MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-10475-TDI/945

ESTRUTURAS DE INTEGRAÇÃO NEURAL *FEEDFORWARD* TESTADAS EM PROBLEMAS DE CONTROLE PREDITIVO

Paulo Marcelo Tasinaffo

Tese de Doutorado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Mecânica Espacial e Controle, orientada pelos Drs. Atair Rios Neto e Valdemir Carrara, aprovada em 18 de dezembro de 2003.

INPE
São José dos Campos
2004

629.7.062.2 : 681.3.019

TASINAFFO, P. M.
Estrutura de integração neural *feedforward.*testadas em problemas de controle preditivo / P. M. Tasinaffo. – São José dos Campos: INPE, 2003.
230p.- (INPE-10475-TDI/945).

1.Redes Neurais. 2.Sistemas não-lineares. 3.Sistemas dinâmicos. 4.Integração numérica. 5. Derivadas médias. 6. Derivadas instantâneas. 7.Cálculo diferencial. 8.Filtros de Kalman. 9.Sistemas de controle preditivo. 10.Controle de atitude de satélites. 11.Transferência de órbitas. I.Título.

Aprovado pela Banca Examinadora em cumprimento a requisito exigido para a obtenção do Título de **Doutor** em **Engenharia e Tecnologia Espaciais/Mecânica Espacial e Controle.**

Dr. Ijar Milagre da Fonseca
Presidente/INPE-SJCampos/SP

Dr. Atair Rios Neto
Orientador/Embraer, SJCampos/SP

Dr. Valdemir Carrara
Co-Orientador/INPE-SJCampos/SP

Dr. Hélio Koiti Kuga
Membro da Banca/INPE-SJCampos/SP

Dr. Cairo Lúcio Nascimento Junior
Membro da Banca
Convidado CTA-SJCampos/SP

Dr. Agenor de Toledo Fleury
Membro da Banca
Convidado POLI/USP-São Paulo/SP

Candidato (a): Paulo Marcelo Tasinaffo

São José dos Campos, 18 de dezembro de 2003.

AGRADECIMENTOS

Agradeço ao meu orientador Professor Atair Rios Neto pela sua notável colaboração, não somente devido às contribuições técnicas de elevado nível, mas também pelo apoio em todos os momentos difíceis. Certamente sem sua inestimável orientação a realização deste trabalho teria se tornado muito mais difícil. Ao meu co-orientador Professor Valdemir Carrara por sempre estar disposto a esclarecer minhas dúvidas e dificuldades.

A meus pais pela ajuda e apoio constantes que foram de importância significativa ao longo deste trabalho. Por todo auxílio e compreensão deixo-lhes a minha sincera gratidão.

Gostaria de destacar a valiosa ajuda e o afetuoso apoio de minha futura esposa, Gisele, que sempre soube ponderar minhas aflições, trazendo-me a luz da tranqüilidade.

A meus irmãos e à família Coicev agradeço por tudo. A todos os amigos e aos membros da igreja Batista da Graça que de certa forma colaboraram com o conforto espiritual para a realização deste trabalho. E a Deus, pois sem Ele nada seria possível.

RESUMO

Este trabalho apresenta, desenvolve e testa metodologia de modelagem de sistemas dinâmicos, aplicada em esquema de controle preditivo. Redes neurais feedforward são usadas na estrutura de integradores numéricos de equações diferenciais ordinárias, modelando a função de derivadas de sistemas dinâmicos autônomos. O integrador neural resultante é um modelo discretizado do sistema dinâmico que é usado como modelo interno em esquema de controle preditivo. O trabalho explora a abordagem com derivadas instantâneas, já existente na literatura, e propõe nova abordagem em que derivadas médias são usadas. O método com derivadas médias permite usar um integrador de Euler com passo fixo e ainda assim garantir a precisão desejada na modelagem. A estrutura simples de um método de primeira ordem facilita e simplifica a obtenção das derivadas parciais dos estados futuros em relação aos controles passados, necessários para a construção do Jacobiano presente na estrutura de controle preditivo. Estas duas metodologias de integração neural são aplicadas e testadas numa estrutura de controle preditivo, para a estimação dos controles que rastreiam trajetórias de referências pré-estabelecidas em um problema de transferência de órbita entre a Terra e Marte. O método das derivadas médias é também aplicado e testado no controle da estabilização da atitude de um satélite artificial funcionando como um corpo rígido.

