura determinando a forma de treinamento da rede (Psaltis et al.,1988): generalizado e especializado.

No esquema generalizado da Figura 3.4, a rede é treinada "off-line" usando-se padrões de treinamento obtidos da planta em malha aberta ou malha fechada, onde o treinamento é realizado em "batch", similar ao aplicado em classificação e reconhecimento de padrões. A rede treinada é utilizada como um controlador "feedforward" (direto) em uma arquitetura de realimentação típica.

No esquema especializado da Figura 3.5, a rede é treinada "on-line" onde o erro entre a referência e a saída é retropropagado através da rede a cada amostra, onde neste caso o Jacobiano da planta é necessário para retropropagação do erro pela planta até a saída do neuro-controlador.

Nesta configuração a rede é treinada "off-line" para aprender o modelo inverso da planta e em seguida utilizada para gerar o controle.

Algumas limitações podem surgir devido à robustez destes sistemas onde a fidelidade do controlador torna-se crítica; um ajuste "on-line" pode auxiliar a diminuir esta limitação.

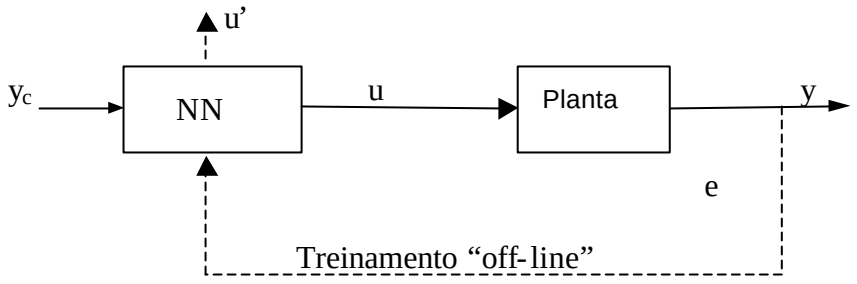


FIGURA. 3.4 - Arquitetura para Controle Direto Inverso Generalizado.

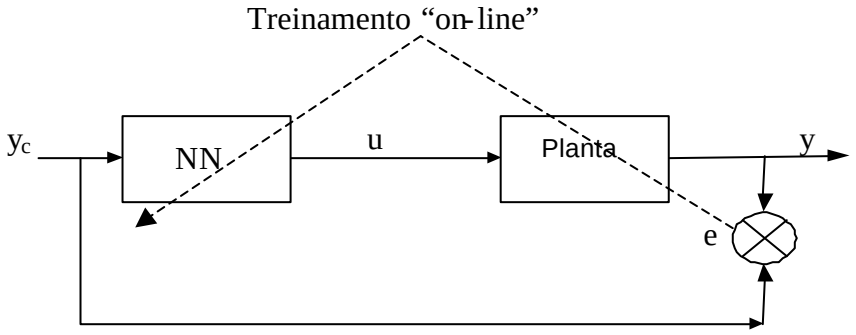


FIGURA. 3.5 - Arquitetura para Controle Direto Inverso Especializado.

3.3.3.2 Controle Retroalimentado (“feedback”) Inverso

Neste esquema de aprendizado “on-line” (Nascimento, 1994; Kawato et al., 1988) da Figura 3.6, a rede neural é configurada em paralelo ao controlador convencional na realimentação e é treinada por retropropagação do erro de realimentação, na medida em que a rede aprende o modelo inverso da planta, o controlador convencional vai diminuindo sua significância até ser eliminado da malha. Este é um caso semelhante ao aprendizado humano, muito utilizado em robótica.

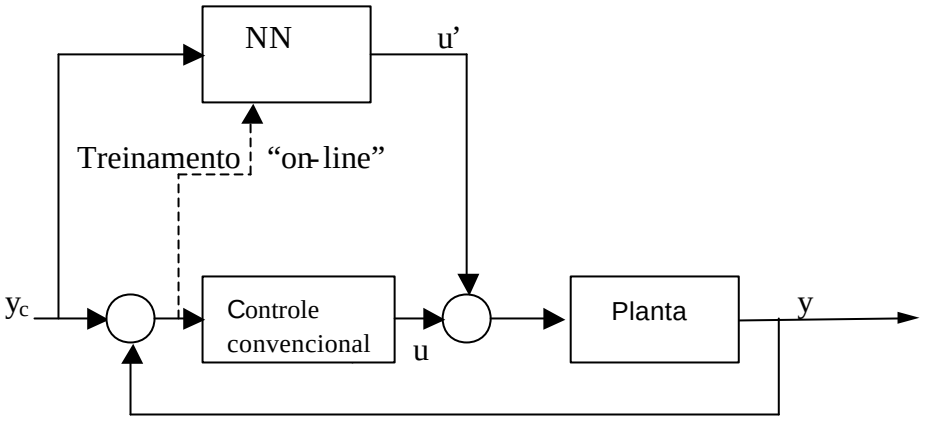


FIGURA. 3.6 -- Arquitetura para Controle Retroalimentado (“feedback”) Inverso.

3.3.3.3 Emulador-Controlador

Neste esquema duas redes neurais são utilizadas:

- Emulador, podendo ser treinado “off-line” no modelo direto da planta ou “on-line” injetando-se entradas aleatórias dentro do espaço de controle.

- Controlador, treinado “on-line” no modelo inverso, gerando as entradas de controle para a planta. Esta arquitetura é consagrada para a aplicação em sistemas não-lineares onde o modelo da planta é pouco conhecido. O emulador depois de treinado provê o Jacobiano para o método de otimização (predição) e a retropropagação do erro de saída pelo emulador gera o erro de controle para a retropropagação do neuro-controlador.

Este método é denominado como “retropropagação contínua no tempo” (“backpropagation-through-time”) (Narendra e Parthasarathy, 1990; Werbos, 1990).

A Figura 3.7 mostra o esquema para a determinação do controlador aplicado neste trabalho, onde a implementação envolve uma abordagem com otimização.

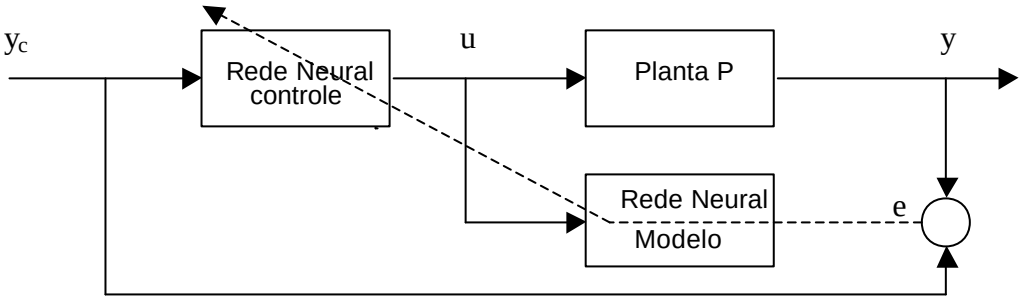


FIGURA. 3.7 - Arquitetura de Modelo Interno.

3.3.4 Controle Preditivo

Controle preditivo é uma abordagem bem adequada à utilização de redes neurais (Phan e Xing, 1995). As etapas constam da identificação do sistema dinâmico, que corresponde ao treinamento de um modelo NARMA implementado por arquitetura de redes neurais e da determinação da lei de controle, normalmente baseada em técnicas de otimização sobre o modelo e seu Jacobiano analítico ou numérico extraído da rede

neural treinada (Zhao et al., 2001; Zhan e Ishida, 1997; Sorensen et al., 1999). Controle preditivo neural tem sido utilizado em processos industriais em geral (Gomm et al., 1997; Zamarreño e Veja, 1999; Turner et al., 1996; Botto e Costa, 1998), onde as dinâmicas são mais lentas e a complexidade dos processos envolvidos limita a elaboração e performance de modelos.

A técnica de horizonte de controle em controle ótimo e preditivo tem sido introduzida como lei de controle computacionalmente possível e natural. Outrossim, o método possui características de desempenho e estabilidade apresentadas em diversas aplicações em sistemas lineares variantes no tempo e não lineares (Mayne e Michalska, 1990).

Neste método, a rede neural fornece a predição da resposta futura da planta sobre o horizonte especificado. As predições obtidas pela rede são processadas por rotinas de otimização sobre o critério quadrático a seguir:

$$J = \sum_{j=N1}^{N2} [y(t_j) - \hat{y}(t_j)]^T R_y^{-1}(t_j) [y(t_j) - \hat{y}(t_j)] + \frac{1}{2} \sum_{j=1}^{Nu} [u(t_j) - u(t_{j-1})]^T R_u^{-1}(t_j) [u(t_j) - u(t_{j-1})] \tag{3.35}$$

com vínculos $|u| \leq u_{max}$ e $|y| \leq y_{max}$.

onde o sinal de controle u é escolhido para minimizar o critério quadrático acima sujeito aos vínculos do modelo dinâmico. A arquitetura está ilustrada na Figura 3.8.

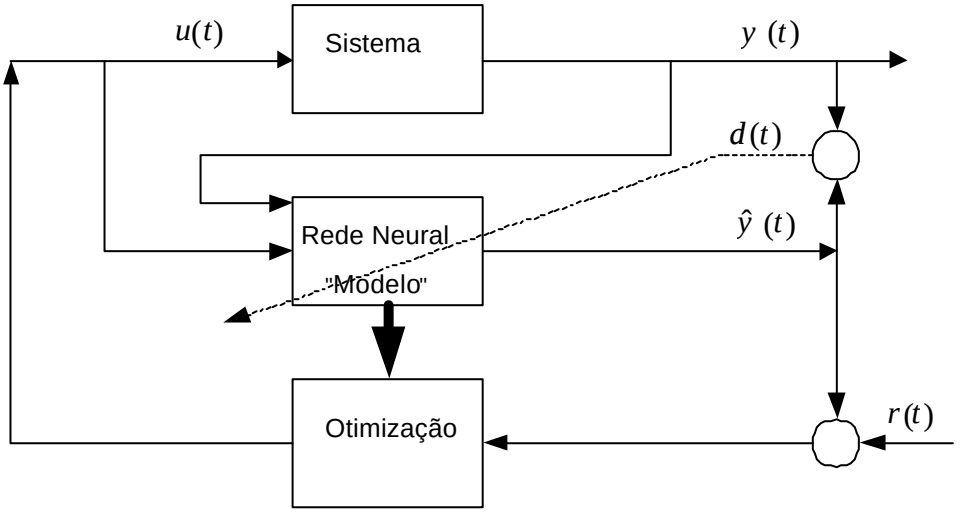


FIGURA. 3.8 - Arquitetura de Controle Preditivo.

As constantes N1 e N2 definem o horizonte sobre o qual o erro e os incrementos de controle são considerados, a constante Nu é o horizonte de controle e os valores de R_y e R_u são as ponderações. A vantagem desta arquitetura é que a malha externa de otimização passa a ser desnecessário depois de completado o aprendizado da rede. A solução para $\hat{y}_p(k+1)$ fica :

$$\hat{y}_p(k+1) = \hat{f}[y_p(k), y_p(k-1), ..., y_p(k-n+1); \\ u(k), u(k-1), ..., u(k-m+1), \hat{w}] \qquad (3.36)$$

onde $\hat{w}$ são os pesos estimados da rede neural $\hat{f}$ que modela o sistema.

CAPÍTULO 4

CONTROLE PREDITIVO EM MODO “PITCH POINTING”

4.1 Introdução

A proposta de utilizar controle preditivo visa seguir uma referência de saída pré-estabelecida, onde o desacoplamento será obtido estabelecendo um horizonte de saída cujas componentes estejam desacopladas, por exemplo, após a aeronave atingir um ângulo de arfagem (“pitch”) positivo, em condições típicas de vôo horizontal, a conseqüência é que sua altitude se eleve; estabelecendo-se como saídas desejadas a altitude constante e um ângulo determinado, consegue-se um desacoplamento destes modos.

Antes de se utilizar controle preditivo neural, verifica-se antes a viabilidade da abordagem de controle preditivo convencional para a obtenção do desacoplamento de modos para a dinâmica linear longitudinal da aeronave.

A avaliação será realizada via modelo de aeronave em modo de operação “pitch pointing/ altitude rate”, que se trata de um comando para arfagem constante sem que a aeronave altere sua altitude, e a contrapartida, dado um comando de velocidade vertical a aeronave mantém a arfagem nula.

Estas manobras possuem importância em pontaria de armas, onde uma pequena alteração de ângulo e/ou posição pode englobar rapidamente o alvo no envelope de tiro do armamento.

4.2 Esquema de Controle Preditivo

A princípio, com a finalidade de ilustrar o conceito mencionado, de utilização de controle preditivo para desacoplamento, os modelos lineares utilizados estão

referenciados nos trabalhos afins (Andry et al., 1983 e Sobel et al.,1985) onde a otimização será da forma descrita anteriormente.

A idéia básica é fazer uma alocação de pólos trivial, com a finalidade de obter os autovalores desejados, porém sem o formalismo de alocação de autovetores como realizado nos trabalhos mencionados. A partir da estrutura mostrada na Figura 4.1, o controlador é determinado baseando-se no modelo de identificação tomado da própria planta do sistema. Os resultados obtidos foram satisfatórios do ponto de vista de resposta estável, tempo de acomodação e erro nas componentes de saída, concluindo-se que o formalismo de controle preditivo é aplicável nesta classe de problemas.

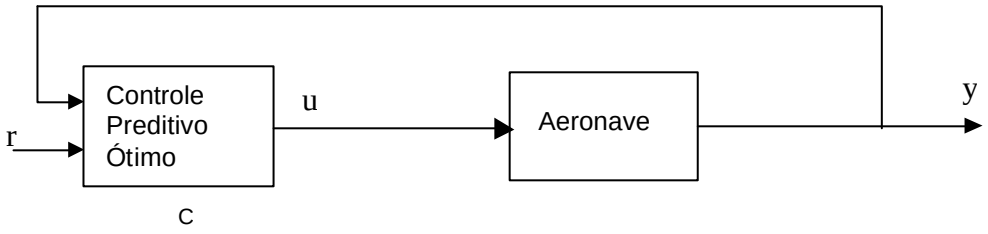


FIGURA. 4.1 – Arquitetura de Controle Preditivo em CCV.

4.3 Aplicação do Controle Preditivo Convencional

A aplicação deste esquema foi realizada com o objetivo de demonstração de conceito em alternativa ao método de atribuição de auto-estrutura para desacoplamento (Apêndice A).

4.3.1 Definição do problema

Os modelos utilizados são lineares correspondendo à dinâmica longitudinal já estabilizadas via alocação trivial de autovalores (função “PLACE” do “Matlab Control Toolbox”), sem critério de atribuição de autovetores, ou seja, os modos estão estáveis porém acoplados.

A formulação do problema de atribuição de auto-estrutura foi elaborada com o objetivo de utilizar a função “SCMPC” do “Matlab Model Predictive Control Toolbox”. O modelo de identificação e o algoritmo de otimização interno da função utilizam programação quadrática com restrições de saída e controle. O algoritmo calcula o erro internamente na forma de índice de desempenho quadrático das saídas de referência desejadas e saídas simuladas do sistema

Os resultados obtidos não foram sensíveis com variações nos horizontes de saída N2 e de controle Nu, isto se deve ao fato do controlador ser o modelo inverso ideal, pois se utilizou o modelo nominal como modelo de identificação.

4.3.2 Dinâmica Longitudinal

Este exemplo é baseado no modelo da dinâmica longitudinal local simplificada da aeronave F-16 em altitude de 3000 pés, e velocidade 0.6 mach (Andry et al., 1983 e Sobel et al.,1985). A Figura 4.2 mostra as grandezas mencionadas na formulação em relação à aeronave.

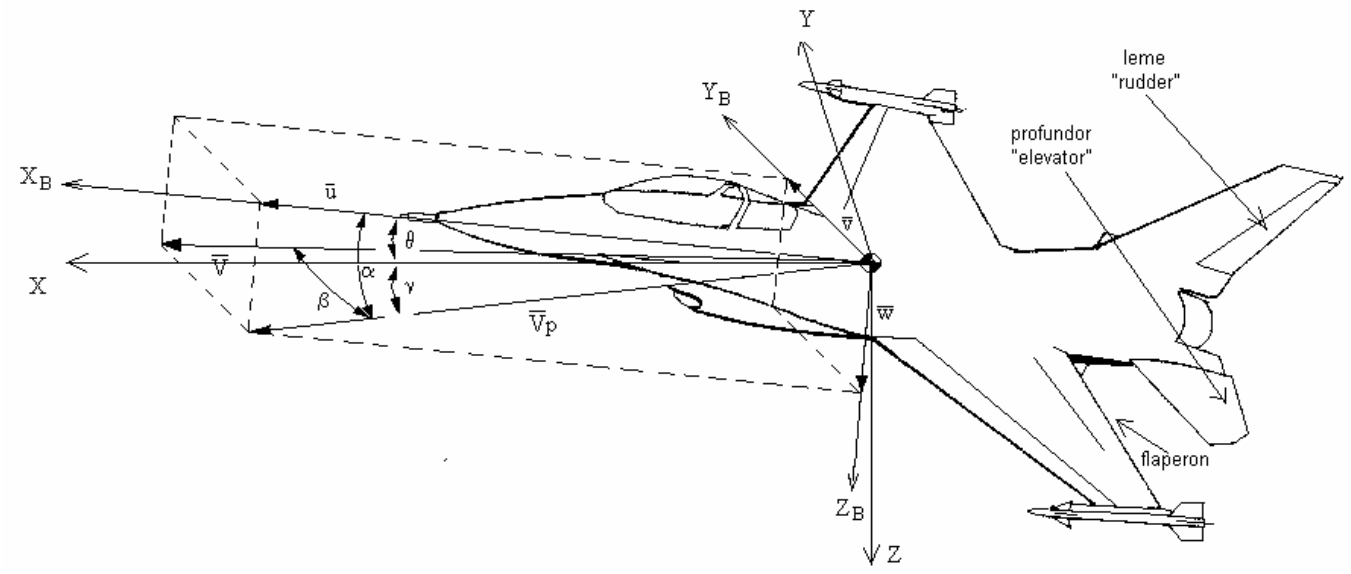


Fig. 4.2 – grandezas e suas referências na aeronave.

As equações lineares são:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{4.1}$$

onde o estado é definido como:

$$x = [\theta \quad q \quad \alpha \quad \delta_e \quad \delta_f]^T \tag{4.2}$$

onde:

θ – ângulo de "pitch"
q – "pitch rate"
α – ângulo de ataque
δ_e – ângulo de "elevator"
δ_f – ângulo de flaperon

o controle definido como:

$$u = \begin{bmatrix} \delta_{ec} \\ \delta_{fc} \end{bmatrix} \tag{4.3}$$

onde:

δ_{ec} – "elevator command"
δ_{fc} – "flaperon command"

As equações linearizadas, na condição de regime do vôo, são funções de derivadas aerodinâmicas, forças e momentos aerodinâmicos, e dinâmica de atuadores:

$$\dot{q} = f_q(q, \alpha, \delta_e, \delta_f)$$
$$\dot{\alpha} = f_\alpha(q, \alpha, \delta_e, \delta_f)$$
$$\dot{\delta}_e = f_e(\delta_e, \delta_{ec})$$
$$\dot{\delta}_f = f_f(\delta_f, \delta_{fc})$$

$$
\begin{aligned}
\dot{\theta} &= q \\
\dot{q} &= -0.87 \cdot q + 43.22 \cdot \alpha - 17.25 \cdot \delta_e - 1.58 \cdot \delta_f \\
\dot{\alpha} &= -0.99 \cdot q + 1.34 \cdot \alpha - 0.17 \cdot \delta_e - 0.25 \cdot \delta_f \\
\dot{\delta}_e &= -20 \cdot (\delta_e - \delta_{ec}) \\
\dot{\delta}_f &= -20 \cdot (\delta_f - \delta_{fc})
\end{aligned}
\tag{4.4}
$$

A matriz de sistema fica da forma:

$$
A = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 \\
0 & -0.87 & 43.22 & -17.25 & -1.58 \\
0 & 0.99 & -1.34 & -0.17 & -0.25 \\
0 & 0 & 0 & -20 & 0 \\
0 & 0 & 0 & 0 & -20
\end{bmatrix}
\tag{4.5}
$$

A matriz de controle:

$$
B = \begin{bmatrix}
0 & 0 & 0 & 20 & 0 \\
0 & 0 & 0 & 0 & 20
\end{bmatrix}^T
\tag{4.6}
$$

e a matriz de saída como:

$$
C = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
1 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\tag{4.7}
$$

Os autovalores de malha aberta são:

$$
? = [-7.66 \quad 5.45 \quad 0.00 \quad -20.00 \quad -20.00]
\tag{4.8}
$$

A saturação das superfícies é:

$$\begin{aligned} \delta_e \max &= \pm 30 \deg \\ \delta_f \max &= \pm 30 \deg \end{aligned} \tag{4.9}$$

O objetivo ideal em "pitch pointing" é comandar uma demanda de ângulo de "pitch" θ enquanto o ângulo de trajetória de vôo ("path angle") γ seja mantido nulo. Para o modo "altitude rate", mantém-se o "path angle" constante para o ângulo de "pitch" nulo.

O "path angle" está relacionado com a "altitude rate" da forma:

$$\gamma = \dot{h} / TAS \tag{4.10}$$

onde TAS é a "True Airspeed" e o "path angle" "γ" está relacionado com o ângulo de ataque "α" da forma:

$$\theta = \gamma + \alpha \tag{4.11}$$

Os autovalores de malha fechada assumida (Sobel et al, 1994) são:

$$? = [-5.6 + 4.2\,j \quad -5.6 - 4.2\,j \quad -1 \quad -19 \quad -19.5] \tag{4.12}$$

O problema é bem adequado para utilização da metodologia de controle preditivo, pois os horizontes de saída são constantes. A Figura 4.3 mostra os modos "pitch pointing" e "altitude rate" para 2 graus de demanda de "pitch" (θ) e "gama" (γ) respectivamente.

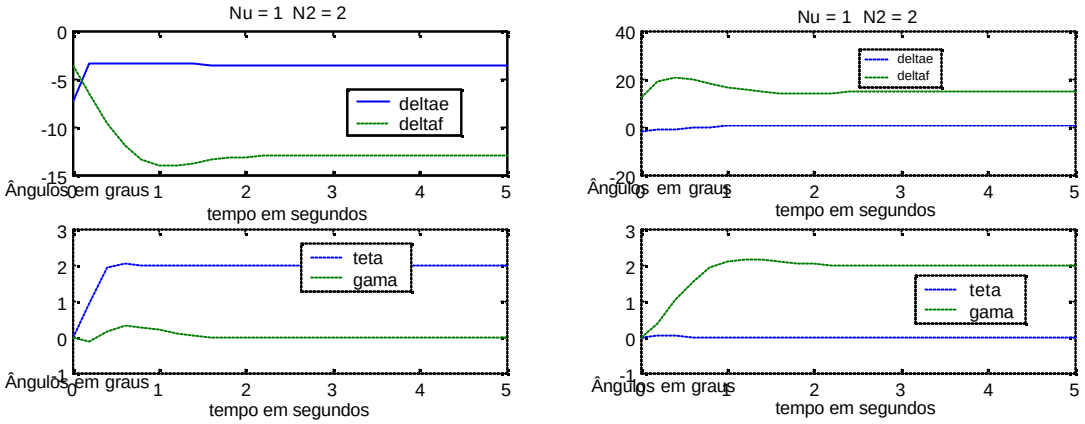
O valor da matriz de ganhos de realimentação obtidos com a função "PLACE" fica:

$$K = \begin{bmatrix} 1.4839 & -0.7581 & -4.2955 & 0.4780 & 0.0886 \\ 0.8553 & 0.1136 & -0.0567 & -0.0748 & -0.0535 \end{bmatrix} \tag{4.13}$$

A matriz utilizada na atribuição de autestrutura (Andry et al., 1983 e Sobel et al.,1985) foi:

$$Ka = \begin{bmatrix} -0.722 & -6.70 & -0.22 & 0.468 & 0.0587 \\ -0.301 & 5.51 & 0.994 & 0.223 & 0.187 \end{bmatrix} \quad (4.14)$$

acrescido da componente de controle “feedforward” (Apêndice A).



(a) (b)

Fig. 4.3 – Demanda de 2 graus : (a) “pitch pointing” e (b) “altitude rate”.

4.4 Conclusões

A primeira contribuição deste trabalho é a demonstração da utilização de controle preditivo para desacoplamento e confirmada pelos resultados obtidos. O índice de desempenho adotado foi o erro quadrático nas saídas tomadas no horizonte de saída N2=2.