STRUTURES OF FEEDFORWARD NEURAL INTEGRATION TESTED IN PROBLEMS OF PREDICTIVE CONTROL

ABSTRACT

This work presents, develops and tests a methodology for the modeling of dynamical systems applied to a scheme of predictive control. Feedforward neural networks are used in the structure of ordinary differential equations (ODE) numerical integrators modeling the derivative function of autonomous dynamic systems. The neural integrator is a discrete model of the dynamic system used as an internal model in the predictive control scheme. The work explores the approach already existing in the literature, which uses the instantaneous derivatives, and proposes a new approach where average derivatives are used. The approach with the average derivatives allows the use an ODE first order Euler numerical integrator with fixed step size, without affecting the specified modeling accuracy. The structure of a first order integrator simplifies and facilitates to get the partial derivatives of futures states with respect to past controls to construct the Jacobian, which is necessary in the predictive control structure implementation. These two methodologies of neural integration are applied and tested in a structure of predictive control for the estimation of the controls that track previously defined reference trajectories in an Earth to Mars orbit transfer problem. The method with the average derivatives is also applied and tested in the attitude stabilization control of a rigid body satellite.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Latinos

$\bar{e}$	ruído de média nula ou a incerteza do sistema
e_m	erro absoluto médio de saída da rede neural já treinada com a função de derivadas médias $\tan_{\Delta t}{}^{k}\alpha_{j}^{i}$ dentro de um domínio de interesse
$\hat{f}_w(x(t),\bar{w}(i))$	matriz das primeiras derivadas com respeito a **w** ou o *Jacobiano* da função de estimação com relação aos parâmetros
$f(x,u)$	função de derivadas
$\hat{f}({}^{k}y^{i}, {}^{k}u, \hat{w})$	estimativa de $\tan_{\Delta t}{}^{k}\alpha^{i}$ obtida pela rede neural
$H^{\#}$	matriz pseudo-inversa de **H**
$H(i)$	Jacobiano completo de todas as saídas y_v da função de estimação não-linear da i-ésima iteração
$H^{u}(t,i)$	Matriz de derivadas parciais ou Jacobiano
$\vec{h}_o$	vetor de momento angular
$I_{n\,x\,n}$	denota a matriz identidade de dimensão **n**
I_u	matriz identidade
I_x , I_y e I_z	Momentos de Inércia
$J(t)$	função de custo ou índice de desempenho
$k_{n\,x\,m}$	ganho de Kalman
m	número total de observações
m	dimensão do número de entradas da função vetorial de estimação $\hat{y}_v = \hat{f}(x_v, \hat{w})$, ou seja, a dimensão do vetor x_v
m	massa do foguete
M_x , M_y e M_z	Torques externos ao sistema