No Capítulo 5 serão abordadas as considerações sobre horizontes de saída e controle.

A seguir, será considerado o desconhecimento do modelo, utilizando-se um modelo de identificação baseado em redes neurais artificiais. Os modelos dinâmicos de referência serão mantidos para comparação e conclusões.

CAPÍTULO 5

CONTROLE PREDITIVO NEURAL

5.1 Introdução

A seguir serão mostrados o formalismo, o desenvolvimento do modelo, a determinação do Jacobiano da rede e o gradiente da função desempenho para a implementação do controle preditivo neural.

A abordagem para a obtenção dos resultados teve como um dos objetivos elaborar rotinas com característica generalizada. Rotinas estas: definição da estrutura da rede, valores obtidos do treinamento, conjunto de validação, correlação de resultados, avaliação do Jacobiano da rede, onde a arquitetura adotada foi “feedforward”.

O desenvolvimento foi em Matlab® e as funções das “toolboxes” “nnet” (redes neurais) e “optimization” (otimização). O algoritmo de determinação do Jacobiano da rede foi desenvolvido e uma nova função do Matlab® para extração do Jacobiano de redes neurais foi obtida com elevado grau de generalização.

As funções permitem uma abordagem modular e versátil, desde a obtenção do conjunto de treinamento até a análise dos resultados, modificação de modelos e características da rede.

Os algoritmos utilizados estão listados no Apêndice D.

5.2 Considerações Sobre Métodos de Otimização

Os métodos de otimização podem ser classificados basicamente em relação à forma de utilização da informação sobre as derivadas das funções que são ou não avaliadas no método. Métodos de busca, que utilizam somente informação sobre a função (“simplex”), (Nelder. e Mead, 1965) são mais adequados quando a função possui não

linearidades e descontinuidades. Métodos que utilizam gradiente são mais eficientes quando a função possui primeira derivada contínua. Métodos de ordem superior, como o de Newton, utilizam informações de segunda derivada da função, a qual, quando numericamente calculada é de carga computacional elevada.

O método mais comum que utiliza gradiente é o conhecido “gradiente descendente”, onde a direção de busca é determinada na direção contraria do gradiente local.

Dos métodos que utilizam informação de gradiente, os mais eficientes em termos computacionais são os “quasi-Newton”. Estes métodos constroem informação de tendência da função a cada iteração para formular um problema quadrático equacionado com a Hessiana, a solução envolve a sua inversão, onde o método de Newton calcula diretamente, o que computacionalmente é dispendioso. O método “quasi-Newton” evita este cálculo utilizando informação da tendência construída da função e aproximando iterativamente a inversa da Hessiana a cada atualização.

Referências a estes métodos e o procedimento de atualização da Hessiana recaem no mais geral deles que é o BFGS (Broyden, 1970; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970).

Os métodos utilizados neste trabalho são encapsulados e são implementados nas funções do Matlab®. A função “trainlm”, que utiliza o método de “Levenberg-Marquardt”, foi adotada para treinamento da rede. A função do “fminunc” , que utiliza o método “quasi-Newton “ , foi adotada para o processo de otimização para a obtenção do controle. Como o objetivo não é uma exploração de métodos de otimização, estes foram escolhidos porque apresentaram um bom desempenho em precisão e carga computacional.

5.3 Determinação do Jacobiano da saída em relação à entrada da Rede

A segunda contribuição deste trabalho foi determinar o Jacobiano analítico de uma rede "feedforward" genérica, com o desenvolvimento e implementação de uma função de propósito geral no Matlab® para a sua obtenção.

O desenvolvimento do Jacobiano é mostrado no Apêndice B.

5.4 Esquema de Controle Preditivo Neural

O esquema de controle preditivo neural é mostrado na figura 5.1 a seguir:

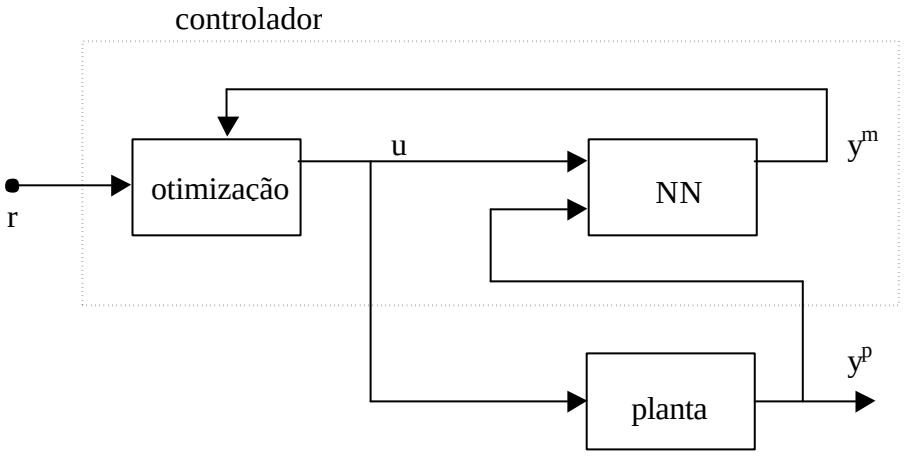


FIGURA. 5.1 – Controle Preditivo Neural.

Seja o índice de desempenho J em um instante "i":

$$J_i(\mathbf{u}) = \frac{1}{2}\{ \sum_{k=i+N_1}^{i+N_2} [\,\hat{\mathbf{y}}(k) - \mathbf{r}(k)\,]^T \mathbf{R}\ [\hat{\mathbf{y}}(k) - \mathbf{r}(k)\,] + \sum_{i}^{i+N_u-1} \Delta\mathbf{u}^T\ \mathbf{R_u}\ \Delta\mathbf{u}\ \} = \frac{1}{2}\{ \sum_{k=i+N_1}^{i+N_2} \hat{\mathbf{e}}(k)^T\ \mathbf{R}\ \hat{\mathbf{e}}(k) + \sum_{i}^{i+N_u-1} \Delta\mathbf{u}^T\ \mathbf{R_u}\ \Delta\mathbf{u}\ \} \quad (5.1)$$

escalar de argumento vetorial, com limites de saída ("bounds") $\mathbf{y}_{min} \le \mathbf{y} \le \mathbf{y}_{max}$ e $\mathbf{e}_{min} \le \mathbf{e} \le \mathbf{e}_{max}$, com restrições ("constraints") da variável manipulada (controle), $\mathbf{u}_{min} \le \mathbf{u} \le \mathbf{u}_{max}$ e $\Delta\mathbf{u} \le \Delta\mathbf{u}_{max}$

A janela de otimização reside no intervalo de predição $I_o = [N_1, N_2]$, o horizonte de controle é considerado dentro deste intervalo $I_u = [N_1, Nu]$, onde $N_2 \ge Nu$, a partir do qual o controle é constante ("receding horizon").

A saída do modelo, tipo NARMA fica:

$$\begin{aligned}\mathbf{y}(k+1) = f[&\mathbf{y}(k), \mathbf{y}(k-1), \mathbf{y}(k-2), ..., \mathbf{y}(k-n), \\ &\mathbf{u}(k), \mathbf{u}(k-1), \mathbf{u}(k-2), ..., \mathbf{u}(k-m)]\end{aligned} \tag{5.2}$$

A predição de "p" passos a frente, no intervalo de otimização $[N_1, N_2]$ é obtida do preditor neural da forma NARMA (Sorensen et al., 1999):

$$\begin{aligned}\hat{\mathbf{y}}(k+p) = f_{NET}[&\hat{\mathbf{y}}(k+p-1), \hat{\mathbf{y}}(k+p-2), .., \hat{\mathbf{y}}(k+p-\min(p,n)), \\ &\mathbf{y}(k-1), ..., \mathbf{y}(k-\max(n-p,0)), \mathbf{u}(k+p-1), ..., \mathbf{u}(k+p-1-m)]\end{aligned} \tag{5.3}$$

Notem-se os valores preditos das saídas $\hat{\mathbf{y}}$ e os valores medidos $\mathbf{y}$.

Cálculo do gradiente da função de desempenho $J_i(\mathbf{u})$ no instante "i":

Seja a parte do índice "J" relativa ao erro somente:

$$I_i^E(\mathbf{u}) = \frac{1}{2} \sum_{k=i+N_1}^{i+N_2} [\, \hat{\mathbf{y}}(k) - \mathbf{r}(k) \,]^T \mathbf{R} \; [\, \hat{\mathbf{y}}(k) - \mathbf{r}(k) \,] \tag{5.4}$$

Para a otimização, o gradiente $\nabla^T I(\mathbf{u})$ é calculado a seguir:

Sabendo-se que:

$$d\frac{(\mathbf{f}(\mathbf{x})^T\mathbf{R}\,\mathbf{f}(\mathbf{x}))}{d\mathbf{x}} = \mathbf{f}(\mathbf{x})^T(\mathbf{R}+\mathbf{R}^T)\frac{d\mathbf{f}(\mathbf{x})}{d\mathbf{x}} \tag{5.5}$$

com $\mathbf{R}$ diagonal, tem-se:

$$d\frac{(\mathbf{f}(\mathbf{x})^T\mathbf{R}\,\mathbf{f}(\mathbf{x}))}{d\mathbf{x}} = 2\,\mathbf{f}(\mathbf{x})^T\mathbf{R}\frac{d\mathbf{f}(\mathbf{x})}{d\mathbf{x}} \tag{5.6}$$

assim, para o problema proposto, o gradiente $\frac{\partial I_i(\mathbf{u})}{\partial \mathbf{u}(j)}$ é calculado, onde $\mathbf{u}$ é uma seqüência do tipo $\{\mathbf{u}(i),\mathbf{u}(i+1),....,\mathbf{u}(i+Nu-1)\}$, solução desejada do problema.

Considerando $N1=1$, ou seja, o 1º. instante discreto da saída "$i+1$" está sujeito ao controle no instante "i", o gradiente pode ser formado como:

$$\frac{\partial I_i^E(\mathbf{u})}{\partial \mathbf{u}(j)} = \sum_{k=1}^{N_2}[\hat{\mathbf{y}}(i+k)-\mathbf{r}(i+k)]^T\,\mathbf{R}\,\frac{\partial\,\hat{\mathbf{y}}(i+k)}{\partial \mathbf{u}(j)} \tag{5.7}$$

cujo gradiente transposto (∇^T) fica:

$$\frac{\partial I_i^E(\mathbf{u})}{\partial \mathbf{u}(j)}^T = \sum_{k=1}^{N_2}\left[\frac{\partial\,\hat{\mathbf{y}}(i+k)}{\partial \mathbf{u}(j)}\right]^T\mathbf{R}\,[\hat{\mathbf{y}}(i+k)-\mathbf{r}(i+k)], \tag{5.8}$$
$$j=i,i+1,...,i+Nu-1,\quad j<i+k$$

A propagação dos elementos envolvidos na formulação a seguir pode ser visualizada na Figura 5.2 :

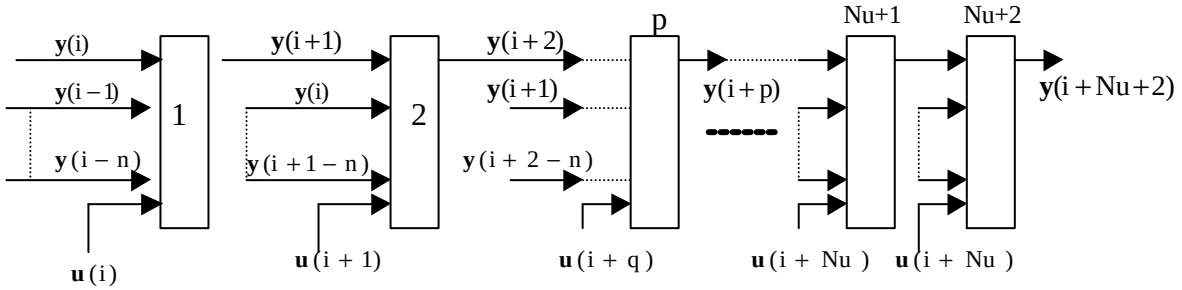


FIGURA. 5.2– Propagação do Modelo Neural.

$$\mathbf{J}_{yu}(i)=\begin{bmatrix} \frac{\partial\hat{\mathbf{y}}(i+1)}{\partial\mathbf{u}(i)}\Big|^{T} & \frac{\partial\hat{\mathbf{y}}(i+2)}{\partial\mathbf{u}(i)}\Big|^{T} & \frac{\partial\hat{\mathbf{y}}(i+3)}{\partial\mathbf{u}(i)}\Big|^{T} & \cdots\cdots & \cdots\cdots & \frac{\partial\hat{\mathbf{y}}(i+N_2-1)}{\partial\mathbf{u}(i)}\Big|^{T} & \frac{\partial\hat{\mathbf{y}}(i+N_2)}{\partial\mathbf{u}(i)}\Big|^{T} \\ 0 & \frac{\partial\hat{\mathbf{y}}(i+2)}{\partial\mathbf{u}(i+1)}\Big|^{T} & \frac{\partial\hat{\mathbf{y}}(i+3)}{\partial\mathbf{u}(i+1)}\Big|^{T} & \cdots\cdots & \cdots\cdots & \frac{\partial\hat{\mathbf{y}}(i+N_2-1)}{\partial\mathbf{u}(i+1)}\Big|^{T} & \frac{\partial\hat{\mathbf{y}}(i+N_2)}{\partial\mathbf{u}(i+1)}\Big|^{T} \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ 0 & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \frac{\partial\hat{\mathbf{y}}(i+N_2-1)}{\partial\mathbf{u}(i+N_u-1)}\Big|^{T} & \frac{\partial\hat{\mathbf{y}}(i+N_2)}{\partial\mathbf{u}(i+N_u-1)}\Big|^{T} \\ 0 & \cdots\cdots & 0 & \frac{\partial\hat{\mathbf{y}}(i+N_u)}{\partial\mathbf{u}(i+N_u-1)}\Big|^{T} & \cdots\cdots & \frac{\partial\hat{\mathbf{y}}(i+N_2-1)}{\partial\mathbf{u}(i+N_u-1)}\Big|^{T} & \frac{\partial\hat{\mathbf{y}}(i+N_2)}{\partial\mathbf{u}(i+N_u-1)}\Big|^{T} \end{bmatrix}_{Nu*du\times N2*dy} \tag{5.9}$$

O cálculo dos elementos da matriz $\mathbf{J}_{\mathbf{T}}\mathrm{yu}_{\mathrm{i}}$ seguem o seguinte procedimento:

Cada elemento da matriz será denominado:

$$\mathbf{J}_{yu}(i+p,i+q)=\frac{\partial\hat{\mathbf{y}}(i+p)}{\partial\mathbf{u}(i+q)} \quad \text{com } p>q \tag{5.10}$$

correspondendo à linha “p” e coluna “q”.

e os Jacobianos intermediários:

$$\mathbf{J}_{yy}(i+p,i+q)=\frac{\partial\hat{\mathbf{y}}(i+p)}{\partial\hat{\mathbf{y}}(i+q)} \tag{5.11}$$

que correspondem aos Jacobianos de cada rede da Figura 5.2 da saída em relação às entradas que são as saídas atrasadas.

Determina-se então, para um número de atrasos genérico “n” (Equação 5.2) da rede, a fórmula recursiva para o elemento da matriz:

$$\mathbf{J}_{yu}(i+p,i+q) = \sum_{j=p-1}^{p-n} \mathbf{J}_{yy}(i+p\,,i+j) * \mathbf{J}_{yu}(i+j\,,i+q)\,, \tag{5.12}$$

$j = p-1, p-2, ..., p-n$, com “j” decrescente, e

para $q = 0, ..., Nu-1$ e $p = q+1, ..., Nu$

Para os estágios onde $p > Nu$, a fórmula fica:

$$\mathbf{J}_{yu}(i+p,i+Nu) = \sum_{j=p-1}^{p-n} \{\mathbf{J}_{yy}(i+p\,,i+j) * \mathbf{J}_{yu}(i+j\,,i+Nu)\} + \mathbf{J}_{yu}(i+p,i+p-1) \tag{5.13}$$

para $p = Nu+2, ..., N2$

lembrando que:

$$\mathbf{J}_{yy}(i+p,i+q) = \mathbf{0} \quad \text{para} \quad p-q > n \tag{5.14}$$

$$\mathbf{J}_{yu}(i+p,i+q) = \mathbf{0} \quad \text{para} \quad p \le q \tag{5.15}$$

As deduções das fórmulas 5.12 e 5.13 são mostradas no Apêndice C.

Lembrando que a matriz assim obtida está transposta, tem-se:

$$\left[\mathbf{J}_{yu}\right]_{(Nu*du \times N2*dy)} = \left[\mathbf{J}_{yu}(k,i)^{T}_{(dy \times du)}\right]^{T}_{(Nu*du \times N2*dy)} \tag{5.16}$$

para $i = 1, ..., Nu$ e $k = i+1, ..., N2$

que é a Equação 5.9 na forma compacta.

Sendo o erro de saída:

$$\mathbf{e}_R(i) = \mathbf{R} * \mathbf{e}(i) = \mathbf{R} * [\hat{\mathbf{y}}(i+k) - \mathbf{r}(i+k)] \tag{5.17}$$

onde:

$$\mathbf{e}_R(i) = \begin{bmatrix} e_R(i+1) \\ e_R(i+2) \\ e_R(i+3) \\ ..... \\ ..... \\ e_R(i+N_2) \end{bmatrix}_{N2*dy \times 1} \tag{5.18}$$

tem-se assim, o gradiente da parte relativa ao erro, em um instante genérico "i":

$$\left[\nabla^T_{\mathbf{u}(j)} I_E(i)\right]_{N2*du \times 1} = \left[\mathbf{J}_{yu}\right]_{Nu*du \times N2*dy} * \left[\mathbf{e}_R(i)\right]_{N2*dy \times 1} \tag{5.19}$$

A parte relativa ao controle é elementar e é dada por (Sorensen et al., 1999) como:

$$\left[\nabla^T_{\mathbf{u}(j)} I_U(i)\right]_{Nu*du \times 1} = \left[\frac{\partial \Delta \mathbf{u}}{\partial \mathbf{u}}(i)\right]_{Nu*du \times Nu*du} * \left[\Delta \mathbf{u}(i)\right]_{Nu*du \times 1} \tag{5.20}$$

onde:

$$\frac{\partial \Delta \mathbf{u}}{\partial \mathbf{u}}(i) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ ..... & ..... & ..... & ..... & ..... \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}_{Nu*du \times Nu*du} \tag{5.21}$$

O gradiente total fica:

$$\nabla_{\mathbf{u}}^{T}(i) = \mathbf{J}_{yu}(i) * \mathbf{e}_{R}(i) \quad + \quad \frac{\partial \Delta \mathbf{u}}{\partial \mathbf{u}}(i)\, \Delta \mathbf{u}(i) \tag{5.22}$$

O processo de otimização utiliza gradiente analítico ou numérico, dependendo da escolha adotada, ao final, somente $\mathbf{u}(i)$ da seqüência obtida é aplicado na planta, daí o processo é reiniciado no instante "$i+1$" seguinte.

CAPÍTULO 6

PROCESSOS E RESULTADOS

Nesta seção serão mostrados os procedimentos utilizados para a obtenção dos resultados assim como resultados intermediários e finais, apresentados de forma similar àquelas do Capítulo 4.

A linguagem utilizada foi o Matlab®, e as "toolboxes" Neural Nets, Optimization.

6.1 Introdução

Redes neurais possuem capacidade de representação ampla de sistemas onde o treinamento "off-line" ou "on-line" conferem a característica adaptativa para sistemas onde o modelo não é conhecido. A forma adimensional das variáveis permite versatilidade de arquiteturas de representação, onde a conectividade densa produz abstração do sentido físico das variáveis, pois a partir da primeira camada pode haver, por exemplo, multiplicações e somas entre posição, aceleração e temperatura. Esta mesma característica permite o mapeamento de qualquer sistema, dependente da quantidade de neurônios das camadas internas.

Neste trabalho, utiliza-se, para efeito de comparação, um modelo linear de aeronave, porque o objetivo principal é testar a validade de uma abordagem alternativa para desacoplamento de modos para o mesmo modelo nominal de aeronave utilizado no Capítulo 4, mostrando que, sem utilizar atribuição de auto-estrutura, consegue-se o mesmo desacoplamento estabelecendo-se horizontes de saída desejados em um esquema de controle preditivo com redes neurais, que aqui são utilizadas para substituir o desconhecimento do modelo.

Para as manobras estabelecidas neste tipo de problema, a aeronave está em vôo nivelado, a 3000 metros de altitude com velocidade constante de 0.6 mach, sendo esta a condição inicial das manobras não típicas indicadas, mesmo que fosse utilizado um

modelo de aeronave completo, não linear, nestas condições de vôo, a aeronave está "trimada", comportando-se aproximadamente como o modelo linear. A rede neural possui funções de ativação não linear caracterizando o problema como não-linear.

6.2 Processos

O processo de desenvolvimento foi concebido dentro de uma abordagem generalista, isto é, todas as funções e rotinas foram elaboradas para automatizar todo o processo, permitindo versatilidade na implementação de diferentes arquiteturas, testes, etc. O processo está descrito no diagrama da Figura 6.1.

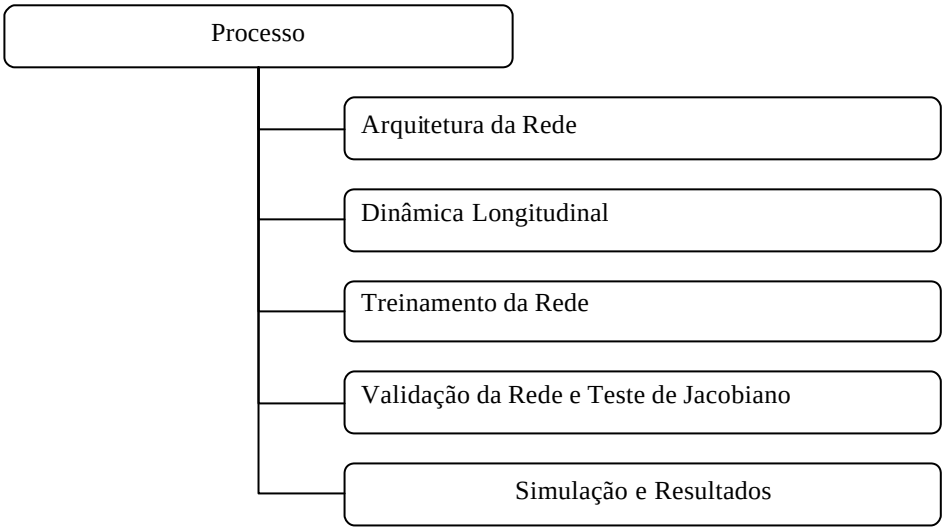


FIGURA. 6.1 – Definição do Processo.

6.3 Arquitetura da Rede

Esta etapa tem o objetivo de definir uma arquitetura de rede adequada para representar o sistema via treinamento de várias arquiteturas e análise de resultados de treinamento e a análise de correlação.

O critério para a definição da arquitetura da rede neural mais adequada foi resultado de avaliações levando-se em conta o conjunto de treinamento, avaliação da sensitividade da rede para as entradas de controle. A Figura 6.2 mostra o esquema adotado.

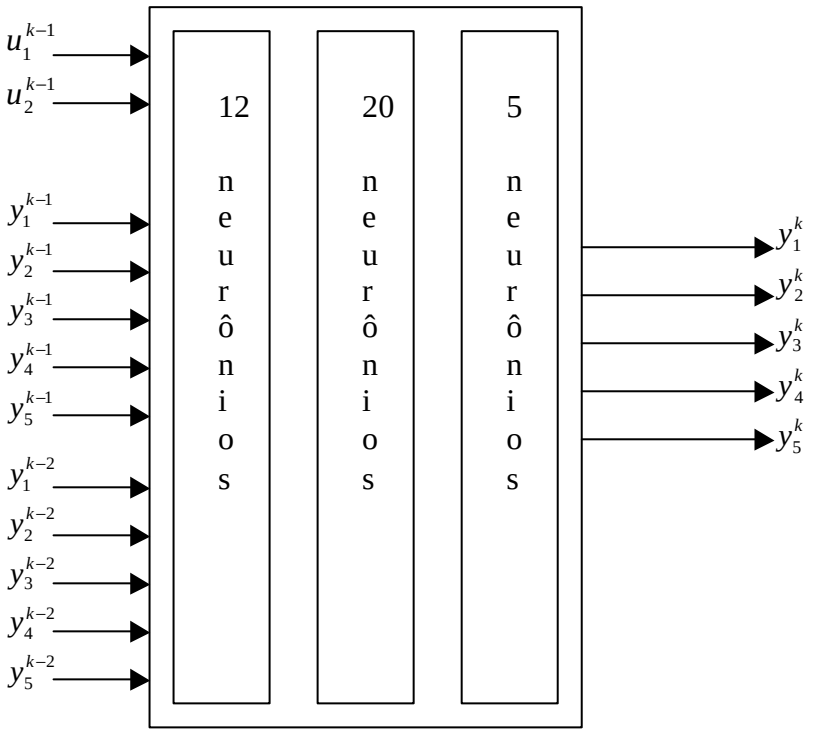


FIGURA. 6.2 – Arquitetura utilizada.

A rede possui duas entradas de controle atrasadas de um passo, referentes às superfícies de comando longitudinal da aeronave: profundor (“elevator”) “δ_{EC}” e do flaperon “δ_{FC}”. As outras entradas são as saídas atrasadas de dois passos. Esta arquitetura mostrou-se adequada para representar o modelo da dinâmica longitudinal da aeronave descrita no Capítulo 4.

6.4 Treinamento da Rede

O padrão de treinamento foi implementado com pulsos de amplitude aleatória dentro dos limites estabelecidos pelo modelo de referência do Capítulo 4. O conjunto de

treinamento tem cerca de 2500 pontos. Serão apresentados dados de três redes (A, B e C) treinadas na seqüência de análise.

A rede A foi treinada com um padrão de pulsos de entrada de controle simultâneos e a uma taxa de amostragem de 100 milisegundos. O resultado foi uma baixa sensitividade às entradas de controle, onde se concluiu que a uma taxa de amostragem pequena, para a natureza da dinâmica envolvida, as entradas correspondentes às saídas atrasadas $,\mathbf{y}(k-1)$ possuíam valores muito próximos do instante presente $\mathbf{y}(k)$, treinando a rede praticamente para um mapeamento de identidade $(\mathbf{y}(k+1) \cong \mathbf{y}(k)$ e conseqüentemente diminuindo a sensitividade às entradas correspondentes ao controle, apesar do desempenho do treinamento ser satisfatório, como mostrado na Figura 6.3, não foi possível utilizá-la para identificação do sistema.

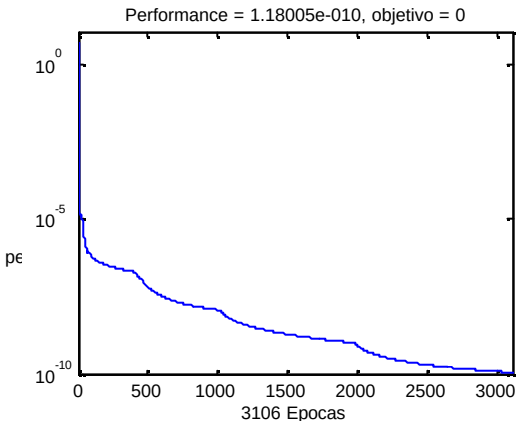


FIGURA. 6.3 – Desempenho de Treinamento para a rede A.

O padrão de treinamento para as entradas correspondentes ao controle foi modificado com a finalidade de estimular a sensitividade da rede para os controles. Para este objetivo o padrão de entrada foi particionado em três conjuntos de dados: o primeiro manteve-se uma das duas componentes do controle nula e a outra com padrão de pulsos já descrito; o segundo trocou-se de componentes com o mesmo padrão; finalmente estimularam-se as duas entradas simultaneamente. A Figura 6.4 mostra o padrão utilizado para a rede C, dentro de um subintervalo do conjunto para melhor

visualização com a amostragem de 0.2 segundos. Com estas modificações no padrão, obteve-se um desempenho melhor na identificação do sistema pela rede neural.

Neste padrão, verificou-se também a largura do pulso de amplitude aleatória necessária para englobar o transitório e regime permanente da dinâmica do sistema para o treinamento.

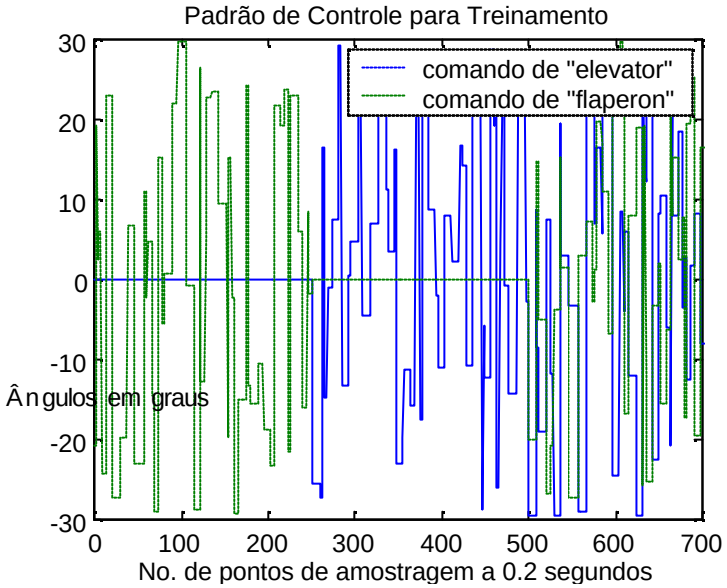


FIGURA. 6.4 – Padrão referente às entradas de comando “elevators” e “flaperons”.

6.5 Validação da Rede

Para a validação da rede A treinada, utilizou-se um padrão de excitação de controle oscilatório mostrado na Figura 6.5a, dentro de um conjunto de possibilidades de entradas de controle satisfatório. As saídas (padrão e rede) são mostradas nas Figura 6.5b a 6.5f.

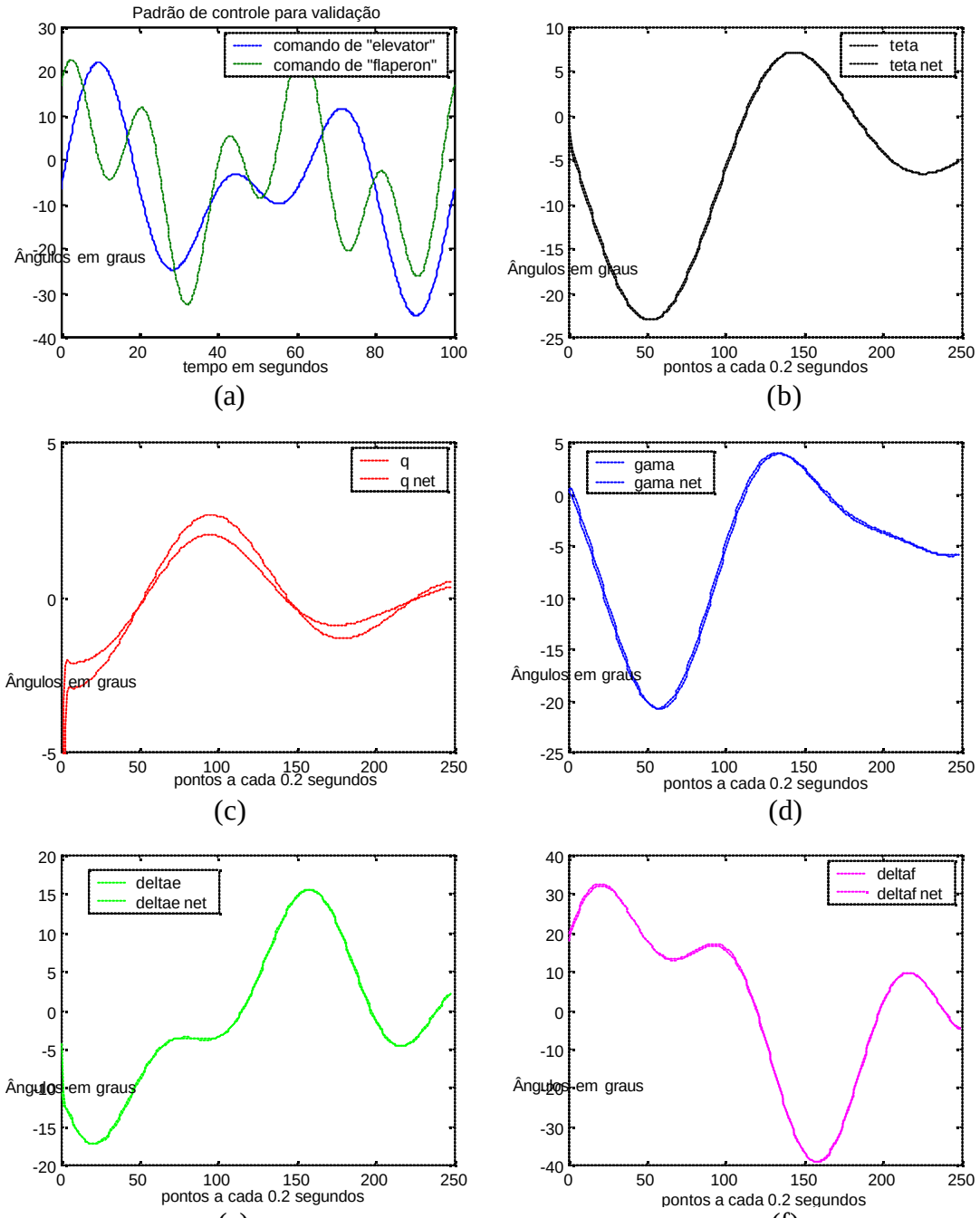


FIGURA. 6.5 – Rede A: (a) Padrão de Controle para validação, (b) saída de “pitch” padrão e da rede, (c) saída de “pitch rate” padrão e da rede, (d) saída de direção de trajetória padrão e da rede, (e) saída de deflexão de profundor padrão e da rede, (f) (e) saída de deflexão de flaperon padrão e da rede.

O erro absoluto entre as saídas padrão e da rede podem ser visualizadas na Figura 6.6.

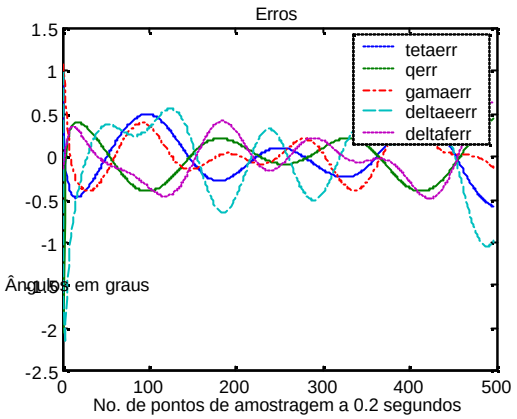
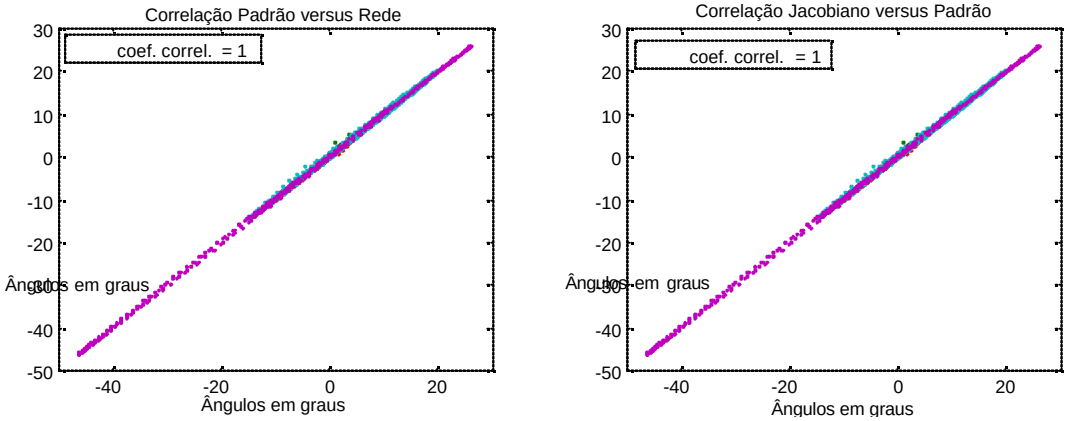


FIGURA. 6.6 – Rede A: Erros entre as saídas padrão e as saídas da rede.

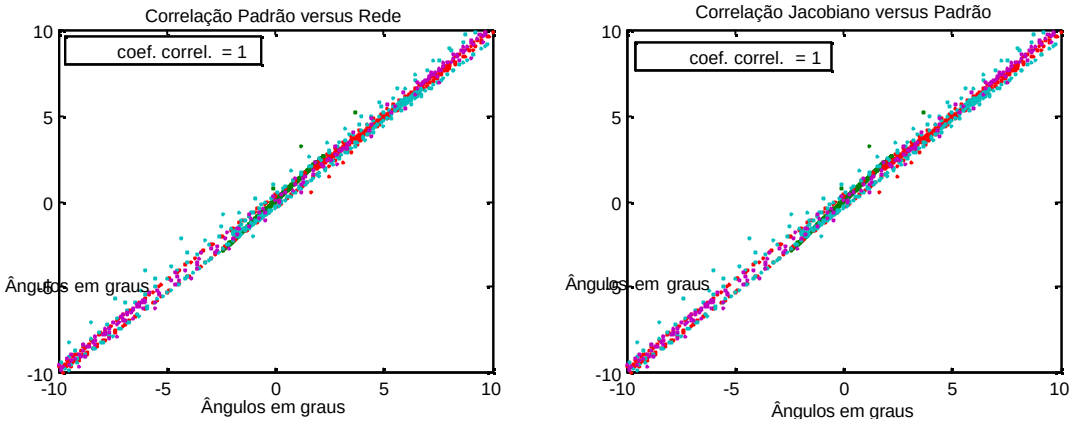
A análise de correlação para a rede A também foi utilizada, onde todas as componentes foram reunidas em um mesmo gráfico de correlação. Verifica-se na Figura 6.7a o valor alcançado entre o padrão e a rede treinada e na Figura 6.7b o Jacobiano com o padrão.



(a) (b)

FIGURA. 6.7 – Rede A: Correlação entre as saídas padrão e da rede: (a) Padrão versus rede e (b) Padrão versus saídas extrapoladas com o Jacobiano.

Ampliando-se os gráficos para visualizar a dispersão relativa (Figura 6.8):



(a) (b)

FIGURA. 6.8 – Rede A: Figura 6.7 ampliada: (a) Padrão versus rede e (b) Padrão versus saídas extrapoladas com o Jacobiano.

As redes apresentaram o mesmo perfil de correlação, o que ocorre a cada novo treinamento é que as sensitividades são diferentes, logo para atingir-se o mesmo desempenho, variáveis de controle são ajustadas para cada rede. No caso da Rede A, o padrão de treinamento utilizado não foi o da Figura 6.4, a conseqüência foi uma rede com baixo ganho nas entradas de controle, o que provocou um fraco desempenho. Nos casos das redes B e C, foram obtidos resultados satisfatórios.

Vale observar que dados angulares e velocidades angulares foram agrupados na correlação com a finalidade de verificar desvios absolutos, porém o efeito de um erro em “pitch rate” e em “pitch” possuem sensitividades diferentes no sistema.

6.5.1 Resultados

Os resultados apresentados são um resumo da análise de diversos casos e foram selecionados os mais representativos do problema. Para cada treinamento executado os pesos sinápticos convergiram para valores diferentes, porém mantendo a correlação com o modelo. O método de treinamento das redes utilizado foi o Levenberg-Marquardt da função “trainlm” do Matlab®, que mostrou o melhor desempenho em atingir o nível de precisão desejado.

A seguir são apresentados casos de teste envolvendo duas redes (B e C), que reúnem os principais resultados obtidos.

A seguir, na Figura 6.9 é mostrado o desempenho do treinamento para a rede B, notando-se que apesar da tolerância atingida ser cinco vezes maior do que a rede A, o padrão de treinamento é que foi determinante para o bom desempenho da rede na identificação do sistema em relação à rede A.

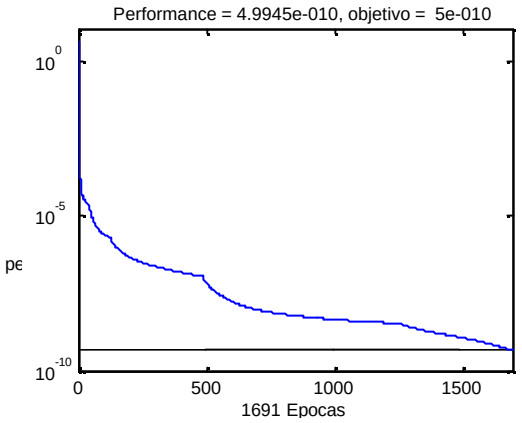


FIGURA. 6.9 - Desempenho de Treinamento para a rede B.

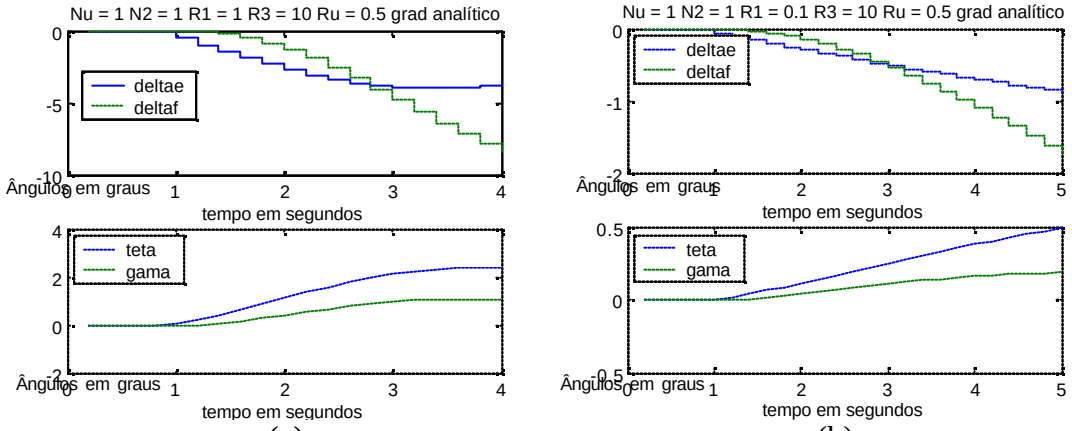
Os casos de teste a seguir foram realizados com o objetivo de estimar a sensitividade do sistema de controle aos parâmetros de ponderação dos erros de saída (matriz R) e da ponderação do controle (Ru) presentes no índice de desempenho da Equação 5.1. Foram escolhidos valores de Nu=1 e N2=2.

A ponderação R(1,1) é realizada sobre as saídas de ângulo de arfagem (“pitch”), a ponderação R(2,2) sobre o “pitch rate” e a ponderação R(3,3) sobre o ângulo de trajetória da aeronave (“path angle”), assim, o controle é determinado a fim de satisfazer o horizonte de saída especificado e com ponderação determinada nas componentes de R diagonal “R(i,i)”, denominadas de agora em diante de “Ri” para simplificação da grafia.

A Figura 6.10, mostra o resultado com o valor de R1 menor; como se trata de um ganho sobre o erro de saída em ângulo de inclinação (“pitch”), há um aumento do tempo de acomodação da resposta em “pitch” do sistema.

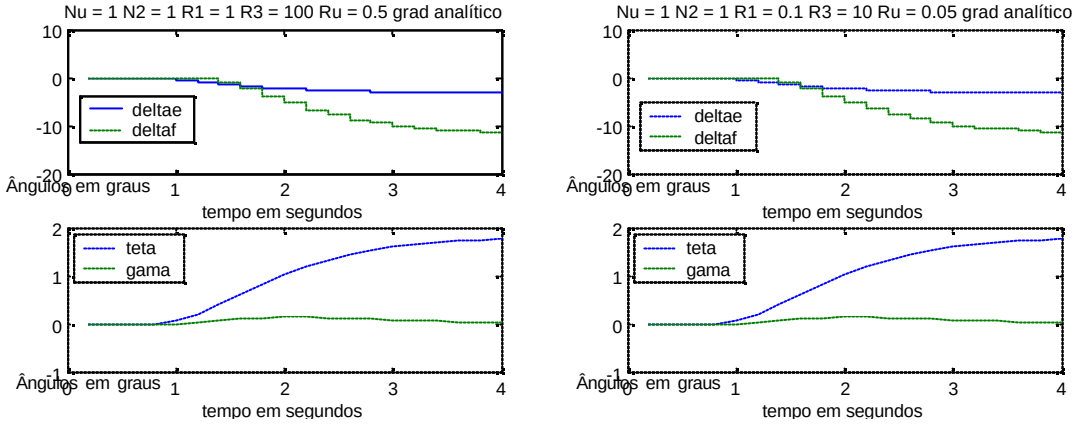
Por outro lado, a Figura 6.11a mostra que a elevação da ponderação na componente para R3 = 100 para R1 = 1, aumentou o ganho na componente de erro em “gama”, assim, evitando-se que o ângulo “gama” se afaste do valor nulo inicial o erro total a ser minimizado é praticamente debitado ao “pitch”, permitindo ao processo de otimização diminuir o tempo de acomodação em inclinação (“pitch”).

Nota-se, comparando-se as Figuras 6.11a e 6.11b, que a relevância está na relação de ponderação entre as saídas e o controle, pois a razão entre estes valores foi mantida, apesar de valores absolutos 10 vezes menores para R1, R2 e Ru.



(a) (b)

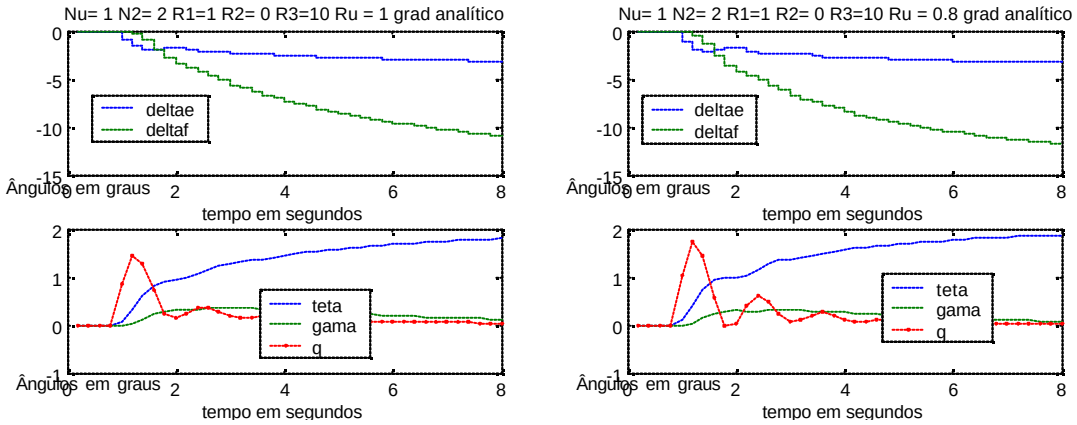
FIGURA. 6.10 – Resultado para “pitch pointing” de 2 graus com a Rede B: a) R1 = 1, R3 = 10, Ru = 0.5 e (b) R1 = 0.1, R3 = 10, Ru = 0.5.



(a) (b)

FIGURA. 6.11 - “Pitch pointing” de 2 graus com a Rede B: (a) R1 = 1, R3 = 100, Ru=0.5 e (b) R1 = 0.1, R3 = 10, Ru=0.05.

A Figura 6.12 mostra que a diminuição da ponderação no controle faz com que a variação do controle entre os instantes seja menos relevante ao índice de desempenho da Equação 5.1, assim, há uma maior variação das estimativas das componentes do controle produzindo um transitório com maior oscilação seguir, os casos de teste envolvem o modo “altitude rate”, para os mesmos parâmetros.

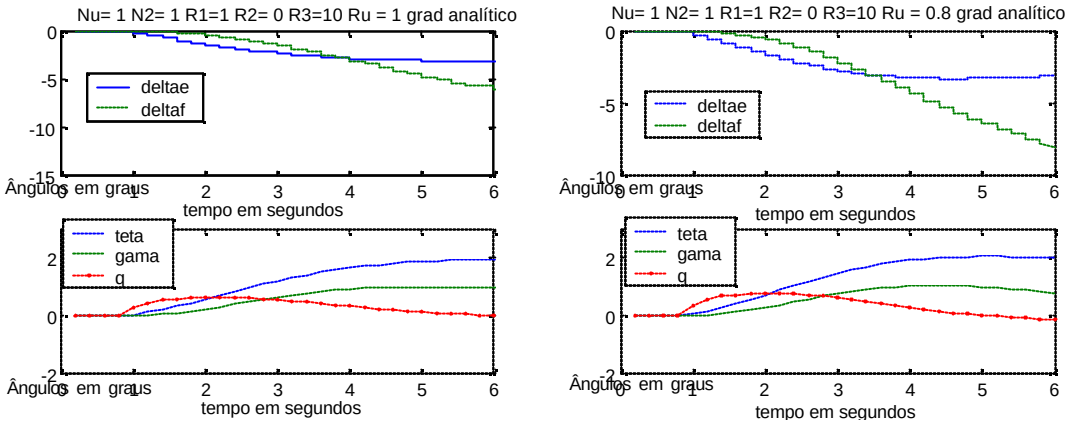


(a) (b)

FIGURA. 6.12 - “Pitch Pointing” de 2graus com a Rede B para Nu=1 e N2=2: (a) Ru = 1 e (b) Ru = 0.8.