$\mathbf{n}$	número total de parâmetros a serem estimados
$\mathbf{n}$	horizonte em que as ações de controle e trajetórias de referência são consideradas
$\mathbf{n}$	dimensão do número de saídas da função vetorial de estimação $\hat{\mathbf{y}}_{\mathbf{v}} = \hat{\mathbf{f}}(\mathbf{x}_{\mathbf{v}}, \hat{\mathbf{w}})$, ou seja, a dimensão do vetor $\mathbf{y}_{\mathbf{v}}$
$\mathbf{n_o}$	número de entradas atrasadas do integrador numérico ou a ordem de aproximação
$\mathbf{n_{y'}}$	número total de variáveis de estado e que não pode ser confundido com $\mathbf{n_y}$ que é o número de entradas atrasadas da rede neural
$\mathbf{n_y}$	número de entradas atrasadas da rede neural
$\mathbf{p}$	número total de padrões de treinamento
$\mathbf{p}^*$	fator de normalização
$\mathbf{P(t)}$	matriz de covariâncias ($\mathbf{n}$ x $\mathbf{m}$) que contêm o processamento das primeiras $\mathbf{t}$ observações
$\bar{p}_{ii}$	valor genérico da diagonal principal da matriz de covariâncias a priori associada aos pesos de entrada do neurônio Perceptron
$\mathbf{r}$	raio da órbita
$\mathbf{R_u(t)}, \mathbf{R_y(t)}$ e $\hat{\mathbf{R}}\mathbf{(t,I)}$	matrizes de covariância de $\mathbf{V_u(t)}$, $\mathbf{V_y(t)}$ e $\mathbf{(\hat{U}(t,I) - U(t))}$
$\mathbf{r_y(t_j)}$ e $\mathbf{r_u(t_j)}$	matrizes de pesos definidas positivas
$\mathbf{tan}_{\Delta \mathbf{t}}{}^{\mathbf{k}} \alpha_{\mathbf{j}}^{\mathbf{i}}$	função de derivadas médias
$\frac{\partial \, \mathbf{tan}_{\Delta \mathbf{t}}{}^{\mathbf{k+2}} \alpha^{\mathbf{i}}}{\partial \, {}^{\mathbf{k}}\mathbf{u}}$	matriz de derivadas parciais das funções de derivadas médias
$\mathbf{T_x}$, $\mathbf{T_y}$ e $\mathbf{T_z}$	torques externos ao sistema
$\mathbf{u} \in \mathbf{R}^{\mathbf{n}}$	vetor das variáveis de controle
$\bar{\mathbf{u}}\mathbf{(t_j, i)}$	vetor das variáveis de controle no instante $\mathbf{t_j}$ na i-ésima iteração

$\overline{\mathbf{U}}(\mathbf{t,i})$	vetor de informações a priori das variáveis de controle
$\hat{\mathbf{U}}(\mathbf{t_{-1}})$	estimativa anterior das variáveis de controle dentro de uma precisão aceitável
$\mathbf{v}$	velocidade transversal
$\mathbf{v(t)}$	t-ésimo elemento do vetor de ruídos $\mathbf{v} = [\mathbf{v_i}]_{\mathbf{m\ x\ 1}}$
$\mathbf{v_y(t_j)}$ e $\mathbf{v_u(t_j)}$	são as componentes não-correlacionadas de ruído para diferentes valores de $\mathbf{t_j}$
$\mathbf{V_y(t)}$	vetor de ruídos de média nula da matriz de covariância $\mathbf{r_y(t)}$
$\mathbf{x} \in \mathbf{R^n}$	vetor das variáveis de estados
$\mathbf{x_v} = [\mathbf{x_1\ x_2 \ldots x_m}]^{\mathbf{T}}$	vetor das variáveis de entrada da função de estimação
$\mathbf{Z_i}$	variável aleatória que representa as ocorrências observadas $\mathbf{z_i}$
$\mathbf{z_i}$	ocorrências observadas de uma variável aleatória
$\mathbf{z(t)}$	t-ésimo elemento do vetor de observações $\mathbf{z} = [\mathbf{z_i}]_{\mathbf{m\ x1}}$
$\mathbf{y_r(t_j)}$	trajetória de referência
$\mathbf{y(i)}$	vetor dos padrões de treinamento
$\mathbf{y(i)}$	vetor das saídas da rede na i-ésima iteração devido a informação a priori de $\overline{\mathbf{w}}\mathbf{(i)}$
$\hat{\mathbf{y}}(\mathbf{t_j})$	saída da rede *feeddforward* treinada para aproximar o modelo do sistema dinâmico no instante $\mathbf{t_j}$
$\mathbf{y_v} = [\mathbf{y_1\ y_2 \ldots y_n}]^{\mathbf{T}}$	vetor das variáveis de saída da função de estimação
$\mathbf{y_v} = \hat{\mathbf{f}}[\mathbf{x_v}, \hat{\mathbf{w}}]$	função vetorial de estimação
$\overline{\mathbf{y}}\mathbf{(i)} = \hat{\mathbf{f}}(\mathbf{x}, \overline{\mathbf{w}}\mathbf{(i)})$	valor da função $\hat{\mathbf{f}}\mathbf{(.)}$ sobre o ponto no qual se realizou a linearização
$\mathbf{y_{ij}^n}$	padrão de treinamento $\mathbf{y_{ij}}$ normalizado
$\overline{\mathbf{y}}_{\mathbf{ij}}^{\mathbf{n}}$	saída da rede normalizada
$\mathbf{y_r(t_j)}$	trajetória de referência no instante $\mathbf{t_j}$