O último caso de teste para a rede B mostra o efeito do horizonte de saída mais curto aliado à ponderação de “Ru” utilizadas na Figura 6.12. A diminuição do horizonte de

saída pode ser relacionada com a diminuição na ordem do sistema, conseqüentemente a oscilação típica em sistemas de maior ordem (Figura 6.12) não está presente para o caso mostrado na Figura 6.13.



(a) (b)

FIGURA. 6.13 – Rede B: “Pitch pointing” com Nu=N2=1: (a) Ru = 1 e (b) Ru = 0.8.

A seguir, os casos de teste são realizados sobre a rede C, treinada em um maior números de épocas atingindo uma tolerância menor (Figura 6.14).

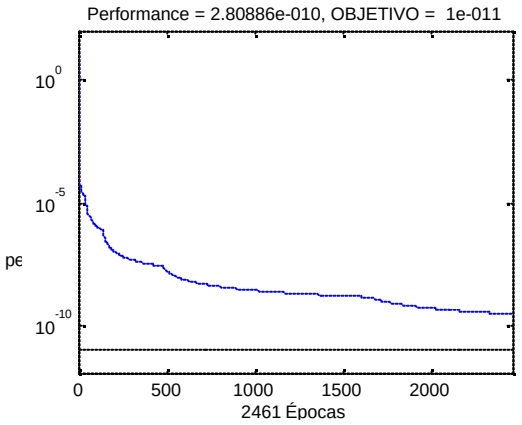


FIGURA. 6.14 - Desempenho de Treinamento para a rede C.

Nos casos de teste a seguir, explorou-se a variação dos horizontes de controle e saídas, a utilização do gradiente numérico para comparação ao desempenho utilizando gradiente analítico.

A ponderação maior da variação do controle no índice de desempenho (Figura 6.15) produz uma maior restrição à sua variação, fazendo com que a componente dos controles sejam mais conservadoras; a conseqüência é uma atuação atrasada e aumento no tempo de acomodação do sistema.

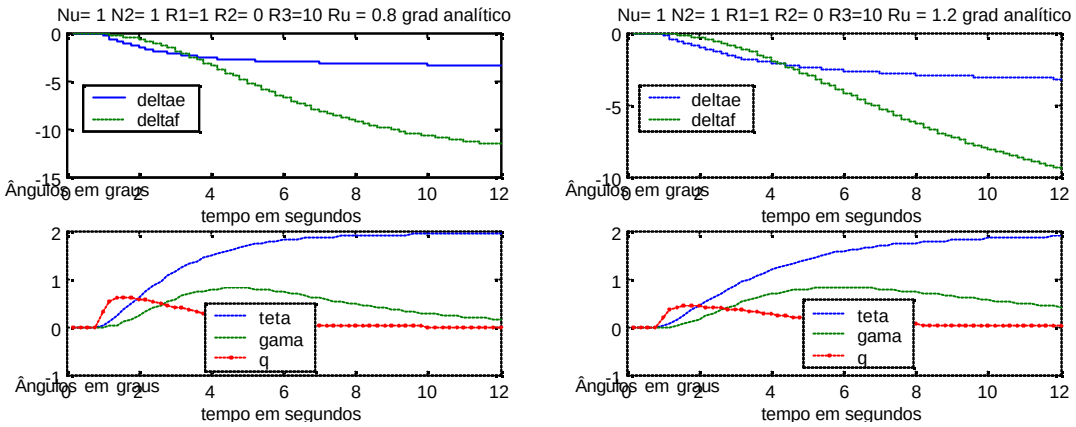
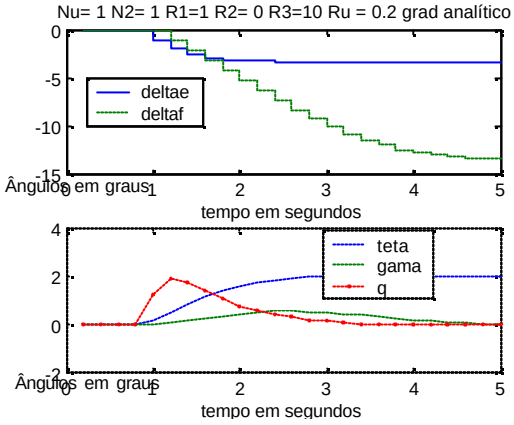


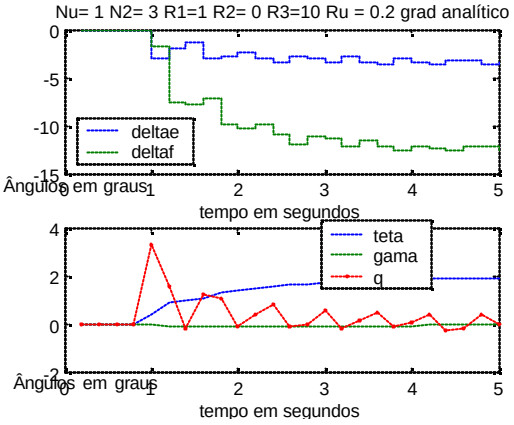
FIGURA. 6.15 – "Pitch pointing" de 2 graus com Nu=N2=1: (a) Ru = 0.8 e (b) Ru = 1.2.

Nas Figuras 6.16, como no caso da comparação entre as Figura 6.12 e 6.13, o aumento do horizonte de saída produz um aumento na ordem do sistema, surgindo assim oscilação em "pitch rate".

Na Figura 6.17 é mostrado que o gradiente numérico e o analítico possuem performances praticamente indiferentes. A vantagem do gradiente analítico é a carga computacional envolvida, pois no caso do gradiente numérico, o algoritmo utilizado é de diferenças finitas, procedimento custoso numericamente.

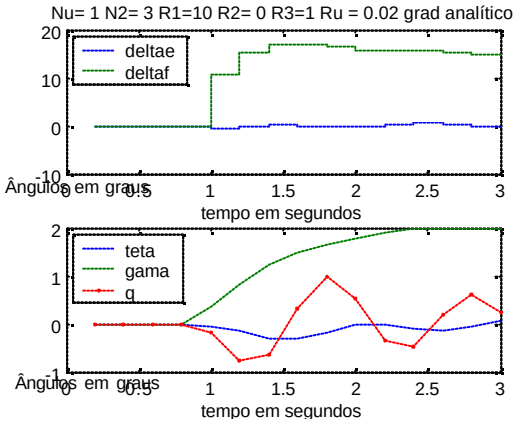


(a)

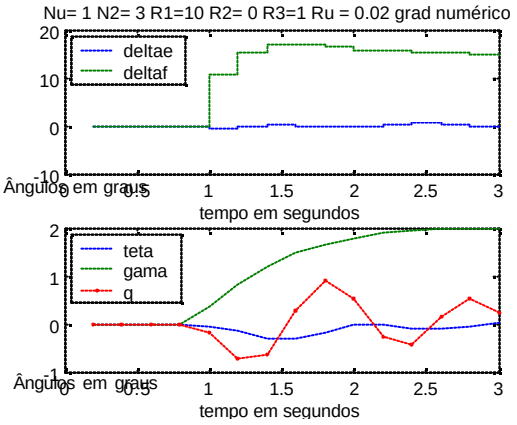


(b)

FIGURA. 6.16 - “Pitch pointing” de 2 graus com a Rede C: (a) N2 = 1 e (b) N2 = 3.



(a)



(b)

FIGURA. 6.17 - “Altitude rate” de 2 graus com N2 = 3 com a rede C: (a) gradiente analítico e (b) gradiente numérico.

A Figura 6.18 mostra resultados com adição de ruído nas componentes de saídas do sistema, o horizonte maior é mais robusto na presença de ruído devido ao aumento de ordem do sistema, que “filtra” o ruído na composição do erro de performance.

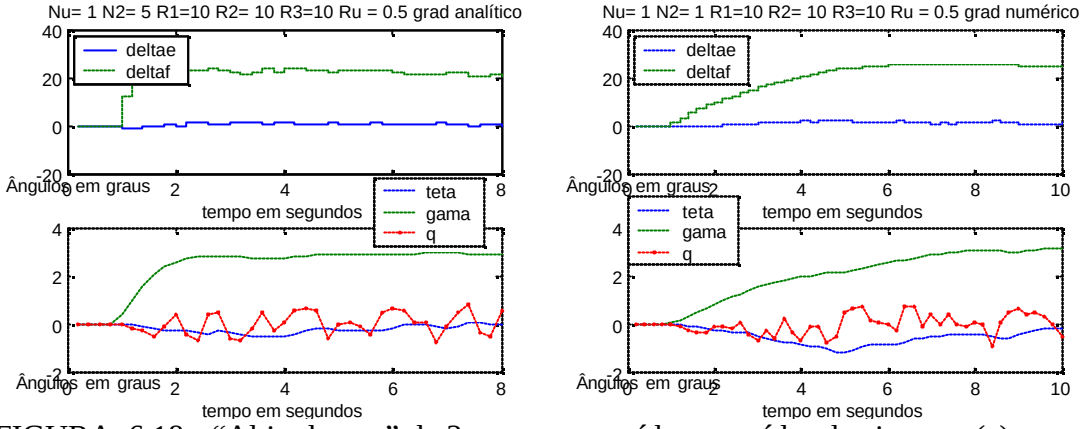
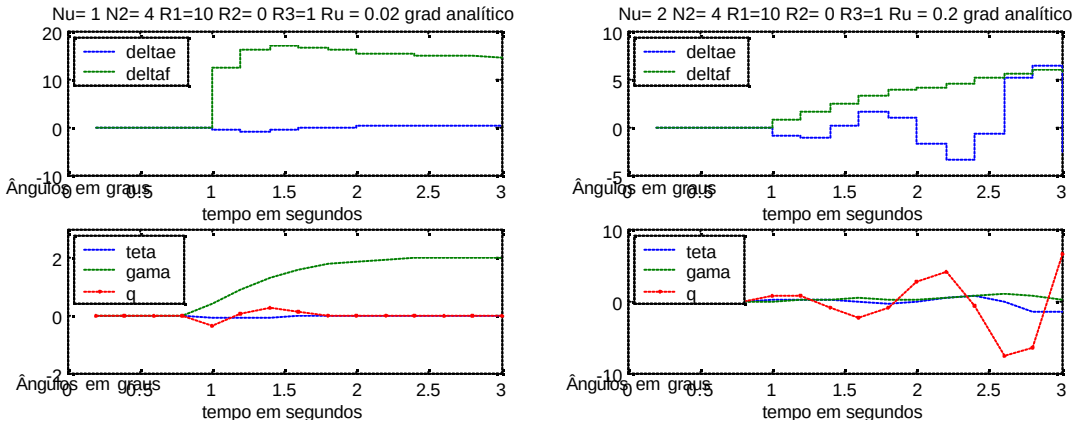


FIGURA. 6.18 - "Altitude rate" de 3 graus com ruído nas saídas do sistema :(a) gradiente analítico com N2=5 e (b) gradiente numérico com N2=1.

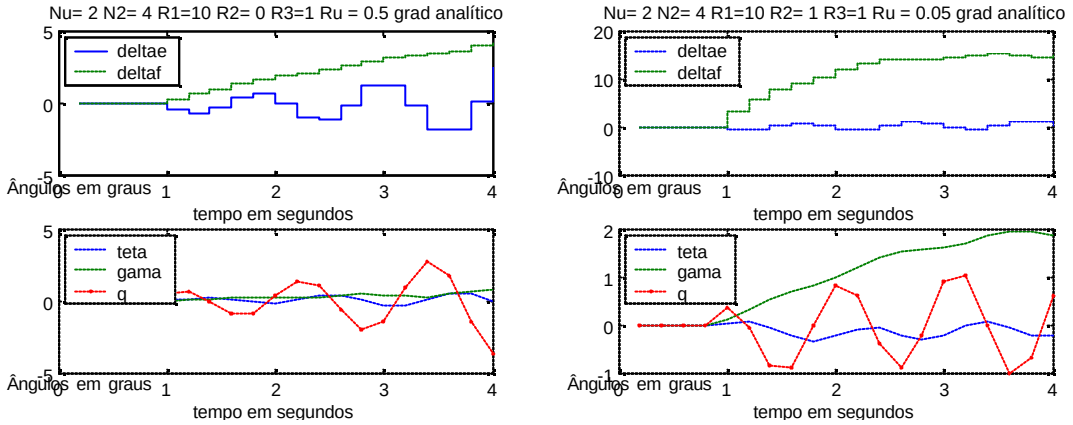
O resultado apresentado na Figura 6.19 mostra que o aumento no horizonte de controle provoca instabilidade do sistema devido ao fato de incluir-se no índice de desempenho componentes de controle que o tornam conservador para a dinâmica envolvida, em outras palavras, introduziram-se equivalentes a "zeros de transmissão" no sistema que induz a atrasos que levam à instabilidade.



(a) (b)

FIGURA. 6.19 – "Altitude rate" de 2 graus com a Rede C: (a) Nu = 1, N2 = 4 e (b) Nu = 2, N2 = 4.

Aumentando-se a ponderação do controle, obtém-se a Figura 6.20, onde R2 é utilizado para ponderar o “pitch rate”, nota-se na Figura 6.20b diminuição da amplitude da oscilação.



(a) (b)

FIGURA. 6.20 - “Altitude rate” de 2 graus com a rede C: (a) R2 = 0 e (b) R2 = 1.

A Figura 6.21 mostra a condição onde foi estabelecida uma ponderação para o “pitch rate”, R2 = 10, forçando o sistema a manter esta grandeza nula, o resultado é estabilização do sistema.

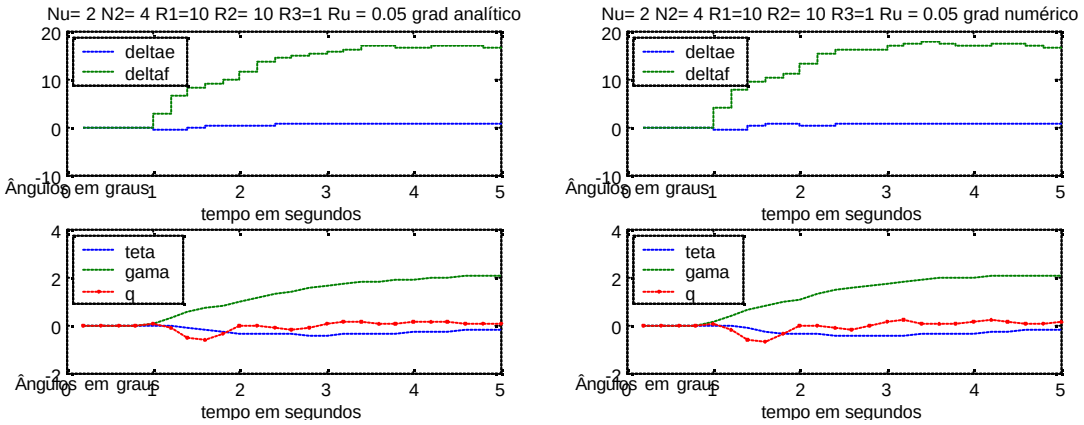


FIGURA. 6.21 - “Altitude rate” de 2 graus com R2 = 10 estabilizado: (a) gradiente analítico e (b) gradiente numérico.

O esquema permite a especificação genérica do perfil de saída; os resultados a seguir mostram também a trajetória com a inclinação da aeronave, que apesar de ser da ordem

de 3 graus, no gráfico o ângulo é ampliado pela escala fornecendo uma visualização mais perceptível da trajetória e inclinação da aeronave a cada ponto.

A Figura 6.22 mostra o caso onde somente há demanda para a inclinação (“pitch”),ou seja, a ponderação R1 = 10, as outras são nulas, permitindo qualquer perfil de saída nas outras componentes. Verifica-se que a trajetória segue um padrão normal para uma demanda de “pitch”.

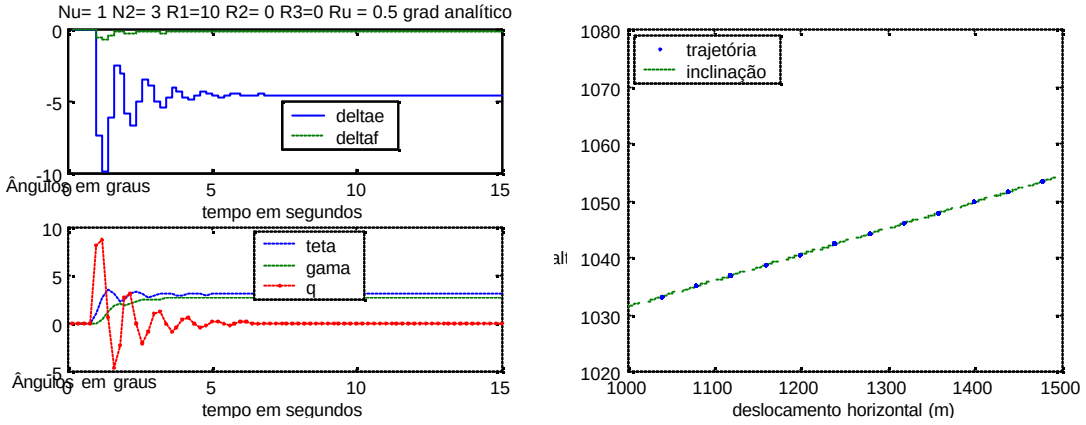


FIGURA. 6.22 - “Pitch demand” de 3 graus com R1 = 10: (a) controles e saídas e (b) trajetória com a inclinação (“pitch”) da aeronave.

Estabelecendo demandas de inclinação de 3 graus e -3 graus para o ângulo de trajetória (“path angle”), com R1=R2=R3=10, obtém-se um movimento descendente ao mesmo tempo em que a inclinação é positiva (Figura 6.23), mostrando assim, o potencial teórico do esquema para aplicações em diretor de vôo para decolagem e aterrissagem, lembrando aqui, que uma implementação em aeronave consiste em uma seqüência de etapas, desde o desenvolvimento até a certificação.

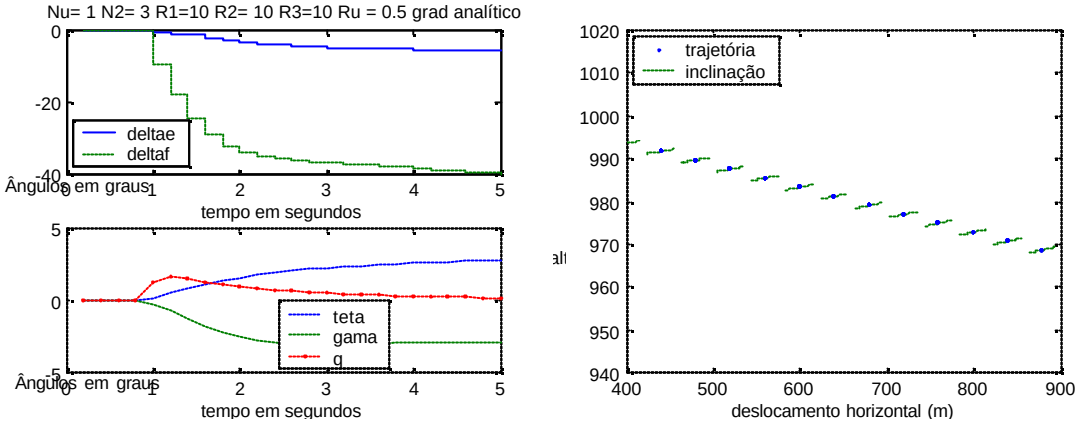


FIGURA. 6.23 - “Pitch pointing de 3 graus / “altitude rate” de -3 graus: (a) controles e saídas e (b) trajetória com a inclinação (“pitch”) da aeronave.

A seguir, verifica-se o efeito da ponderação R2, no “pitch rate”, que melhora a estabilização do sistema e também produz um perfil de trajetória com melhor performance. A Figura 6.24 mostra o caso sem a ponderação, R2=0; nota-se que o fato do transitório possuir um tempo de acomodação maior, faz com que o “pitch pointing” seja acompanhado de uma elevação de altitude. Nota-se também que a inclinação está propositadamente exagerada devido à escala do gráfico.

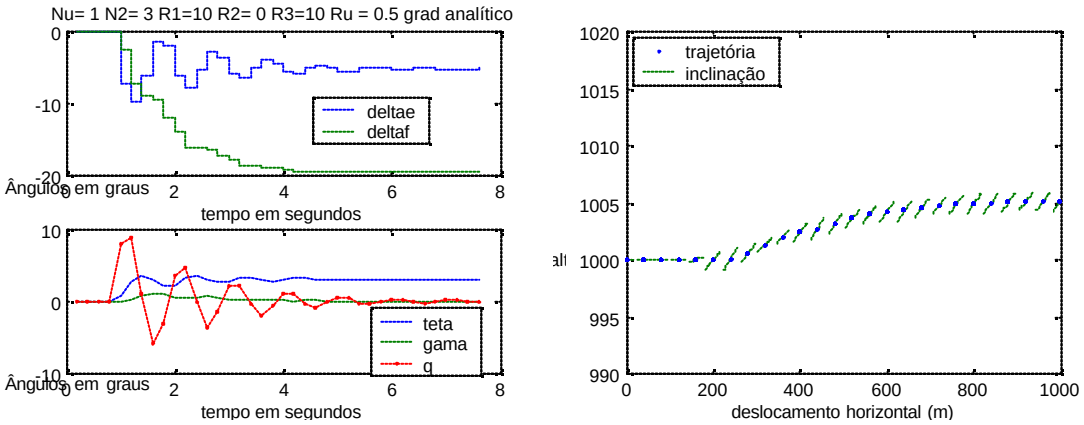


FIGURA. 6.24 - “Pitch pointing de 3 graus com R2 =0 (a) controles e saídas e (b) trajetória com a inclinação (“pitch”) da aeronave.

A Figura 6.25 mostra o caso para R2=10, onde o transitório é mais suave e como não há oscilação de “gama”, praticamente a alteração da altitude é mínima, melhorando a performance do modo.

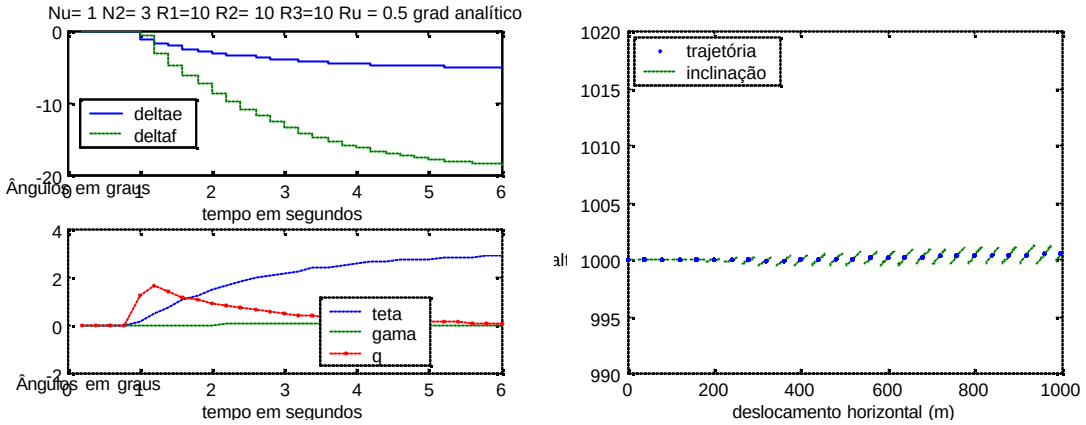


FIGURA. 6.25 - “Pitch pointing de 3 graus com R2 = 10 (a) controles e saídas e (b) trajetória com a inclinação (“pitch”) da aeronave.

A Figura 6.26 tem o objetivo de ilustrar o modo altitude rate com a trajetória típica de elevação de altitude com inclinação praticamente nula.

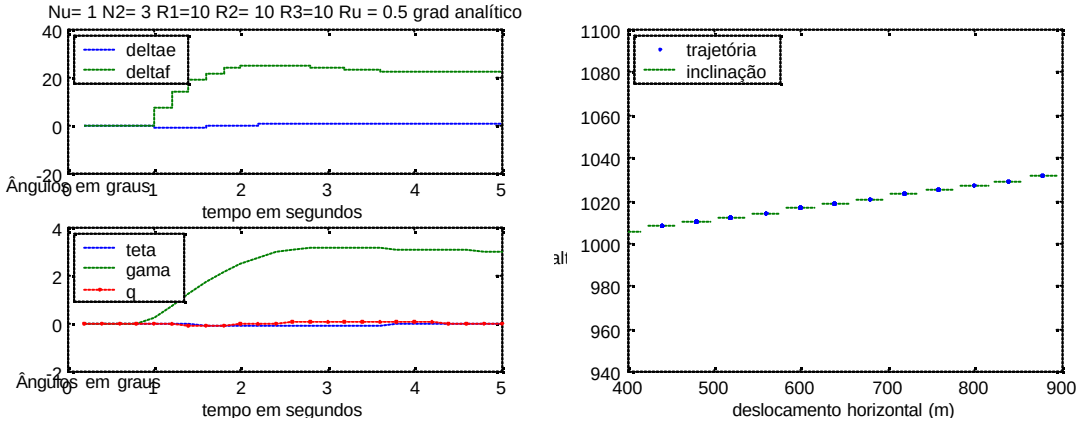


FIGURA. 6.26 - “Altitude Rate de 3 graus com R2 = 10 (a) controles e saídas e (b) trajetória com a inclinação (“pitch”) da aeronave.

A Figura 6.27 mostra o caso onde a rede B falha em horizontes mais longos, isto se deve ao fato da rede B não ter sido treinada no padrão da rede C e conseqüentemente a sua propagação, acumulando erros, perde a representatividade do modelo do sistema.

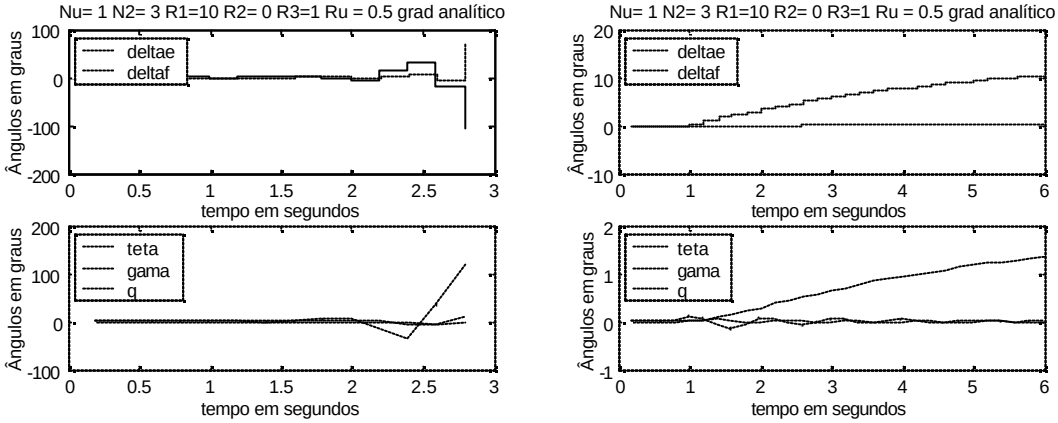


FIGURA. 6.27 - "Altitude Rate de 2 graus com a rede B e C para Nu=1 e N2=3: (a) instabilidade na rede C e (b) performance adequada na rede C.

Verifica-se que o conjunto de parâmetros é relevante para a resposta do sistema e não tomados de forma isolada, o que é de se esperar, pois a rede é não linear. Em geral, horizontes curtos produzem transitórios mais rápidos, horizonte de controle curto favorece a estabilidade aliada a um horizonte de predição mais longo e o horizonte de controle mais longo favorece a robustez em detrimento da estabilidade.

Concluindo, verificou-se que o esquema de controle preditivo neural apresentou comportamento compatível com modelo de controle preditivo ideal do Capítulo 4, ainda a possibilidade de estabelecer quaisquer componentes de saída e controle de sistemas instáveis.

CAPÍTULO 7

CONCLUSÕES E RECOMENDAÇÕES

Neste trabalho foram apresentados os resultados de pesquisa tecnológica para obtenção de controle configurado de aeronaves com uso de redes neurais. Para tanto, verificou-se inicialmente a possibilidade de utilização de controle preditivo convencional, para controle configurado nas dinâmicas longitudinal e latero-direcional em modelos lineares. Demonstrada a viabilidade de controle configurado com abordagem por controle preditivo, sem necessidade de atribuição de auto-estrutura, desenvolveu-se e testou-se um esquema de controle preditivo neural.

O propósito inicial foi verificar a aplicação do formalismo de controle preditivo com a finalidade de desacoplamento de modos, onde o modelo da planta era idêntico à planta. A partir de alocação de pólos trivial, obteve-se um sistema estabilizado, porém acoplado. Introduzindo o controle preditivo para seguir saídas de referência pré-estabelecidas, obtiveram-se as respostas desejadas e conseqüentemente o desacoplamento.

A verificação da viabilidade de controle preditivo deu sustentação para a utilização de redes neurais, tendo as perspectivas de se ter esquemas de tempo real com capacidade de adaptação “on-line”, permitida pela capacidade de aprendizado das redes neurais.

Outrossim, a condição de vôo da aeronave do problema abordado, levaria um modelo não linear e completo a comportar-se da mesma forma do modelo “trimado”, portanto é mais razoável treinar a rede para estes modelos mais simples em condição de vôo, do que treinar para um modelo completo, com uma rede mais complexa e de carga computacional considerável atualmente.

Algumas arquiteturas de rede foram exploradas e a mais conveniente baseou-se em se utilizar o menor número de saídas atrasadas na entrada. O padrão de treinamento teve as

finalidades de captar o transitório e excitar as entradas de controle. Vale ressaltar que, como a rede é normalizada, não é eficiente aumentar a escala das entradas de interesse, pois a saturação dos neurônios deteriora o desempenho.

A tolerância de treinamento adequada para que os Jacobianos fossem calculados com precisão suficiente para extrapolação foi obtida com treinamentos e testes de validação do Jacobiano, ou seja, a partir de um ponto extrapolava-se o valor do passo seguinte.

O desenvolvimento analítico para obtenção do Jacobiano da rede e após o gradiente da função de desempenho do controle e suas implementações forneceram robustez ao sistema como demonstrado pelos resultados no Capítulo 6. Como visto também, a sintonia do controle depende de vários parâmetros e conjunto com a característica da rede treinada, pois apesar treinamento eficiente e validação aceitável, os ganhos serão diferentes.

Quanto a sugestões para trabalhos futuros:

- Adição de perturbações

Utilizar modelos com perturbação em sinal e estrutural do sistema, para verificação de robustez.

- Estudo de sensitividade da rede

A validação de identificadores neurais (Corrêa et al., 1999) a partir de sistemas caóticos tem apresentado resultados em termos de formalizar o problema de sensitividade, por exemplo, o caso abordado neste trabalho em relação à rede “A”, que apesar de atingir a tolerância não foi satisfatória como identificador.

- Formalização do tratamento de redes utilizando-se o Jacobiano para estudos de estabilidade local.

Desde que o Jacobiano é uma linearização local da rede, a idéia é estudar a possibilidade de aplicação de ganhos principais (Maciejowski, 1989) para estabelecer um formalismo para a análise da performance e robustez e estudo de estabilidade avançada (Slotine e Li, 1991) visando generalização.

- Estudo de função de transferência generalizada.

Estudo de abordagem de função de transferência generalizada (Fung et al., 1997), para redes neurais. Estudo do formalismo para explorar a possibilidade de análise de resposta em frequência da rede e determinação de margens de estabilidade.

- Simplificação dos cálculos do Jacobiano e gradiente.

Estudo da dinâmica do gradiente para estabelecer uma aproximação analítica robusta e conseqüente melhora em desempenho computacional. Alternativa não explorada neste trabalho é a utilização de outras arquiteturas e funções de ativação que simplificariam a determinação do Jacobiano da rede (Curvo, 2001).

- Controle de sistemas instáveis.

Aplicação de controle preditivo neural em sistemas instáveis dentro de um formalismo que assegure uma avaliação de performance e robustez.

Concluindo, vale ressaltar que as sugestões de formalização para estabilidade, performance e robustez são limitadas em regiões de operação típicas, ou seja, trata-se de análise local e não global.

- Computação de alto desempenho

Estudo de implementação do algoritmo dentro do paradigma de paralelismo, que é bem adequado para sistemas conexionistas paralelos como as redes neurais.

REFERÊNCIAS BIBLIOGRÁFICAS

Andry, A.,N.; Shapiro, E., Y.; Chung, J.,C. Eigenstructure assignment for linear systems. **IEEE Transactions on Aerospace and Electronic Systems**, v. l. AES-19, n. 5, p.711-729, Sept. 1983.

Botto, M. A.; da Costa, J. S. A comparison of nonlinear predictive control techniques using neural network models. **Journal of Systems Architecture**. v. 44, p. 597-616, 1998.

Broyden, C.G. The convergence of a class of double-rank minimization algorithms. **Journal Inst. Math. Applic**., v. 6, p. 76-90, 1970.

Corrêa, M.V.; Aguirre, L.A.; Braga, A.P. Validação de modelos neurais identificados a partir de um sistema caótico. In: Brazilian Conference on Neural Networks, 4.; Congresso Brasileiro de Redes Neurais, 4., July 20-22, 1999, São José dos Campos. . **Proceedings.** São José dos Campos: **.** ITA, p. 152-157. 1999.

Curvo, M. **Redes neurais artificiais e estimação de parâmetros aplicados ao problema de modelagem e controle estocástico adaptativo de aeronaves de alto desempenho**. São José dos Campos. 170 p. (INPE-8700 -TDI/793). Tese (Doutorado em Engenharia e Tecnologia Espacial) – Instituto de Pesquisas Espaciais, 2001.

Faller, W., E.; Schreck, S., J. Real-time prediction of unsteady aerodynamics: application for aircraft control and maneuverability enhancement. **IEEE Transactions on Neural Networks**, v. 6, n. 6, p.1461-1468, Nov. 1995.

Fletcher, R.. A new approach to variable metric algorithms. **Computer Journal**, v. 13, p. 317-322, 1970.

Fung, C.F.; Billings, S.A.; Zhang, H. Generalised transfer functions of neural networks. **Mechanical Systems and Signal Processing**, v. 11, n. 6, p. 843-868, 1997.

Gili, P. A.; Battipede, M. Adaptive neurocontroller for a nonlinear combat aircraft model. **Journal of Guidance, Control and Dynamics**, v. 24, n. 5, p. 910-917, Sept.-Oct. 2001.

Goldfarb, D. A Family of variable metric updates derived by variational means. **Mathematics of Computing**, v. 24, p. 23-26, 1970.

Gomm, J., B.; Evans, J., T.; Williams, D. Development and performance of a neural-network predictive controller. **Control Engeneering Practice**, v. 5, n. 1, p. 49-59, 1997.

Gosh, A.,K.; Raisinghani, S.C.; Khubchandani, S. G. Estimation of aircraft lateral-directional parameters using neural networks. **Journal of Aircraft**, v. 35, n. 6, p. 876-881, Nov.-Dec. 1998.

Hunt, K.,J. ; Sbarbaro, D.; Zbikowski, R.; Gawthrop, P.J. Neural networks for control systems: a survey. **Automatica**, v. 28, n. 6, p. 1083-1112, 1992.

Kawato, M.; Uno, Y.; Isobe, M.; Suzuki, R. Hierarchical neural network model for voluntary movement with application to robotics. **IEEE Control Systems Magazine**, v. 8, n. 8, p. 8-17, 1988.

Klein, G.; Moore, B.C. Eigenvalue: Generalized eigenvector assignment with state feedback. **IEEE Transactions on Automatic Control**, v. AC22, n. 1, p. 140-141, Feb. 1977.

Lee, T. ; Kim, Y. nonlinear adaptive flight control using back stepping and neural networks controller. **Journal of Guidance, Control and Dynamics**, v. 4, p. 675-682, July-Aug. 2001.

Liebst, B.,S.; Garrard, W., L.; Adams, W., M. Design of an active flutter suppression system. **Journal of Guidance**, v .9, n. 1, p. 64-71, Jan.-Feb. 1986.

Maciejowski, J.M. **Multivariable feedback design**. Cambridge: Addison-Wesley Publishing, 1989, 424 p.

Mayne, D.,Q.; Michalska, H. Receding horizon control of nonlinear systems. **IEEE Trans. on Automatic Control**, v. 35, n. 7, p. 814-824, July 1990.

Narendra, K.; Parthasarathy, K. Identification and control of dynamical systems using neural networks. **IEEE Trans. on Neural Networks**, v. 1 , n. 1, p. 4-27, Mar. 1990.

Narendra, K. Neural networks for control: theory and practice**. Proceedings of the IEEE**, v. 84, n. 10, p. 1385-1406, Oct. 1996.

Nascimento Jr, C.L.; Takashi Yoneyama,T. **Inteligência artificial em controle e automação**, São Paulo: Edgard Blücher, 2000.

Nascimento Jr., C.L. **Artificial neural networks in control and optimization**. PhD Thesis, Control Systems Centre, University of Manchester Institute of Science and Technology (UMIST), Manchester, UK, 1994.

Nelder, J.A.; Mead, R. A simplex method for function minimization. **Computer Journal**, v .7, p. 308-313, 1965.

Omatu, S.; Khalid, M.; Yusof, R. **Neuro-control and its applications**. London: Springer-Verlag, 1995, 255 p.

Otawara, K.; Fan, L.T. Synchronizing high-dimensional chaos by an artificial neural network. In: IEEE Conference on Decision and Control, 35., 11-13 Dec., Kobe, 1996. **Proceedings,** New York: IEE, v. 2, p. 2183-2184.

Phan, D., T.; Xing, L. **Neural networks for identification, prediction and control**. London: Springer-Verlag, 1995. 238 p.

Porter, B.; Crossley, R. **Modal control: theory and applications**. London: Taylor and Francis, 1972. 233 p.

Porter, B.; D'Azzo, J.J. Closed loop eigenstructure assignment by state feedback in multivariable linear systems. **International Journal of Control**, v. 27, n. 3, p. 487-492, 1978.

Porter, B.; D'Azzo, J.J. Algorithm for closed loop eigenstructure assignment by state feedback in multivariable linear systems. **International Journal of Control**, v. 27, n. 3, p. 943-947, 1978.

Psaltis, D.; Sideris, A.; Yamamura, A. A multilayered neural network controller. **IEEE Control System Magazine**, v. 8, p. 17-21, 1990.

Rios Neto, A. Design of a Kalman filtering based neural predictive control method. [CD-ROM]. In: Congresso Brasileiro de Automática, 8., (CBA 2000). Florianópolis, 11-14 Set. 2000. **Anais**. Florianópolis: UFSC/SBA 2000.

Saerens, M.; Soquet, A. A neural controller based on backpropagation algorithm. In: IEE International Conference on Artificial Neural Networks, 1., London, 1989. **Proceedings**, (Conf. Publ. n. 313), London, UK, Oct. 1989. p. 211-215. (Conf. Publ. n. 313).

Shanno, D.F. Conditioning of quasi-Newton methods for function minimization. **Mathematics of Computing**, v. 24, p. 647-656, 1970.

Silva, J. A.; Rios Neto, A. Neural predictive satellite attitude control based on Kalman filtering algorithms. In: International Symposium in Space Dynamics, Biarritz, France, 26-30 June 2000. **Proceedings**. CNES, p. 565-574. (INPE-9261-PRE/4930).

Silva, J. A. **Controle preditivo utilizando redes neurais artificiais aplicado a veículos aeroespaciais**. São José dos Campos. 229 p. (INPE-8480 -TDI/778). Tese (Doutorado em Engenharia e Tecnologia Espacial) – Instituto de Pesquisas Espaciais, 2001.

Siouris, G., M.; Lee, J.,G.; Choi, J.,W. Design of a modern pitch pointing control system. **IEEE Trans. on Aerospace and Electronic Systems**. v. 31, n. 2, p.730-738, Apr.1995.

Slotine, J.J.; Li, W. **Applied nonlinear control**. New Jersey: Prentice-Hall, 1991, 459 p.

Snyder, S., D.; Tanaka, N. Active control of vibration using neural network. **IEEE Trans. on Neural Networks**, v. 6, n. 4, p. 819-828, July 1995.

Sobel, K.M; Shapiro, E.Y. Eigenstructure assignment for design of multimode flight control systems. **IEEE Control Systems Magazine**. p. 9-14, May 1985.

Sobel, K.M.; Shapiro, E.Y. A design methodology for pitch pointing flight control systems. **Journal of Guidance**, n. 2, p. 181-187, Mar.-Apr. 1985.

Sobel, K., M.; Shapiro, E.,Y.; Andry, A., N., Jr. Eigenstructure assignment. **International Journal of Control**, v. 59, n .1, p.13-37, 1994.

Sorensen, P., H.; Norgaard, M.; Ravn, O.; Poulsen, N., K. Implementation of neural network based non-linear predictive control. **Neurocomputing**, v.28, p. 37-51, 1999.

Srinathkumar, S. Eigenvalue / eigenvector assignement using output feedback. **IEEE Trans. on Automatic Control**, v. AC-23, n. 1, p. 79-81, Feb. 1978.

Sun, Q.; Alouani, A., T. Linear system state estimation using Hopfield net. In: SSST/CSA 92, Southheastern Symposium on System Theory, 24., Annual Symposium on Communications, Signal Processing Expert Systems, and ASIC VLSI Design, 3., **Proceedings**, Greensboro, NC: Agricultural and Technical State University, 1992. p. 198-202.

Turner, P.; Montague, G.; Morris, J. Dynamic neural networks in nonlinear predictive control : an industrial application. **Computer Chemical Engineering**, v. 20, p. S937-S942, 1996.

Werbos, Pj., J. Overview of designs and capabilities. In: _____ **Neural networks for control**, Cambridge, MA: MIT Press, 1990. p. 59-65.

Zamarreño, J., M.; Vega, P. Neural predictive control. application to a highly non-linear system. **Engineering Applications of Artificial Intelligence**. v. 12, p. 149-158, 1999.

Zhan, J.; Ishida M. The multi-step predictive control of nonlinear siso process with a neural model predictive control (nmpc) method. **Computer Chemical Engineering**, v. 21, n .2, p. 201-210, 1997.

Zhao, H.; Guiver, J.; Neelakantan, R.; Biegler, L.,T. A nonlinear industrial model predictive controller using integrated pls and neural net state-space model. **Control Engineering Practice**, v. 9, p. 125-133, 2001.

Zurada, J. M**. Introduction to artificial neural systems**. St. Paul, MN: West, 1992. 679 p.

APÊNDICE A

SÍNTESE POR ATRIBUIÇÃO DE AUTO-ESTRUTURA

O problema de atribuição de auto-estrutura consiste em alocação de autovalores e autovetores desejados simultaneamente (auto-estrutura) por realimentação, que será mostrado a seguir, e em seguida determinar o controle direto, baseado em modelo de referência para a entrada de referência desejada presente no trabalho de Sobel e Shapiro (Sobel e Shapiro, 1985).

Considere-se um sistema linear (Equação A.1) multivariável e invariante no tempo:

$$\begin{aligned} \dot{x} &= Ax + Bz \\ y &= Cx \end{aligned} \qquad \text{(A. 1)}$$

onde:

$x \in R^n, \quad z \in R^m, \quad y \in R^r$

Assume-se que:

$$\begin{aligned} &\text{rank } [B] = m \\ &\text{rank } [C] = r \end{aligned} \qquad \text{(A. 2)}$$

A solução geral (Porter, B. e Crossley, R., 1972) para sistemas não forçados (sem perda de generalidade) é:

$$x(t) = \exp(At)\, x(0) \qquad \text{(A. 3)}$$

$x(0) = x(t) \quad \text{para} \quad t = 0$

A é uma matriz de ordem n e tem-se que:

$$A\mathbf{u_i} = \lambda_i \mathbf{u_i}\ , \qquad i = 1,2,...n \tag{A. 4}$$

onde λ_i e $\mathbf{u_i}$ são respectivamente os autovalores e autovetores associados. Assumindo que os autovalores de A são distintos, pode-se escrever a Equação A.4 da forma:

$$[A - \lambda_i I_n]\mathbf{u_i} = 0, \quad i = 1,2,...n \tag{A. 5}$$

onde I_n é a matriz identidade. A solução terá solução não-nula $\mathbf{u_i}$ $(i = 1,2,....,n)$, desde que:

$$\left|A - \lambda_i I_n\right| = 0, \quad i = 1,2...n \tag{A. 6}$$

e os autovalores são obtidos como raízes da equação característica (Equação A.6). Por outro lado, a matriz A^T tem um significado importante em análise modal. Se A^T possui autovalores distintos μ_j, $(j = 1,2,...,n)$, e autovetores ν_j, $(j = 1,2,...,n)$ pode-se afirmar que:

$$A^T \nu_j = \mu_j \nu_j\ , \qquad j = 1,2,....,n \tag{A. 7}$$

e vale a relação semelhante à

$$\left|A^T - \mu_i I_n\right| = 0, \quad i = 1,2...n \tag{A. 8}$$

porém, como sabe-se que em geral para uma dada matriz, $\left|X^T\right| = \left|X\right|$, tem-se:

$$\left|A - \mu_i I_n\right| = 0, \quad i = 1,2...n \tag{A. 9}$$

concluindo-se que os autovalores λ_i e μ_i são iguais, porem não necessariamente os autovetores u_i e ν_j serão iguais. Da Equação A.7 tem-se:

$$A^T v_j = \lambda_j v_j , \quad j = 1,2,....,n \tag{A. 10}$$

Transpondo-se a Equação A.10:

$$v_j^T A = \lambda_j v_j^T , \quad j = 1,2,....,n \tag{A. 11}$$

pós multiplicando a Equação A.11 por u_i , $i \neq j$, tem-se:

$$v_j^T A \, \mathbf{u_i} = \lambda_j v_j^T \mathbf{u_i} , \quad i \neq j, \quad i = 1,2,....,n, \quad j = 1,2,....,n \tag{A. 12}$$

De maneira similar, se cada membro da Equação A.4 for pré-multiplicado por v_j^T tem-se:

$$v_j^T A \, \mathbf{u_i} = \lambda_i v_j^T \mathbf{u_i} , \quad i \neq j, \quad i = 1,2,....,n, \quad j = 1,2,....,n \tag{A. 13}$$

Subtraindo a Equação A.13 da Equação A.12, tem-se:

$$(\lambda_i - \lambda_j) \, v_j^T \mathbf{u_i} = 0 , \quad i \neq j, \quad i = 1,2,....,n, \quad j = 1,2,....,n \tag{A. 14}$$

o qual implica que:

$$v_j^T \mathbf{u_i} = 0 , \quad i \neq j, \quad i = 1,2,....,n, \quad j = 1,2,....,n \tag{A. 15}$$

O que significa que os autovetores de A e A^T correspondentes a autovalores diferentes são ortogonais; sendo normalizados, para o caso de mesmos autovalores, isto é, $\lambda_i = \lambda_j$, tem-se que:

$$v_i^T \mathbf{u_i} = 1 , \quad i = 1,2,....,n \tag{A. 16}$$

Assim, as Equações A.15 e A.16 combinadas produzem a relação:

$$v_j^T \mathbf{u_i} = \mathbf{u_i}^T v_j = \delta_{ij} \quad (i, j = 1,2,....,n) \tag{A. 17}$$

onde δ_{ij} é o delta de Kronecker.