$\mathbf{y_n(t_j)}$	resposta do sistema dinâmico discretizado pelo conjunto integrador e rede
$\frac{\partial^{\mathbf{k+n}}\mathbf{y^i}}{\partial^{\mathbf{k}}\mathbf{u}}$	matriz de derivadas parciais dos estados adiantados em relação aos controles presentes
$\mathbf{W}$	variável aleatória
$\mathbf{w}$	velocidade radial
$\mathbf{w}$	valor verdadeiro da estimativa que não se conhece e deseja-se encontrar
$\mathbf{w}$	realização ou ocorrência da variável aleatória
$\overline{\mathbf{w}}$	estimativa inicial do valor médio de **w**
$\hat{\mathbf{w}}\mathbf{(t)}$	vetor (**n x 1**) dos parâmetros que contêm o processamento das primeiras **t** observações
$\overline{\mathbf{w}}\mathbf{(i)}$	estimativa a priori tomada da iteração anterior e iniciada em $\overline{\mathbf{w}}\mathbf{(1)} = \overline{\mathbf{w}}$
$\vec{\mathbf{w}} = \begin{bmatrix}\mathbf{w_x} & \mathbf{w_y} & \mathbf{w_z}\end{bmatrix}^{\mathbf{T}}$	vetor de velocidades angulares em relação aos eixos fixos ao corpo

Gregos

$\mathbf{0} < \alpha\mathbf{(i)} \leq \mathbf{1}$	parâmetro empírico a ser ajustado para garantir a hipótese de linearização
$\mathbf{0} < \beta << \mathbf{1}$	porcentagem associada ao valor do peso $\mathbf{w_i}$
β	coeficiente de ajuste das exponenciais amortecidas para as trajetórias de referência
θ	ângulo guiado do empuxo do foguete
μ	constante gravitacional
ω	eventos elementares
φ , θ e ϕ	ângulos de Euler
$\varphi_{\mathbf{ref}}$, $\theta_{\mathbf{ref}}$ e $\phi_{\mathbf{ref}}$	referência para os ângulos de Euler

CAPÍTULO 1

INTRODUÇÃO

1.1 Objetivos e Metodologia.

Redes neurais artificiais abrem novas possibilidades de soluções de controle de sistemas. Uma rede *feedforward* do tipo multi-camadas de *perceptrons* pode aproximar arbitrariamente bem uma função contínua. Hornik et al (1989) e Hunt et al (1992) mostram que uma função contínua pode ser arbitrariamente bem aproximada, com qualquer precisão desejada não-nula, por uma rede *feedforward* com somente uma simples camada interna, onde cada unidade neural na camada interna tem uma não-linearidade contínua sigmoidal. Este é o teorema fundamental das redes neurais e que justifica e viabiliza, teoricamente, a aplicação direta delas em problemas modernos e complexos de engenharia, embora na prática seja muito difícil definir e decidir pelos parâmetros básicos da arquitetura neural a ser treinada. Parâmetros como a inicialização dos pesos da rede, número de camadas internas e número de neurônios de cada camada interna só podem ser especificados empiricamente, dificultando ou não, diretamente, na convergência do aprendizado da rede até uma precisão desejada. A utilização prática de redes neurais artificiais justifica-se também, modernamente, pelos avanços extraordinários alcançados pela engenharia de *hardware*.