Estas bases vetoriais são denominadas “recíprocas”. Propriedades podem ser derivadas a partir das relações anteriores:

Sejam as matrizes modais U, V de A e A^T respectivamente:

$$U = [\mathbf{u_1} \quad \mathbf{u_2} \quad ..... \quad \mathbf{u_n}] \tag{A. 18}$$

$$V = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad ..... \quad \mathbf{v}_n] \tag{A. 19}$$

e a matriz de autovalores de A e A^T:

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & 0 & ..... & 0 \\ 0 & \lambda_2 & 0 & ..... & 0 \\ & & ... & & \\ & & & ... & \\ 0 & 0 & ..... & & \lambda_n \end{bmatrix} = \text{diag}\{\lambda_1 \quad \lambda_2 \quad ... \quad \lambda_n\} \tag{A. 20}$$

Pode-se obter ainda:

$$\begin{aligned} AU &= U\,\Lambda \\ A^T V &= V\,\Lambda \\ V^T U &= I_N \end{aligned} \tag{A. 21}$$

Segue da Equação A.21:

$$\begin{aligned} V^T &= U^{-1} \\ V &= (U^{-1})^T \\ U^{-1} A\, U &= \Lambda \\ U\,\Lambda\, U^{-1} &= A \end{aligned} \tag{A. 22}$$

Utilizando-se destes resultados, pode-se transformar o sistema para coordenadas modais; partindo de:

$$x(t) = U\ \xi(t) \tag{A. 23}$$

o sistema pode ser transformado para:

$$\dot{\xi}(t) = U^{-1}AU\ \xi(t) \tag{A. 24}$$

onde:

$$\xi_0 = U^{-1}x(0) \tag{A. 25}$$

$$\xi_0 = V^{T}x(0) \tag{A. 26}$$

$$\dot{\xi}(t) = \Lambda\ \xi(t) \tag{A. 27}$$

o que implica:

$$\dot{\xi}_i(t) = \lambda_i\ \xi_i(t)\,, \quad i = 1,2,....,n \tag{A. 28}$$

onde o sistema transformado está desacoplado (diagonal), a solução fica:

$$\dot{\xi}_i(t) = \xi_i(0)\ \exp\ (\lambda_i\ t)\,, \quad i = 1,2,...,n \tag{A. 29}$$

Transformando o sistema para as coordenadas originais tem-se:

$$x(t) = U\ \xi(t) = [\mathbf{u_1}\quad \mathbf{u_2}\quad .....\quad \mathbf{u_n}] \begin{bmatrix} \xi_1(t) \\ \xi_2(t) \\ ..... \\ ..... \\ \xi_n(t) \end{bmatrix} \tag{A. 30}$$

o que implica em:

$$x(t) = \mathbf{u_1}\xi_1(0)\exp(\lambda_1 t) + \mathbf{u_2}\xi_2(0)\exp(\lambda_2 t) + ... + \mathbf{u_n}\xi_n(0)\exp(\lambda_n t) \quad \text{(A. 31)}$$

Para $t = 0$, tem-se:

$$x(0) = \mathbf{u_1}\xi_1(0) + \mathbf{u_2}\xi_2(0) + ... + \mathbf{u_n}\xi_n(0) \quad \text{(A. 32)}$$

Partindo da Equação A.25, obtém-se:

$$\xi_i(0) = \nu_i^T x(0), \quad (i = 1,2,....,n) \quad \text{(A. 33)}$$

A Equação A.32 fica então, utilizando-se a Equação A.33:

$$x(t) = \mathbf{u_1}\nu_1^T x(0)\exp(\lambda_1 t) + \mathbf{u_2}\nu_2^T x(0)\exp(\lambda_2 t) + ... + \mathbf{u_n}\nu_n^T x(0)\exp(\lambda_n t) \quad \text{(A. 34)}$$

que na forma compacta fica:

$$x(t) = \sum_{i=1}^{n}[\exp(\lambda_i t)]\,\mathbf{u_i}\,\nu_i^T x(0) \quad \text{(A. 35)}$$

A Equação A.35 mostra claramente que a dinâmica contínua não forçada do sistema descrito pela Equação A.1 é uma combinação linear de n funções da forma $[\exp(\lambda_i t)]\mathbf{u_i}$, $i = 1,2,...,n$, as quais descrevem os modos dinâmicos do sistema.

Assim, um modo é determinado pelo seu autovetor associado $\mathbf{u_i}$ e sua característica temporal associada ao autovalor λ_i.

Seja agora, o controle $z(t)$ aplicado no sistema representado pela Equações A.36 e A.37:

$$\dot{x} = Ax + Bz \quad \text{(A. 36)}$$

$$y = Cx \tag{A. 37}$$

com $x \in R^n$, $z \in R^m$, $y \in R^r$.

A estratégia de controle consagrada para alocação de autovalores pode ser obtida via realimentação de saída, estado ou ambos; uma abordagem relevante pode ser obtida inclusive com os teoremas inerentes (Andry, A.,N.,Jr., Shapiro, E.Y., Chung, J.,C., 1983).

- **Realimentação de estado**

Seja o sistema:

$$\dot{x} = Ax + BKx \tag{A. 38}$$

$$y = Cx \tag{A. 39}$$

e a Equação A.38 fica:

$$\dot{x} = Ax + BKx = (A + BK)x \tag{A. 40}$$

Dado um conjunto de autovalores desejados $\{\lambda_i^d\}$, $i = 1,2,...,n$ e um conjunto de autovetores desejados $\{\mathbf{u}_i^d\}$, $i = 1,2,...,n$, deve-se determinar uma matriz K (m x n) tal que o conjunto de autovalores $\{\lambda_i^d\}$ seja um subconjunto dos autovalores de $A + BK$ e o conjunto de autovetores $\{\mathbf{u}_i^d\}$ possua a menor distância euclidiana dos autovetores de $A + BK$, ou seja projetando-se os autovetores desejados no subespaço cuja base são as colunas de $(\lambda_i I - A)^{-1}B$, para $i = 1,2,....,n$.

- **Realimentação de saída**

Seja o sistema:

$$\dot{x} = Ax + BFy \tag{A. 41}$$

$$y = Cx \tag{A. 42}$$

A equação A.41 fica:

$$\dot{x} = Ax + BFCx = (A + BFC)x \tag{A. 43}$$

Dado um conjunto de autovalores desejados $\{\lambda_i^d\}, i = 1,2,...,r$ e um conjunto de autovetores desejados $\{\mathbf{u}_i^d\}, i = 1,2,...,r$, deve-se determinar uma matriz F (m x r) tal que o conjunto de autovalores $\{\lambda_i^d\}$ seja um subconjunto dos autovalores de $A + BFC$ e o conjunto de autovetores $\{\mathbf{u}_i^d\}$ possua a menor distância euclidiana dos autovetores de $A + BFC$, ou seja projetando-se os autovetores desejados no subespaço cuja base são as colunas de $(\lambda_i I - A)^{-1}B$, para $i = 1,2,....,r$.

O objetivo é, observando a Equação A.43, reduzir ao máximo, senão for possível torná-las nulas, determinadas componentes do autovetor $\mathbf{u}_i$ a fim de reduzir ou eliminar a contribuição do autovalor λ_i na resposta temporal do sistema, ou seja, eliminar a influência de um determinado modo na dinâmica do sistema.

Teorema (Srinathkumar, S., 1978): Seja o sistema dado pela Equação A.1, controlável e observável e as matrizes B e C de "full rank" (Equação A.2), então $\max(m,r)$ autovalores de malha fechada podem ser atribuídos e $\max(m,r)$ autovetores podem ser parcialmente atribuídos com $\min(m,r)$ componentes em cada autovetor arbitrariamente escolhidas usando realimentação de saída, isto é, $z = Fy$.

Assim tem-se:

$$(A + BFC)u_i = \lambda_i u_i \tag{A. 44}$$

$$u_i = (\lambda_i I - A)^{-1} BFCu_i \tag{A. 45}$$

$$u_i = (\lambda_i I - A)^{-1} Bm_i = L_i m_i \tag{A. 46}$$

A partir dos autovetores atingíveis $\{u_i^a\}$:

$$u_i^a = L_i z_i \tag{A. 47}$$

formula-se a norma do erro:

$$J = \left\| u_i^d - u_i^a \right\|^2 = \left\| u_i^d - L_i z_i \right\|^2 \tag{A. 48}$$

e via mínimos quadrados obtém-se:

$$u_i^a = L_i (L_i^T \, L_i)^{-1} L_i^T \, u_i^d \tag{A. 49}$$

A partir de uma referência de saída desejada (“track”)

$$y_t = Dx \tag{A. 50}$$

obtém-se o controle com as componentes “feedforward” e “feedback”:

$$u = (\Omega_{22} + FC\Omega_{12}) u_c - Fy \tag{A. 51}$$

onde o ganho F é obtido da forma:

$$F = (Z - A_1 U) * (CU)^{-1} \tag{A. 52}$$

$$\begin{aligned} U &= [u_1^a \quad u_2^a \quad ... \quad u_r^a] \quad (n \times r) \\ Z &= [\lambda_1 z_1 \quad \lambda_2 z_2 \quad ... \quad \lambda_r z_r] \quad (m \times r) \\ A_1 &= [A_{11} \quad A_{12}] \end{aligned} \tag{A. 53}$$

APÊNDICE B

DETERMINAÇÃO DE JACOBIANO DA REDE NEURAL

A partir da Figura 2.4, repetida abaixo, serão obtidos, o Jacobiano da rede, a função de desempenho e o gradiente analítico.

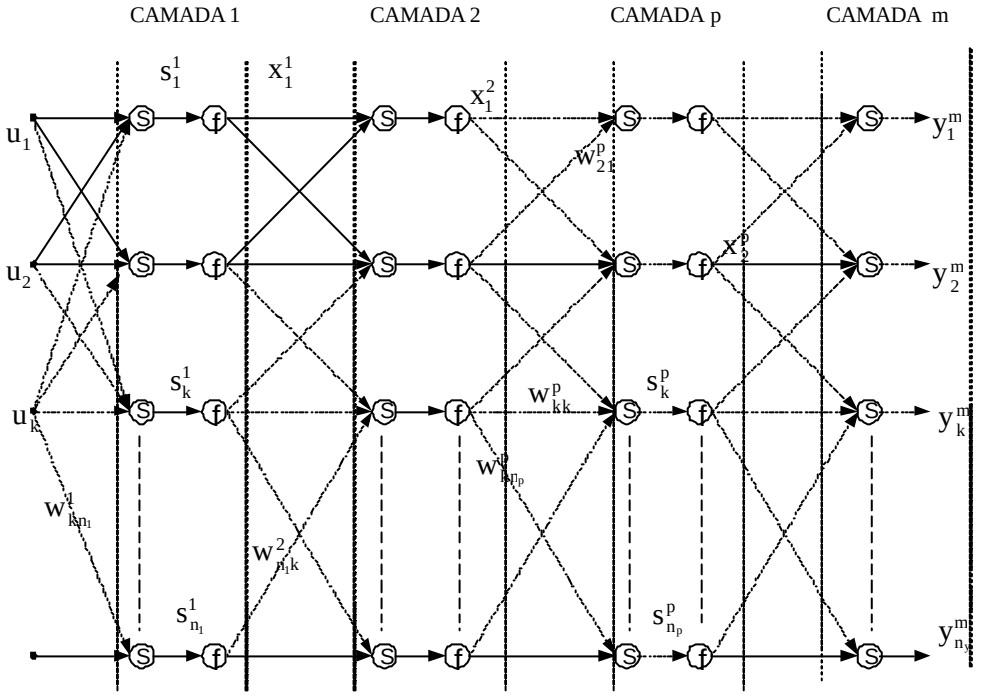


Fig 2.4 – Rede Multicamada “feedforward”

Seja a rede com entradas $\mathbf{u}(k), \mathbf{y}(k)$, onde:

$$\mathbf{u}(k) = [u_1(k), u_2(k), ..., u_{du}(k)] \qquad \text{(B. 1)}$$

$$\mathbf{y}(k) = [y_1(k),\ y_2(k), ..., \ y_{dy}(k)] \qquad \text{(B. 2)}$$

e saídas:

$$\mathbf{y}(k+1) = [y_1(k+1),\ y_2(k+1), ..., \ y_{dy}(k+1)] \qquad \text{(B. 3)}$$

Seja a matriz de ponderações (pesos sinápticos) na camada "p" da rede com "m" camadas totais, cujos pesos conectam as "n_{p-1}" saídas da camada "p-1" às n_p entradas da camada "p":

$$\mathbf{W}^p = \begin{bmatrix} w^p_{1,1} & w^p_{1,2} & w^p_{1,3} \ldots\ldots\ldots\ldots & w^p_{1,n_p} \\ w^p_{2,1} & w^p_{2,2} & w^p_{2,3} \ldots\ldots\ldots\ldots & w^p_{2,n_p} \\ & & & \\ w^p_{n_{p-1},1} & w^p_{n_{p-1},2} & w^p_{n_{p-1},3} \ldots\ldots\ldots\ldots & w^p_{n_{p-1},n_p} \end{bmatrix}_{(n_{p-1} \times n_p)} \qquad \text{(B. 4)}$$

Assumindo-se que:

Função de rede linear unitária na primeira e ultima camadas.

Função de rede f(net) não linear, contínua, de classe C^1 (1as. derivadas contínuas) em todos os neurônios.

Pesos sinápticos já treinados "off-line" no modelo direto da planta.

Deseja-se determinar a sensitividade da saída "y" em relação à entrada (considerada genérica) "u" da rede, ou seja, o Jacobiano da rede. A partir da figura 2.4, para uma saída qualquer x_i^p em uma camada intermediaria p qualquer:

$$x_i^p = f(s_i^p) \qquad \text{(B. 5)}$$

$$s_i^p = \sum_{j=1}^{n_{p-1}} x_j^{p-1} \cdot w_{ji}^p \qquad \text{(B. 6)}$$

O objetivo é calcular para uma dada componente u_j da entrada, a sensitividade $\frac{\partial y_i^m}{\partial u_j}$ na camada de saída "m", e consequentemente, o Jacobiano da rede, definido como:

$$J(\mathbf{y}^m, \mathbf{u}) = \begin{bmatrix} \frac{\partial y_1^m}{\partial u_1} & \frac{\partial y_1^m}{\partial u_2} & \cdots\cdots\cdots & \frac{\partial y_1^m}{\partial u_{n_u}} \\ \frac{\partial y_2^m}{\partial u_1} & \frac{\partial y_2^m}{\partial u_2} & \cdots\cdots\cdots & \frac{\partial y_2^m}{\partial u_{n_u}} \\ . & . & . & . \\ . & . & . & . \\ \frac{\partial y_{n_y}^m}{\partial u_1} & \frac{\partial y_{n_y}^m}{\partial u_2} & \cdots\cdots\cdots & \frac{\partial y_{n_y}^m}{\partial u_{n_u}} \end{bmatrix}_{(n_y \times n_u)} \qquad \text{(B. 7)}$$

Cálculo das sensitividades para as camadas “1”, “2”, camada genérica “p” e camada de saída “m”:

Camada “1”:

$$\frac{\partial s_i^1}{\partial u_j} = \frac{\partial}{\partial u_j}\left(\sum_{k=1}^{n_u} u_k \cdot w_{ki}^1\right) = w_{ji}^1, \quad i = 1,2,.....,n_1 \qquad \text{(B. 8)}$$

$$\frac{\partial x_i^1}{\partial u_j} = f'\{s_i^1(\mathbf{u})\} \cdot w_{ji}^1, \quad i = 1,2,.....,n_1 \qquad \text{(B. 9)}$$

Camada “2”:

$$\frac{\partial s_i^2}{\partial u_j} = \sum_{k=1}^{n_1} \frac{\partial}{\partial u_j}(x_k^1 \cdot w_{ki}^2) = \sum_{k=1}^{n_1} \frac{\partial x_k^1}{\partial u_j} \cdot w_{ki}^2, \quad i = 1,2,.....,n_2 \qquad \text{(B. 10)}$$

$$\frac{\partial x_i^2}{\partial u_j} = f'\{s_i^2(\mathbf{u})\} \cdot \sum_{k=1}^{n_1} \frac{\partial x_k^1}{\partial u_j} \cdot w_{ki}^2, \quad i = 1,2,.....,n_2 \qquad \text{(B. 11)}$$

Camada “p”:

$$\frac{\partial s_i^p}{\partial u_j} = \sum_{k=1}^{n_{p-1}} \frac{\partial}{\partial u_j}(x_k^{p-1} \cdot w_{ki}^p) = \sum_{k=1}^{n_{p-1}} \frac{\partial x_k^{p-1}}{\partial u_j} \cdot w_{ki}^p, \quad i = 1,2,.....,n_p \qquad \text{(B. 12)}$$

$$\frac{\partial x_i^p}{\partial u_j} = f'\{s_i^p(\mathbf{u})\} \cdot \sum_{k=1}^{n_{p-1}} \frac{\partial x_k^{p-1}}{\partial u_j} \cdot w_{ki}^p \quad , \quad i = 1,2,.....,n_p \tag{B. 13}$$

Camada de saída “m”:

$$\frac{\partial y_i^m}{\partial u_j} = \sum_{k=1}^{n_{m-1}} \frac{\partial x_k^{m-1}}{\partial u_j} \cdot w_{ki}^m \quad , \quad i = 1,2,.....,n_y \tag{B. 14}$$

Na forma matricial:

Camada “1”:

$$\begin{bmatrix} \bar{s}_1^1 \\ \bar{s}_2^1 \\ \cdot \\ \cdot \\ \cdot \\ \bar{s}_{n_1}^1 \end{bmatrix}_{(n_1 \times 1)} = [\,\mathbf{W}^1_{(n_u \times n_1)}\,]^T * \begin{bmatrix} \bar{u}_1 \\ \bar{u}_2 \\ \cdot \\ \cdot \\ \cdot \\ \bar{u}_{n_u} \end{bmatrix}_{(n_u \times 1)} \tag{B. 15}$$

$$\mathbf{J}(\mathbf{x}^1,\mathbf{u})_{(n_1 \times n_u)} = [f'\{\mathrm{diag}(\bar{\mathbf{s}}^1)\}]_{(n_1 \times n_1)} * [\,\mathbf{W}^1_{(n_u \times n_1)}\,]^T \tag{B. 16}$$

Camada “2”:

$$\bar{\mathbf{x}}^1 = f\,(\,\bar{\mathbf{s}}^1\,) \tag{B. 17}$$

$$\begin{bmatrix} \bar{s}_1^2 \\ \bar{s}_2^2 \\ \cdot \\ \cdot \\ \cdot \\ \bar{s}_{n_2}^2 \end{bmatrix}_{(n_2 \times 1)} = [\,W^2_{(n_1 \times n_2)}\,]^T * \begin{bmatrix} \bar{x}_1^1 \\ \bar{x}_2^1 \\ \cdot \\ \cdot \\ \cdot \\ \bar{x}_{n_1}^1 \end{bmatrix}_{(n_1 \times 1)} \tag{B. 18}$$

$$\mathbf{J}(\mathbf{x}^2,\mathbf{u})_{(n_2\times n_u)} = [f'\{\mathrm{diag}(\bar{\mathbf{s}}^2)\}]_{(n_2\times n_2)} * [\ \mathbf{W}^2_{(n_1\times n_2)}\ ]^T * \mathbf{J}(\mathbf{x}^1,\mathbf{u})_{(n_1\times n_u)} \quad \text{(B. 19)}$$

Camada genérica “p”:

$$\bar{\mathbf{x}}^{p-1} = f\ (\ \bar{\mathbf{s}}^{p-1}\ ) \quad \text{(B. 20)}$$

$$\begin{bmatrix} \bar{s}^p_1 \\ \bar{s}^p_2 \\ \cdot \\ \cdot \\ \cdot \\ \bar{s}^p_{n_p} \end{bmatrix}_{(n_p\times 1)} = [\ \mathbf{W}^p_{(n_{p-1}\times n_p)}\ ]^T * \begin{bmatrix} \bar{x}^{p-1}_1 \\ \bar{x}^{p-1}_2 \\ \cdot \\ \cdot \\ \cdot \\ \bar{x}^{p-1}_{n_{p-1}} \end{bmatrix}_{(n_{p-1}\times 1)} \quad \text{(B. 21)}$$

$$J(\mathbf{x}^p,\mathbf{u})_{(n_p\times n_u)} = \left[f'\{\mathrm{diag}(\bar{\mathbf{s}}^p)\}\right]_{(n_p\times n_p)} * [\ W^p_{(n_{p-1}\times n_p)}\ ]^T * J(\mathbf{x}^{p-1},\mathbf{u})_{(n_{p-1}\times n_u)} \quad \text{(B. 22)}$$

Camada de saída “m”:

$$\bar{\mathbf{x}}^{m-1} = f\ (\ \bar{\mathbf{s}}^{m-1}\ ) \quad \text{(B. 23)}$$

$$\begin{bmatrix} \bar{y}^m_1 \\ \bar{y}^m_2 \\ \cdot \\ \cdot \\ \cdot \\ \bar{y}^m_{n_y} \end{bmatrix}_{(n_y\times 1)} = \mathbf{W}^m_{(n_y\times n_{m-1})} * \begin{bmatrix} \bar{x}^{m-1}_1 \\ \bar{x}^{m-1}_2 \\ \cdot \\ \cdot \\ \cdot \\ \bar{x}^{m-1}_{n_{m-1}} \end{bmatrix}_{(n_{m-1}\times 1)} \quad \text{(B. 24)}$$

$$\mathbf{J}(\mathbf{y}^m,\mathbf{u})_{(n_y \times n_u)} = [f'\{\mathrm{diag}(\overline{\mathbf{y}}^m)\}]_{(n_y \times n_y)} * [\mathbf{W}^m_{(n_{m-1} \times n_y)}]^T * \mathbf{J}(\mathbf{x}^{m-1},\mathbf{u})_{(n_{m-1} \times n_u)} \quad \text{(B. 25)}$$

Assim tem-se o Jacobiano do emulador, obtido iterativamente da entrada para a saída da rede.

APÊNDICE C

DEDUÇÃO DA FÓRMULA DE RECORRÊNCIA

(Equações 5.12 e 5.13)

Cada elemento da matriz $\mathbf{J_{yu}}$ será denominado como:

$$\mathbf{J}_{yu}(i+p,i+q) = \frac{\partial \hat{\mathbf{y}}(i+p)}{\partial \mathbf{u}(i+q)} \quad \text{com } p > q \tag{5.10}$$

e os Jacobianos intermediários $\mathbf{J_{yy}}$:

$$\mathbf{J}_{yy}(i+p,i+q) = \frac{\partial \hat{\mathbf{y}}(i+p)}{\partial \mathbf{y}(i+q)} \tag{5.11}$$

que correspondem aos Jacobianos de cada rede da Figura 5.2 da saída em relação às entradas que são as saídas atrasadas.

Para uma entrada de controle **u(i)**, propagada até N2, e para uma rede com entradas que são as saídas com genéricos "n" atrasos, tem-se para o primeiro passo, de acordo com a Figura C.1:

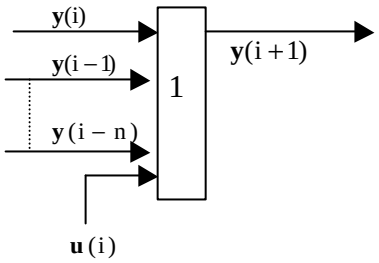


FIGURA. C.1 – 1º. estágio de propagação de u(i)

O Jacobiano no 1º. Estágio de propagação p=1 e q = 0, é (da Equação 5.10) $J_{yu}(i+1,i)$.

No segundo estágio de propagação, tem-se a Figura C.2:

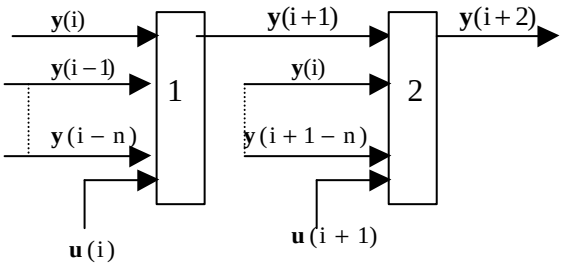


FIGURA. C.2 – 2º. estágio de propagação de u(i)

O Jacobiano $J_{yu}(i+2,i)$ é obtido (Equações 5.10 e 5.11) como:

$$J_{yu}(i+2,i) = J_{yy}(i+2,i+1) * J_{yu}(i+1,i) \qquad \text{(C. 1)}$$

Para o 3º. Estágio, da Figura C.3:

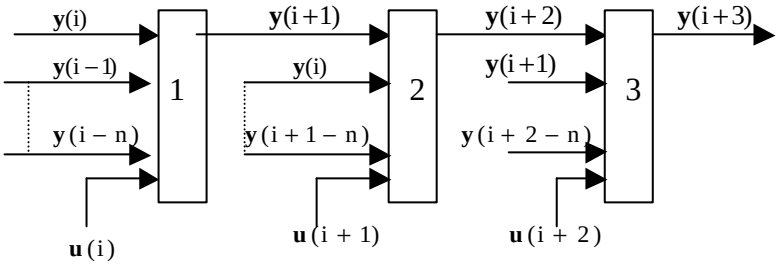


FIGURA. C.3 – 3º. estágio de propagação de u(i)

O Jacobiano $J_{yu}(i+3,i)$ é obtido como:

$$J_{yu}(i+3,i) = J_{yy}(i+3,i+2) * J_{yu}(i+2,i) + J_{yy}(i+3,i+1) * J_{yu}(i+1,i) \qquad \text{(C. 2)}$$

Para o p-ésimo estágio:

$$\begin{aligned} J_{yu}(i+p,i) = {} & J_{yy}(i+p,i+p-1) * J_{yu}(i+p-1,i) + \\ & J_{yy}(i+p,i+p-2) * J_{yu}(i+p-2,i) + \\ & J_{yy}(i+p,i+p-3) * J_{yu}(i+p-3,i) + ... \\ & J_{yy}(i+p,i+p-n) * J_{yu}(i+p-n,i) \end{aligned} \qquad \text{(C.3)}$$

que pode ser representada pela fórmula recursiva (Equação 5.12):

$$\mathbf{J}_{yu}(i+q+1,i+q)=\frac{\partial\hat{\mathbf{y}}(i+q+1)}{\partial\mathbf{u}(i+q)} \qquad \text{para} \qquad p = q+1,$$

$$\mathbf{J}_{yu}(i+p,i+q)=\sum_{j=p-1}^{p-n}\mathbf{J}_{yy}(i+p\,,i+j)*\mathbf{J}_{yu}(i+j\,,i+q)\,, \qquad \text{para } p > q+1 \qquad (5.12)$$

$j = p-1, p-2, ..., p-n$, com "j" decrescente, e

para $q = 0, ..., Nu-1$ e $p = q+1, ..., Nu$

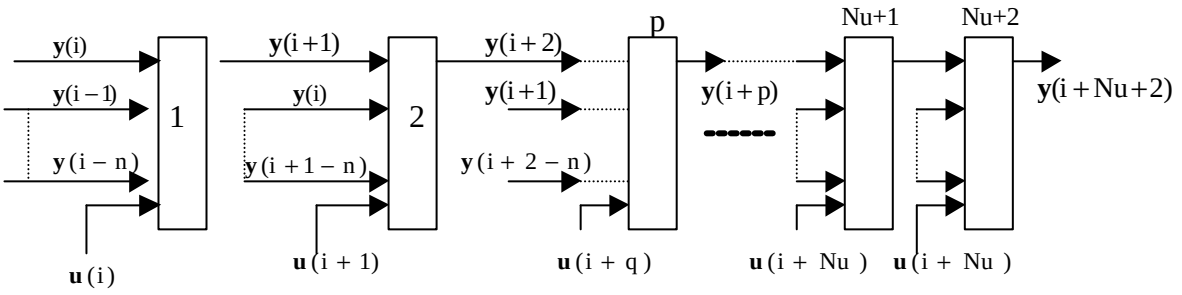


FIGURA. C.4 –Estágio de propagação de u(i+Nu)

Para os estágios onde $p > Nu$, o controle $\mathbf{u}(i + Nu)$ além de ser propagado a partir do estágio "Nu+1", contribui também a cada estágio até "N2", daí, à fórmula 5.12 é acrescido o termo referente a estas componentes para $p = Nu+2, ..., N2$ e $q = Nu$:

$$\mathbf{J}_{yu}(i+p,i+Nu)=\sum_{j=p-1}^{p-n}\{\mathbf{J}_{yy}(i+p\,,i+j)*\mathbf{J}_{yu}(i+j\,,i+Nu)\}+\mathbf{J}_{yu}(i+p,i+p-1) \qquad (5.13)$$

lembrando que:

$$\mathbf{J}_{yy}(i+p, i+q) = \mathbf{0} \quad \text{para} \quad p - q > n \qquad \text{e}$$

$$\mathbf{J}_{yu}(i+p, i+q) = \mathbf{0} \quad \text{para} \quad p \leq q$$

APÊNDICE D

ALGORITMOS

- Rotina de definição da arquitetura da rede, simulação e treinamento

```
function [net,y,P,TG,csys,dcsys,dt,tfinal,delayu,delayy,normaliz] = controlador_v07(d,dt,tfinal,simulink)
%   CONTROLE PREDITIVO COM REDES NEURAIS PITCH POINTING LINEAR
%   VERSAO COM 2 ENTRADAS CELL-ARRAY U(2X1) E Y(5X1) E 1 SAIDA CELL-ARRAY y(5X1)
%   VERSAO DE IDENTIFICAÇAO DE MODELO ""INTERNO"" DA PLANTA 09/05/2002
%   HIPOTESE DE CONTROLE DIRETO INVERSO PARA PREDIÇAO DE PLANTA
%   28/05/2001 - y(k+1 = f(y(k)...+.u(k))
%    1 CAMADA INTERNA 15 neuronios 05/05/2002
%    NORMALIZACAO DE DADOS 05/06/2002
%    INSERCAO DE SIMULACAO DE VALIDACAO 06/06/2002
%    CORRECAO DO ESQUEMA DE NORMALIZACAO, ESTAVA ERRADO ANTERIORMENTE 15/06/2002
%   [net,ysim,P,TG,csys,dt,tfinal,delayu,delayy,normaliz] = controlador_v02(d,dt,tfinal,simulink)
%   net_in =    rede treinada  se nao vazio [] simulacao, senao treinamento e simulacao
%   d   =       atraso de transporte em no. de intervalos "dt" =  d * dt
%   dt  =       discretizacao (seg)
%   delayu  =   atraso no canal de entrada de controle - vetor [0 1 2 ...]
%   delayy  =   atraso no canal de entrada das saidas atrasadas - vetor [0 1 2 ...]
%   tfinal  =   tempo de simulacao para treinamento da rede (seg)+
%   simulink=   geracao de modelo de rede em simulink (y/n)
%   ATUALIZACAO PARA CONTROLE EM 27/08/2002 - UTILIZAR PARA VALIDACAO JACOBIANO_VALIDACAO_V04
%   CORRECAO DE EMPILHAMENTO PARA RECENTE;ANTERIOR 28/08/2002

%   ARQUITETURA PARA AUMENTAR SENSITIVIDADE DO CONTROLE - 22/10/2002

%   ERRO DE DADOS DE TREINAMENTO - RETROFIT DE PILHA ETC.... 19_11_2002

%----------------------MODELO LINEAR LONGITUDINAL----------------------------------
%estado x=[teta q alfa deltae deltaf]'
```

```
%teta   - pitch
%q      - pitch rate
%alfa   - attack angle
%deltae - elevator deflection
%deltaf - flaperon deflection


%demanda de referencia ref =  [ deltaec deltafc]'
%deltaec - elevator deflection command
%deltaef - flaperon deflection command
%vetor de controle - u = [deltae deltaf]


A=[ 0  1        0        0       0
    0 -0.8693  43.223  -17.251  -1.5766
    0  0.9933  -1.3411  -0.1689 -0.2518
    0  0        0      -20       0
    0  0        0        0      -20];

B=[0 0;0 0;0 0;20 0;0 20];

%vetor de saida y =  [teta q gama deltae deltaf]

C= [1   0      0     0    0
    0   1      0     0    0
    1   0     -1     0    0
    0   0      0     1    0
    0   0      0     0    1];

D=zeros(5,2);
%
%polos de CL desejados
%eig(A);

p=[-5.6+4.2*j, -5.6-4.2*j, -1, -19, -19.5];

%   controle por realimentacao trivial de estado "full" sem atribuicao de autovetores.

K = place(A,B,p)
break
ABK = A-B*K;
%eig(ABK);
```

```
xInitial=[0 0 0 0 0];

t=0:dt:tfinal;

csys = ss(ABK,B,C,D);
dcsys = c2d(csys,dt);

%   sistema estabilizado sem atribuicao de autovetores
%   insercao da matriz de saida conforme saida desejada
%   limites para as entradas


tetamin   = deg2rad(-50); tetamax   = deg2rad(50);
qmin      = deg2rad(-30); qmax      = deg2rad(30);
gamamin   = deg2rad(-40); gamamax   = deg2rad(40);
deltaemin = deg2rad(-30); deltaemax = deg2rad(30);
deltafmin = deg2rad(-30); deltafmax = deg2rad(30);

plim = [deltaemin deltaemax
       deltafmin deltafmax
       tetamin tetamax
       qmin qmax
       gamamin gamamax
       deltaemin deltaemax
       deltafmin deltafmax];
%-------------------------------------------------------------------------

%----------------GERACAO DE ENTRADAS ALEATORIAS COM ZERO-HOLD------
---------------
st = length(t);
dtN = 5 * dt;
dcount = t(fix(st/8));
count = 0;
count1 = dcount;
count2 = 2*dcount;
count3 = 3*dcount;
count4 = 4*dcount;
count5 = 5*dcount;
count6 = 6*dcount;
for i = 1 : st
   if i > count
      if t(i) < count1
         u1 = 0;
```

```
            u2 = deltafmax * (2.*rand - 1);
        elseif t(i) < count2 & t(i) >= count1
            u1 = deltaemax * (2.*rand - 1);
            u2 = 0;
            %elseif t(i) < count3 & t(i) >= count2
            %u1 = deltaemax * (2.*rand - 1);
            %u2 = deltafmax/4;
            %elseif t(i) < count4 & t(i) >= count3
           %u1 = deltaemax * (2.*rand - 1);
           %u2 = deltafmin/4;
           %elseif t(i) < count5 & t(i) >= count4
           % u1 = deltaemax/4;
            %u2 = deltafmax * (2.*rand - 1);;
            %elseif t(i) < count6 & t(i) >= count5
           % u1 = deltaemin/4;
            %u2 = deltafmax * (2.*rand - 1);;
        else
            u1 = deltaemax * (2.*rand - 1);
            u2 = deltafmax * (2.*rand - 1);
        end
        count = i + dtN;
    end
    U1(1,i) = u1;
    U2(1,i) = u2;
end

U = [U1 ; U2]';
[Y,t,X] = lsim(csys,U,t);

%-------------------------------------------------------------------------------

%-----------------GERACAO DE PADROES DE TREINAMENTO EM
ESTRUTURAS--------------------

%   geracao de dados para treinamento PU = vetor de entradas
%   estrutura sequencial gerada PY (saidas atrasadas) en NDELAY atrasos
%   geracao de dados para treinamento T = vetor de target
%   estrutura sequencial gerada T
%   para cada conjunto de entradas atrasadas de DELAY, haver´a um target T = saida
em tempo presente

% SEMPRE MAIOR OU IGUAL A 1  [1 2 3 ....] NUNCA [0 1 ...] E DELAYY >
DELAYU
```

```
delayu = 1;
delayy = 2;

if delayy < delayu
   disp('erro de atrasos u>y')
   return
end

maxdelay = max(delayy,delayu);

T0     = t(maxdelay);          % INSTANTE INICIAL DE ENTRADAS E TARGETS DE
TREINAMENTO
NFINAL = st - maxdelay;
TFINAL = t(NFINAL);            % INSTANTE FINAL DE ENTRADAS E TARGETS
DE TREINAMENTO

aux=[];
uu=[];

Y_ = con2seq(Y');    %TRANSFORMACAO DE MATRIZ Y(dy  x N) EM CELULAS
Y_{i}, i = 1:N
U_ = con2seq(U');

auxy = [];auxu = [];
for k = 1:NFINAL
   for i = k:maxdelay+k-1
      auxy = [Y_{i};auxy];
      if i >= k+(maxdelay - delayu)
         auxu = [U_{i};auxu];
      end
   end
   Pa{2,k} = auxy;
   auxy =[];
   Pa{1,k} = auxu;
   auxu = [];
end
for i = 1:NFINAL - 1 - d       % ATRASO MINIMO = 1 ATRASO GENERICO 1+d (d
= n*DT)
   TG{1,i} = Y_{maxdelay+i+d};
   P{1,i}  = Pa{1,i};
   P{2,i}  = Pa{2,i};
end
```

```
 %  --------------NORMALIZACAO SIMETRICA DE DADOS----SE FOR NAO
SIMETRICA MUDAR Nmax E Nmin-----------------------------------
Nmax =  1;
Nmin = -1;


maxP1 = max(max([P{1,:}]));
maxP2 = max(max([P{2,:}]));
maxP  = max([maxP1 maxP2]);
maxT  = max(max([TG{1,:}]));

minP1 = min(min([P{1,:}]));
minP2 = min(min([P{2,:}]));
minP  = min([minP1 minP2]);
minT  = min(min([TG{1,:}]));

normaliz = [minP maxP minT maxT Nmin Nmax];   %SAIDA PARA VALIDACAO


plimn = (Nmax - Nmin) .* (plim - minP) ./ (maxP - minP) + Nmin;

Pn1  = (Nmax - Nmin) .* ([P{1,:} ]  - minP) ./ (maxP - minP) + Nmin;
Pn2  = (Nmax - Nmin) .* ([P{2,:}]  - minP) ./ (maxP - minP) + Nmin;
TGn1 = (Nmax - Nmin) .* ([TG{1,:}] - minT) ./ (maxT - minT) + Nmin;
for i = 1:length(P)
   Pn{i}  = [Pn1(:,i); Pn2(:,i)];
   P_{i}  = [P{1,i}; P{2,i}];
   TGn{i} = TGn1(:,i);
end
%-------------------------------------------------------------------------------

%-------------------------------------------------------------------------------

%------------------REDE NEURAL PARA MODELO DE "CSYS"--------------------------
------
%   ARQUITETURA
%   INSTANCIACAO, DEF. DE No. DE ENTRADAS E No. DE CAMADAS
net=network;

net.numInputs = 1;
net.numLayers = 3;
```

```
%   DEFINICAO DAS CONEXOES DE "BIAS" NAS CAMADAS
net.biasConnect = [1;1;1];
%   DEFINICAO DAS CONEXOES DAS ENTRADAS NAS CAMADAS
net.inputConnect = [1;0;0];
%   DEFINICAO DAS CONEXOES ENTRE CAMADAS - Jcamada CONECTADA
NA Iesima CAMADA
net.layerConnect = [0 0 0 ;1 0 0 ;0 1 0];
%   DEFINICAO DAS CONEXOES DE SAIDAS NAS CAMADAS - OBTER
SAIDAS NA Jesima CAMADA
net.outputConnect = [1 1 1];
%   DEFINICAO DA CAMADA DE SAIDA DE REFERENCIA PARA
TREINAMENTO COM "TARGET" - Jesima CAMADA
net.targetConnect = [0 0 1];

%   SUBOBJETOS
%   DEFINICAO DA DIMENSAO DA ENTRADA DA CAMADA 1(controle) E
SATURACOES
net.inputs{1}.size  =       size(U,2) * delayu + size(Y,2) * delayy;
%net.inputs{1}.range =       plimn(1:2,:);
%   DEFINICAO DA DIMENSAO DA ENTRADA DA CAMADA 2 (SAIDAS
ATRASADAS)
%net.inputs{2}.size =        size(Y,2) * Ny;
%net.inputs{2}.range =       plimn(3:7,:); %(irrelevante p/ dados normalizados)
%   DEFINICAO DO No. DE NEURONIOS DA CAMADA 1 (CONTROLE)
net.layers{1}.size =        12; %
%   DEFINICAO DA FT DA CAMADA 1
net.layers{1}.transferFcn = 'purelin';
%   DEFINICAO DA INICIALIZACAO PELO METODO "Nguyen-Widrow"
net.layers{1}.initFcn =     'initnw';
%   DEFINICAO DO No. DE NEURONIOS DA CAMADA 2 (SAIDAS
ATRASADAS)
net.layers{2}.size =        20; %
%   DEFINICAO DA FT DA CAMADA 2
net.layers{2}.transferFcn = 'tansig';
%   DEFINICAO DA INICIALIZACAO PELO METODO "Nguyen-Widrow"
net.layers{2}.initFcn =     'initnw';
%   DEFINICAO DO No. DE NEURONIOS DA CAMADA 3
net.layers{3}.size =        5;
%   DEFINICAO DA FT DA CAMADA 3
net.layers{3}.transferFcn = 'purelin';
%   DEFINICAO DA INICIALIZACAO PELO METODO "Nguyen-Widrow"
net.layers{3}.initFcn =     'initnw';
%   DEFINICAO DO No. DE NEURONIOS DA CAMADA 4 (SAIDA)
```

```
%net.layers{4}.size =         size(Y,2);
%   DEFINICAO DA FT DA CAMADA 4
%net.layers{4}.transferFcn = 'purelin';
%   DEFINICAO DA INICIALIZACAO PELO METODO "Nguyen-Widrow"
%net.layers{4}.initFcn =     'initnw';
%   DEFINICAO DOS TDL (TAPPED DELAY TIMES) NAS ENTRADAS
%   Jesima ENTRADA NA Iesima CAMADA

%   ALGORITMO TREINAMENTO LEVENBERG-MARQUARDT
net.trainFcn =              'trainlm';
%net.adaptFcn =              'trains';
%   CRITERIO DE ERRO - MEAN SQUARED ERROR
net.performFcn =            'mse';
%   INICIALIZACAO DA REDE
net = initlay(net);

%   PARAMETROS DE TREINAMENTO
net.trainParam.show     = 5;         % taxa de amostra de status
net.trainParam.epochs   = 5000;      % numero de passos de treinamento
net.trainParam.goal     = 1e-11;     % precisao para tolerancia
net.trainParam.mu_inc   = 1e+4;      % BUSCA - incremento de Mu - default = 10
net.trainParam.mu_dec   = 0.1;       % BUSCA - decremento de Mu - default = 0.1
net.trainParam.mu       = 0.005;     % BUSCA - Mu inicial - default = 0.001
net.trainParam.min_grad = 1e-10;     % BUSCA - GRAD - default = 1e-10
net.trainParam.mem_reduc= 1;         % BUSCA - memoria - default = 1

if simulink == 'y'
   gensim(net)
end

%--------------------TREINAMENTO-----------------------------------------------
[net,tr,ynet,yerrnet,Pf] = train(net,Pn,TGn);
%------------------------------------------------------------------------------

%--------------------GERACAO DE MODELO SIMULINK------------------------------
%------------------------------------------------------------------------------

%--------------------SIMULACAO DA REDE TREINADA------------------------------
ysim = sim(net,Pn);
%------------------------------------------------------------------------------

%--------------------FORMA VETORIAL DA SAIDA A PARTIR DE ESTRUTURA-----------
```

```
for i = 1:length(ysim)
   ysimn(:,i) = ysim{net.numLayers,i};
end
%-------------------------------------------------------------------------

%--------------------DESNORMALIZACAO DE DADOS----------------------------------
y = (maxT - minT) .* (ysimn - Nmin) ./ ( Nmax - Nmin) + minT;
%-------------------------------------------------------------------------

%DADOS NOVOS DEVEM PASSAR POR NORMALIZ/DENORMALIZ
%pnewn = tramnmx(pnew,minp,maxp);
%anewn = sim(net,pnewn);
%anew = postmnmx(anewn,mint,maxt);

%   SALVA NET
save net_new
%-------------------------------------------------------------------------------------
```

- Rotina de cálculo do Jacobiano

```
function [JJnet, Jnet, Ji, Jl] = jacobiano_v03(net, y)

%   calculo do jacobiano da rede em torno do ponto [ui(t),ui(t-1)...yi(t), yi(t-1)...]
%   para cada instante "Tk"
%   net eh a estrutura da rede treinada.

%   O Jacobiano J(y,U) eh calculado para cada entrada U = {y(k-1), y(k-2),...,u(k), u(k-
1),...}
%   HIPOTESES
%   - NAO EXISTE "LATCHES" --- JLAYER(i,j) = [] PARA j>i
%   - NAO EXISTE RECORRENCIA DE NOH --- JLAYER(i,i) = []
%   - TODAS AS ENTRADAS POSSUEM O MESMO "TAPPED DELAY LINE"

warning off
%-------------------------------------------------------------------------%
%   derivadas das funcoes da i-esima camada                               %
for i = 1:net.numLayers                                                   %
   deriv{i} = feval(net.layers{i}.transferFcn,'deriv');                   %
end                                                                       %
%-------------------------------------------------------------------------%

%------determinacao do Jacobiano de cada camada
%-----camada de entradas
% formacao do vetor de saida por camada Si


%determinacao das dimensoes das saidas de todas as camadas

%-------------------------------------------------------------------------

for i = 1:net.numLayers
   for j = 1:net.numInputs
      if ~isempty(net.IW{i,j})
         size_Iw(i,j) = size(net.IW{i,j},1);
      end
   end
end
for i = 1:net.numLayers
   for j = 1:net.numLayers
      if ~isempty(net.LW{i,j})
         size_Lw(i,j) = size(net.LW{i,j},1);
```

```
      end
   end
end

for i = 1:net.numLayers
   for j = 1:net.numInputs
      if ~isempty(net.IW{i,j})
         %for k = 1:size_Iw(i,j)
         Si{i} = feval( deriv{i}, NaN, y{i} );          %OBS: FEVAL(DPURELIN
CALCULA ERRADO FORNECE DIMENSAO 1 PARA ARRAYS???
            %end                                 %     O EFEITO A SEGUIR EH
IRRELEVANTE POIS EH UM PRODUTO UNITARIO
      end                                       %     CORRIGIDO DPURELIN EM 20/01/2003
   end
end
for i = 1:net.numLayers
   for j = 1:net.numLayers
      if ~isempty(net.LW{i,j})
         %for k = 1:size_Lw(i,j)
         Sl{i} = feval( deriv{i}, NaN, y{i} );
            %end
      end
   end
end
for i = 1:net.numLayers
   for j = 1:net.numInputs
      if ~isempty(net.IW{i,j})
         Ji{i,j} = diag(Si{i}) *  net.IW{i,j};         % SE FOR I OU 1 DAH NAO
ALTERA O RESULTADO
      end
   end
end
for i = 1:net.numLayers
   for j = 1:net.numLayers
      if ~isempty(net.LW{i,j})
         Jl{i,j} = diag(Sl{i}) *  net.LW{i,j};
      end
   end
end

%-----composicao dos Jacobianos das camadas para obtencao do Jacobiano global
%
%-----Jacobianos das camadas de entrada
```

```
%-----Jacobianos das camadas intermediarias e saida
iflag = 0;
for i = net.numLayers:-1:1
   for j = net.numLayers-1:-1:1
      if net.layerConnect(i,j)
         Jnet{i,j,j} = Jl{i,j};
         ii = i; jj = j; iflag = 1;
      end
   end
   if iflag, break,end
end

for i = net.numLayers:-1:1
   for j = net.numLayers-1:-1:1
      for k = j-1:-1:1
         soma = 0;
         for m = k+1:j
            if net.layerConnect(m,k)
              soma = soma + Jnet{i,j,m} * Jl{m,k};
            end
            Jnet{i,j,k} = soma;
         end
      end

   end
end

for i = 1:net.numLayers
   for j = 1:net.numInputs
      if net.inputConnect(i,j)
         JJnet{j} = Jnet{ii,jj,j} * Ji{i,j};
      end
   end
end
warning on
```

- Rotina de testes de validação da rede e do Jacobiano

```
function [y, y_Jac] =
Jacobiano_validacao_v06(net,normaliz,delayu,delayy,csys,d,dtnet,tfinal)

%---------------------------------VALIDACAO-------------------------------------
%   CORRECAO DE NORMALIZACAO 27/08/2002 - USAR COM
CONTROLADOR_V02
%   VERSAO A SER UTILIZADA COM PADRAO DE ENTRADA COM ATRASOS
EXPLICITADOS  18/07/2002
%   INSERIDO DT PARA JACOBIANO- DTJ
%   22/10/2002 - USAR COM CONTROLADOR V03 - ENTRADAS U E Y NA
MESMA CAMADA

tetamin   = deg2rad(-50); tetamax   = deg2rad(50);
qmin      = deg2rad(-30); qmax      = deg2rad(30);
gamamin   = deg2rad(-40); gamamax   = deg2rad(40);
deltaemin = deg2rad(-30); deltaemax = deg2rad(30);
deltafmin = deg2rad(-30); deltafmax = deg2rad(30);

%-----------------GERACAO DE ENTRADAS DE VALIDACAO--------------------------
-----
Ts = 50;
A = 1;
t = 0:dtnet:tfinal;
%   geraçao de entradas oscilatorias + aleatorias
U1 = A * deltaemax * 0.5 *((sin((2*pi/Ts)*t)) + sin((3*pi/Ts)*t) + 0.5 * (2.*rand - 1));
U2 = A * deltafmax * 0.5 *((sin((2*pi/Ts)*t + 0.5) + sin((5*pi/Ts)*t + 1)) + 0.5
*(2.*rand - 1));
U = [U1 ; U2]';
[Y,t,X] = lsim(csys,U,t);

%-------------------------------------------------------------------------------


%------------------GERACAO DE PADROES DE TREINAMENTO EM
ESTRUTURAS--------------------
%   geracao de dados para treinamento PU = vetor de entradas
%   estrutura sequencial gerada PY (saidas atrasadas) en NDELAY atrasos
%   geracao de dados para treinamento T = vetor de target
%   estrutura sequencial gerada T
%   para cada conjunto de entradas atrasadas de DELAY, haverah um target T = saida
em tempo presente
```

```
%   teste de jacobiano_v04 10/02/2003

maxdelay = max(delayy,delayu);

T0     = t(maxdelay);          % INSTANTE INICIAL DE ENTRADAS E TARGETS DE
TREINAMENTO
NFINAL = length(t) - maxdelay;
TFINAL = t(NFINAL);            % INSTANTE FINAL DE ENTRADAS E TARGETS
DE TREINAMENTO

aux=[];
uu=[];

Y_ = con2seq(Y');    %TRANSFORMACAO DE MATRIZ Y(dy x N) EM CELULAS
Y_{i}, i = 1:N
U_ = con2seq(U');

auxy = [];auxu = [];
for k = 1:NFINAL
   for i = k:maxdelay+k-1
      auxy = [Y_{i};auxy];
      if i >= k+(maxdelay - delayu)
         auxu = [U_{i};auxu];
      end
   end
   Pa{2,k} = auxy;
   auxy =[];
   Pa{1,k} = auxu;
   auxu = [];
end
for i = 1:NFINAL - 1 - d       % ATRASO MINIMO = 1 ATRASO GENERICO 1+d (d
= n*DT)
   TG{1,i} = Y_{maxdelay+i+d};
   P{1,i}  = Pa{1,i};
   P{2,i}  = Pa{2,i};
end

%----------------------------NORMALIZACAO DE DADOS --------------------------------
minP = normaliz(1);
maxP = normaliz(2);
minT = normaliz(3);
maxT = normaliz(4);
Nmin = normaliz(5);
```

```
Nmax = normaliz(6);

Pn1  = (Nmax - Nmin) .* ([P{1,:}]  - minP) ./ (maxP - minP) + Nmin;
Pn2  = (Nmax - Nmin) .* ([P{2,:}]  - minP) ./ (maxP - minP) + Nmin;
TGn1 = (Nmax - Nmin) .* ([TG{1,:}] - minT) ./ (maxT - minT) + Nmin;
for i = 1:length(P)
   Pn{i}  = [Pn1(:,i); Pn2(:,i)];
   TGn{i} = TGn1(:,i);
end

%-------------------------------------------------------------------------

%-------------------SIMULACAO DA REDE TREINADA-------------------------------
ysim = sim(net,Pn);%  eh estrutura de entrada para "sim(net)", onde os ATRASOS SAO
EXPLICITOS  18/07/2002

warning off
y_Jac = {[ysim{net.numLayers,:}]};
for j = 1:NFINAL-maxdelay
   for i = 1:net.numInputs
      if j > 1
         yaux = ysim(:,j-1);
         Jnet = Jacobiano_v04(net, yaux);    % Jacobiano em "Tk-1"
         yjac{i,j-1} = Jnet * ( Pn{i,j} - Pn{i,j-1} ); % J(Tk-1) * (U(Tk)-U(Tk-1))
      end
   end
   if j ==1
      y_Jac{j} = ysim{net.numLayers,j};
   else
      y_Jac{j} = ysim{net.numLayers,j-1} + yjac{i,j-1};   % y_j(Tk) = y_bar(Tk-1) +
Jdeltau(Tk-1)
   end
end
y_Jacn = [y_Jac{:}];
warning on
  %loop task == 'jacobiano'
%-------------------------------------------------------------------------
%-------------------FORMA VETORIAL DA SAIDA (4a camada) A PARTIR DE
ESTRUTURA-------------
for i = 1:length(ysim)
   ysimn(:,i) = ysim{net.numLayers,i};
end
%-------------------------------------------------------------------------
```

```
%--------------------DESNORMALIZACAO DE DADOS----------------------------------
y = (maxT - minT) .* (ysimn - Nmin) ./ ( Nmax - Nmin) + minT;
y_Jac = (maxT - minT) .* (y_Jacn - Nmin) ./ ( Nmax - Nmin) + minT;
%-------------------------------------------------------------------------------
%ESTATISTICAS
tt = size(t,1);
yy = size(y,2);
YY = size(Y,1);
yyj = size(y_Jac,2);

%------TESTES DE AMOSTRAS COM 95% DE NIVEL DE CONFIANCA ALFA = 0.05
for k = 1:net.layers{net.numLayers}.size   %ultimo layer como saida
   [h,ht,cv,ci]= KStest_cell('net outputs', 0.05, y(k,:));
end
for k = 1:net.layers{net.numLayers}.size   %ultimo layer como saida
   [h,ht,cv,ci]= KStest_cell('targets',0.05, Y(:,k));
end
for k = 1:net.layers{net.numLayers}.size   %ultimo layer como saida
   [h,ht,cv,ci]= KStest_cell('jacobian outputs',0.05, y_Jac(k,:));
end
%-------------------------------------------------------------------

%---- NET x TARGET
sizy = min(yy,YY);
ysum = sum(y);
Ysum = sum(Y(1+d+1:yy+d+1,:)');
ybar = ysum - ones(size(ysum,1),1)*mean(ysum);
Ybar = Ysum - ones(size(Ysum,1))*mean(Ysum);
sty = std(ybar);
stY = std(Ybar);
RyY = ybar * Ybar' / (sty*stY*(sizy -1));

%---- JACOBIAN x TARGET
yyj = size(y_Jac,2);
min_ploty = min(yyj,YY);
yjsum = sum(y_Jac);
yjbar = yjsum - ones(size(yjsum,1),1)*mean(yjsum);
stjy = std(yjbar);
RyjY = yjbar * Ybar(1:min_ploty)' / (stjy*stY*(min_ploty -1));

%---- JACOBIAN x NET
```

```
yyj = size(y_Jac,2);
min_ploty = min(yyj,yy);
yjsum = sum(y_Jac);
yjbar = yjsum - ones(size(yjsum,1),1)*mean(yjsum);
stjy = std(yjbar);
Ryjy = yjbar * ybar(1:min_ploty)' / (stjy*sty*(min_ploty -1));


%------------------SAIDAS GRAFICAS-------------------------------------------------------------
--------
y = rad2deg(y);Y = rad2deg(Y); U = rad2deg(U);
%figure;plot(t,U);legend('deltaec','deltafc');
%figure;plot(y(:,1:yy)'-
Y(1+d+1:YY,:));legend('tetaerr','qerr','gamaerr','deltaeerr','deltaferr');

figure;plot(y(:,1:yy)',Y(1+d+1:yy+d+1,:),'.');
title('correlation');xlabel('net output');ylabel('Target')
v = axis;
maxv = max(abs(v));
axis([-maxv maxv -maxv maxv]);
legend(['coef. correl.  = ', num2str(RyY,3)],2);

h = figure;plot(1:yy,y(1,1:yy)','k:',1:yy,Y(1+d+1:yy+d+1,1),'k-');
hold on
figure(h);plot(1:yy,y(2,1:yy)','r:',1:yy,Y(1+d+1:yy+d+1,2),'r-');
figure(h);plot(1:yy,y(3,1:yy)','b:',1:yy,Y(1+d+1:yy+d+1,3),'b-');
figure(h);plot(1:yy,y(4,1:yy)','g:',1:yy,Y(1+d+1:yy+d+1,4),'g-');
figure(h);plot(1:yy,y(5,1:yy)','m:',1:yy,Y(1+d+1:yy+d+1,5),'m-');
legend('teta net','teta','q net','q','gama net','gama','deltae net',...
    'deltae','deltaf net','deltaf')
xlabel('___ Target data       --- net outputs')
hold off
%-----------------------------------------------------------------------------------------------


y_Jac = rad2deg(y_Jac);
figure;plot(t,U);legend('deltaec','deltafc');
tt = size(t,1);
yy = size(y_Jac,2);
YY = size(Y,1);
min_ploty = min(yy,YY);
figure;plot(y_Jac(:,1:min_ploty)'-
Y(1+d+1:min_ploty+1,:));legend('tetaerr','qerr','gamaerr','deltaeerr','deltaferr');
```

```
figure;plot(y_Jac(:,1:min_ploty)',Y(1+d+1:min_ploty+1,:),'.');
title('correlation');xlabel('Jacobian');ylabel('Target data')
axis;
v = axis;
maxv = max(abs(v));
axis([-maxv maxv -maxv maxv]);
legend(['coef. correl.  = ', num2str(RyjY,3)],2);
h1 =
figure;plot(1:min_ploty,y_Jac(1,1:min_ploty)','k:',1:min_ploty,Y(1+d+1:min_ploty+1,1
),'k-');

hold on
figure(h1);plot(1:min_ploty,y_Jac(2,1:min_ploty)','r:',1:min_ploty,Y(1+d+1:min_ploty+
1,2),'r-');
figure(h1);plot(1:min_ploty,y_Jac(3,1:min_ploty)','b:',1:min_ploty,Y(1+d+1:min_ploty
+1,3),'b-');
figure(h1);plot(1:min_ploty,y_Jac(4,1:min_ploty)','g:',1:min_ploty,Y(1+d+1:min_ploty
+1,4),'g-');
figure(h1);plot(1:min_ploty,y_Jac(5,1:min_ploty)','m:',1:min_ploty,Y(1+d+1:min_ploty
+1,5),'m-');
legend('teta j','teta','q j','q','gama j','gama','deltae j',...
    'deltae','deltaf j','deltaf')
xlabel('___Target data   --- Jacobian outputs')
hold off

figure;plot(y_Jac(:,1:min_ploty)',y(:,1:min_ploty)','.');
title('correlation');xlabel('Jacobian');ylabel('Net')
v = axis;
maxv = max(abs(v));
axis([-maxv maxv -maxv maxv]);
legend(['coef. correl.  = ', num2str(Ryjy,3)],2);

h12 =
figure;plot(1:min_ploty,y_Jac(1,1:min_ploty)','k:',1:min_ploty,y(1,1:min_ploty),'k-');
hold on
figure(h12);plot(1:min_ploty,y_Jac(2,1:min_ploty)','r:',1:min_ploty,y(2,1:min_ploty),'r-
');
figure(h12);plot(1:min_ploty,y_Jac(3,1:min_ploty)','b:',1:min_ploty,y(3,1:min_ploty),'b
-');
figure(h12);plot(1:min_ploty,y_Jac(4,1:min_ploty)','g:',1:min_ploty,y(4,1:min_ploty),'g
-');
figure(h12);plot(1:min_ploty,y_Jac(5,1:min_ploty)','m:',1:min_ploty,y(5,1:min_ploty),'
m-');
```

```
legend('teta j','teta net','q j','q net','gama j','gama net','deltae j',...
    'deltae net','deltaf j','deltaf net')
xlabel('___ net outputs  --- Jacobian outputs ');
hold off
```

- Rotina função objetivo para a otimização

```
function [funcao,gradiente] =
funObjetivo_v14(u,u_1,Jdu,y_in,net,normaliz,r,R,Ru,Nu,N2)

%   CALCULO DA FUNCAO DESEMPENHO E SEU GRADIENTE ANALITICO -
27/08/2002
%   ENTRADAS DEVEM SER NORMALIZADAS ----27/08/2002
%   PROPAGACAO DA REDE E DO JACOBIANO DO INSTANTE "i" AT´E O
INSTANTE "N2"  N1=1 - 09/10/2002
%   ENTRADA DE CONTROLE  NET.INPUTS{1}
%   ENTRADA SAIDAS ATRASADAS NET.INPUTS{2}
%   DIMENSAO DA ENTRADA DE CONTROLE COM ATRASOS
NET.INPUTS{1}.SIZE
%   DIMENSAO DA ENTRADA SAIDAS ATRASADAS NET.INPUTS{2}.SIZE
%   r - referencia eh fixa no tempo para este caso senao seria r(i) do modelo de
referencia
%   R - MATRIZ DE PONDERACAO DOS ERROS - DIAGONAL
%   PROPAGACAO DA REDE DE "i" ATEH "N2", ITERAR AS SAIDAS DA REDE
COM CONTROLE CTE.

%   ITERAR SIGNIFICA EMPILHAR AS SAIDAS A CADA CHAMADA DE
"SIM(NET)"
%   NORMALIZACAO DA REFERENCIA r - 08/09/2002
%   SENSITIVIDADE EM RELACAO AO CONTROLE PRINCIPAL, SEM OS
ATRASADOS - 10/09/2002

%   ------------CALCULO DA DIAGONAL DA MATRIZ DE JACOBIANOS------------

   % entrada de controle no horizonte Nu a priori
%   PREENCHIMENTO DOS VETORES COM ZEROS NAS INICIALIZACOES

%   CORRECAO DA V_05, FORMACAO DA MATRIZ J, ESTAVA ERRADO
J23=J12! 30/10/2002
%   FMINUNC
%   VERSAO ATUAL 13/11/2002
%   DETERMINAÇAO CORRETA DO JACOBIANO GERAL P/ N ATRASOS DE
SAIDA NA ENTRADA DA REDE 15/02/2203
%-------------------------------------------------------------------------------------------
%   VERSAO FINALISSIMA CORRETA TESTADA COM GRADIENTE
NUMERICO E HORIZONTES VARIAVEIS  16/02/2003
%-------------------------------------------------------------------------------------------
minP = normaliz(1);
```

```
maxP = normaliz(2);
minT = normaliz(3);
maxT = normaliz(4);
Nmin = normaliz(5);
Nmax = normaliz(6);

Rg = R;

Ny = net.outputs{net.numLayers}.size;
leng_y = length(y_in);
ndelay_y = leng_y / Ny;
lu = length(u);
leng_u = lu / Nu;  % NAO HA ATRASOS NO CONTROLE DE ENTRADA UK-1
SOMENTE

%-------------------------------------------------------------
un        = (Nmax - Nmin) .* (u  - minP) ./ (maxP - minP) + Nmin;
u_1n      = (Nmax - Nmin) .* (u_1  - minP) ./ (maxP - minP) + Nmin;
Pnet{2,1}  = (Nmax - Nmin) .* (y_in  - minP) ./ (maxP - minP) + Nmin;
rn        = (Nmax - Nmin) .* (r  - minP) ./ (maxP - minP) + Nmin;
Paux{1}=Pnet{2,1};

%  PROPAGACAO DA REDE DE 1 A N2

for k = 1:N2
   % o controle vai de 1 ateh Nu e permanece constante ateh N2
   if k <= Nu
      Pnet{1,1}  = un((k-1)*leng_u+1:k*leng_u);
   end

   P = {[Pnet{1,1};Pnet{2,1}]};
   ysim = sim(net,P);   % as saidas sao geradas a cada instante "i" normalizadas
   %   desnormalizacao da saida
   yy = ysim{net.numLayers};
   y_ = (maxT - minT) .* (yy - Nmin) ./ ( Nmax - Nmin) + minT;

   if Ny < leng_y
      Paux{1}(Ny+1:leng_y) = Pnet{2}(1:leng_y-Ny);
   end

   Paux{1}(1:Ny) = yy; %ysim{net.numLayers,1};
   Pnet{2,1} = Paux{1};    %   no final tem-se aqui Y(i+1)
   %calculo dos erros
```

```
   y{k} = ysim;
   ee = (y_ - r');
   for i = 1:Ny
      ee(i) = ee(i) + sign(ee(i)) * eps;
      if ee(i) == 0
         ee(i) = eps;
      end
   end
   e{k} = ee;
end

for i = 1:N2
       JJnet(i) = jacobiano_v03(net, y{i});
end
for i = 1:N2
       Jyu{i,i} = JJnet{i}(:,1:leng_u);
end

%   CALCULO DA MATRIZ DE DERIVADAS
%   Jacobianos Jyu Jyy de 1 ateh N2
for i = 1:ndelay_y
   for j = i+1:N2
      Jyy{j,j-i} = JJnet{j}(:,leng_u+1+(i-1)*Ny:leng_u+Ny+(i-1)*Ny);
   end
end

for i = 1:Nu
   for k = i+1:N2
      soma = zeros(size(Jyu{1,1}));
      for j = k-1:-1:i
         if k-j <= ndelay_y
            soma = soma + Jyy{k,j} * Jyu{j,i};
         end
      end
      Jyu{k,i} = soma;
                  if i == Nu
                     Jyu{k,i} = soma + Jyu{k,k};
                  end
   end
end
Jyu = Jyu';

%   CALCULO DO GRADIENTE FINAL
```

```
aux=[];
for i = 1:Nu
   soma = zeros(leng_u,1);
   for j = i:N2
      soma  = soma + Jyu{i,j}' * Rg * e{j};
   end
   aux = [aux;soma];
end

grad_u = Ru * Jdu * (u - u_1);

gradiente = aux + grad_u;

for i = 1:lu
   if gradiente(i) == 0
      gradiente(i) = eps;
   end
end
%   CALCULO DA FUNCAO

soma = 0;
for j = 1:N2
   soma  = soma + e{j}' * R * e{j};
end
funcao = 0.5 * soma;

funcao_u = 0.5 * (u - u_1)' * Ru * (u - u_1);
funcao = funcao + funcao_u;
```

- Rotina de otimização, simulação e resultados

```
function [y,ynet,options] = otimizacao_v09(net,delayy,normaliz,dt,TFINAL)
%   PLOTAR CONTROLE EM STEPS   14/02/2003
%   VERSAO FINALISSIMA COM V14  FUNCIONANDO CORRETAMENTE
16/02/2002
%-----------------------------------------------------------------
minP = normaliz(1);
maxP = normaliz(2);
minT = normaliz(3);
maxT = normaliz(4);
Nmin = normaliz(5);
Nmax = normaliz(6);

N2 = 1; Nu = 1;

Ny = net.outputs{net.numLayers}.size;
sizu = 2*Nu;
sizy = Ny * delayy
R = zeros(5,5);

%   PARAMETROS D SINTONIA-----------------------------
H0 = 1000; %altitude m
Zvert  = [H0];
Xalong = [0];
VT = 200; %m/s 0.6 mach 3000ft

r = deg2rad([0 0 3 0 0]);

R(1,1) = 10; R(3,3) = 10; R(2,2) = 10;
gRu = 0.5;

Ru  = gRu * eye(sizu);

umax = 0.5 * ones(sizu,1); umin = - umax;

b = 0.0 * ones(sizu-2,1);
%-----------------------------------------------------
for i = 1:sizu
   for j = 1:sizu
      if i == j
         Au(i,j) = 1;
      elseif i == j+1
```

```
            Au(i,j) = -1;
        else
            Au(i,j) = 0;
        end
    end
end
if b == 0
    Au = [];
end

%     JACOBIANO DO Ip RELATIVO AO DELTAU
for i = 1:sizu
    for j = 1:sizu
        if i == j
            Jdu(i,j) = 1;
        elseif i == j+Nu
            Jdu(i,j)  = -1;
        else
            Jdu(i,j)  = 0;
        end
    end
end


options = optimset('GradObj','off',
'LargeScale','off','DerivativeCheck','off','Diagnostics','on','Display','iter',...
                'TolX',1e-10,'TolFun',1e-
10,'MaxIter',25,'LevenbergMarquardt','off','DiffMinChange',1e-10,'HessUpdate','dfp')


switch optimget(options,'GradObj')
case 'off'
    gradstr = 'numérico ';
case 'on'
    gradstr = 'analítico';
end

i=0;j=0;
y_in=[];
ty = [];
y=[];yn=[];uu=[];

u0 = zeros(sizu,1);
```

```
u00 = u0;
X0 = [0 0 0 0 0]';
i_flag = 0;

[A,B,C,D,ABK] = pitch_dynamic
%vetor de saida y =  [teta q gama deltae deltaf]
i = 0;j = 0;
i_inic=5;

h = figure;
h2= figure;
h3=axes;
set(h3,'FontSize',14);

for tsim = 0:dt:TFINAL
   tsim;
   i = i + 1;
   j = j + 1;
   if j>1
      ty(j) = tsim;
      ui = u0(1:2);
      uu{j} = ui;
      [t,X] = ode45(@pitch_dynamic_ode,[ty(j-1) ty(j)],X0,[],ABK,B,ui);
      sX = size(X,1);
      y{j} = C * X(sX,:)' + D * ui;
      X0 = X(sX,:)';
      y_in = y{j} + 0.2.*y{j}*(2*rand-1); %RUIDO 1/2/2003
      y_in = [y{j} ; y_in]
      if length(y_in) > sizy
         y_in(sizy+1:length(y_in)) = [];
      end
      yin{j} = y_in;

      figure(h)
      movegui(h,'northwest');
      hp1=subplot(211);
      uuu = [uu{:}];
      uuu = rad2deg(uuu);
      [tg,ug1]=stairs(ty(2:j),uuu(1,:));[tg,ug2]=stairs(ty(2:j),uuu(2,:))
      plot(tg,ug1,tg,ug2,':');hl=legend('deltae','deltaf',2);hl.FontSize=14;
      xlabel('tempo em segundos','FontSize',14); ylabel('Ângulos em
graus','FontSize',14);
```

```
        title(['Nu= ' num2str(Nu) ' N2= ' num2str(N2) ' R1=' num2str(R(1,1)) ' R2= '
num2str(R(2,2)) ' R3=' num2str(R(3,3))...
        ' Ru = ' num2str(gRu) ' grad ',gradstr],'FontSize',14)
        set(hp1,'FontSize',14);
        hp2=subplot(212);
        yy = rad2deg([y{:}]);
        plot(ty(2:j),yy(1,:),'-',ty(2:j),yy(3,:),':',ty(2:j),yy(2,:),'.-
'),hl=legend('teta','gama','q',2);hl.FontSize=14;
        xlabel('tempo em segundos','FontSize',14); ylabel('Ângulos em
graus','FontSize',14);
        set(hp2,'FontSize',14);
        figure(h2)
        movegui(h2,'southeast');
        dx = dt * VT * cos(y{j}(3));
        dy = dt * VT * sin(y{j}(3));
        Xalong = [Xalong  Xalong(j-1) + dx];
        Zvert  = [Zvert   Zvert(j-1)  + dy];
        [xac,zac] = AC_fig(y{j}(1),Xalong(j-1),Zvert(j-1),dx)

plot(Xalong,Zvert,'.',xac,zac),hl=legend('trajetória','inclinação',2);set(hl,'FontSize',14);
        axis([0 Xalong(j) 0 2*H0])
        hold on
        xlabel('deslocamento horizontal (m)','FontSize',14); ylabel('altitude
(m)','FontSize',14);
        pause(0.05)
        if i >= i_inic  %& i_flag == 1 % zero desativa otimizacao
           u_1 = u0;
           un_ =
fminunc(@funObjetivo_v14,u0,options,u0,Jdu,yin{j},net,normaliz,r,R,Ru,Nu,N2)

%X=FMINCON(FUN,X0,A,B,Aeq,Beq,LB,UB,NONLCON,OPTIONS,P1,P2,...)
           %un_ =
fmincon(@funObjetivo_v11,u0,Au,b,[],[],umin,umax,[],options,u0,y_in,net,normaliz,r,
R,Ru,Nu,N2);
           %   renormalizacao para tranformar saidas em entradas
           if all(1./un_)
              u0  = un_
           else
              u0 = u_1
           end
           i = 0;
           i_inic = 1;
        end
```

```
        end
    end
```

- **Rotina da dinâmica longitudinal**

```
function [A,B,C,D,ABK] = pitch_dynamic

%----------------------MODELO LINEAR LONGITUDINAL-----------------------------
-
%estado x=[teta q alfa deltae deltaf]'
%teta   - pitch
%q      - pitch rate
%alfa   - attack angle
%deltae - elevator deflection
%deltaf - flaperon deflection


%demanda de referencia ref =  [ deltaec deltafc]'
%deltaec - elevator deflection command
%deltaef - flaperon deflection command
%vetor de controle - u = [deltae deltaf]
%   SEM NORMALIZACAO  TESTE EM 17/10/2002

A=[ 0  1        0        0       0
    0 -0.8693  43.223  -17.251  -1.5766
    0  0.9933  -1.3411  -0.1689 -0.2518
    0  0        0      -20       0
    0  0        0        0      -20];

B=[0 0;0 0;0 0;20 0;0 20];

%vetor de saida y =  [teta q gama deltae deltaf]

C= [1   0      0     0    0
    0   1      0     0    0
    1   0     -1     0    0
    0   0      0     1    0
    0   0      0     0    1];

D=zeros(5,2);
%
%polos de CL desejados
%eig(A);

p=[-5.6+4.2*j, -5.6-4.2*j, -1, -19, -19.5];
```

```
%   controle por realimentacao trivial de estado "full" sem atribuicao de autovetores.
K = place(A,B,p);
ABK = A-B*K;
```

- Rotina para o integrador ODE

```
function X = pitch_dynamic_ode(T,X0,ABK,B,U)
X = ABK * X0 + B * U;
```