Neste trabalho, esta capacidade de representação de redes neurais *feedforward* é aproveitada, juntamente com o teorema do valor médio, para se propor, desenvolver e testar uma nova forma de representação de sistemas dinâmicos. Adota-se uma dupla modificação, com relação à representação tradicional de sistemas dinâmicos através de redes neurais. Pela metodologia tradicional (Hunt et al, 1992) é comum ela ser realizada pelo critério de entradas atrasadas ou metodologia *Nonlinear Auto Regressive Moving Average with Exogeneous Inputs* (NARMAX). Por outro lado, aqui, como primeira modificação, utiliza-se a metodologia de se inserir a rede *feedforward* dentro de uma estrutura de integrador numérico de equações diferenciais ordinárias, onde a rede neural necessita somente aprender a função de derivadas do sistema dinâmico autônomo (Wang e Lin ,1998b; Rios Neto, 2001). Em seguida o conjunto rede neural e integrador

numérico é inserido dentro de uma estrutura de controle preditivo como já era feito anteriormente pelo método NARMAX. Como segunda modificação, elabora-se um novo método de representação de sistemas dinâmicos utilizando a rede neural apenas para representar a função de derivadas *médias* do sistema dinâmico autônomo original e não mais a função de derivadas *instantâneas* como proposto originalmente por Wang e Lin (1998b) e Rios Neto (2001). As principais diferenças entre esses dois métodos são: a primeira é que no método das derivadas *médias* o passo de integração é fixo, enquanto no outro, permite-se variar o passo de integração para auxiliar o ajuste do erro da propagação das condições iniciais do sistema dinâmico; e a segunda diferença básica é que no método das derivadas médias consegue-se ter uma elevada precisão sendo ele mesmo um método de integração de primeira ordem, enquanto no método das derivadas instantâneas só se conseguirá ter uma elevada precisão se a rede neural for utilizada dentro de um integrador de ordem mais elevada. Esta última diferença tem uma importância fundamental: o método das derivadas médias possuirá expressões analíticas para a retro-progagação bem mais simples do que as exigidas pelo método das derivadas instantâneas, tanto para o caso do treinamento neural, propriamente dito, como também para a aplicação da estrutura de controle preditivo.

Explorar-se-á também a facilidade de tratar os problemas de treinamento da rede neural e de controle preditivo através de uma abordagem estocástica unificada, proposta por Rios Neto (2000). Este tipo de metodologia tem a vantagem de unificar os problemas de treinamento da rede e de controle do sistema dinâmico. Pode-se utilizar a filtragem de Kalman em duas situações distintas, porém complementares entre si: consegue-se treinar ou estimar uma rede neural *feedforward* para aprender a dinâmica de um sistema físico invariante no tempo com qualquer grau de precisão desejado, que em seguida, poderá ser naturalmente utilizada por uma estrutura de controle preditivo, que nada mais é que outro problema de estimação, cuja rede, já treinada, é o vínculo de um funcional quadrático que rastreará uma trajetória de referência segundo uma política de controle suave com o intuito fundamental de se projetar e elaborar um controlador neural. A contribuição do trabalho se completa com a aplicação e teste desta metodologia em problemas de transferência de órbitas e controle de atitude de satélites.

1.2 Revisão Bibliográfica.

Após um longo período de experimentação e pesquisa, controladores baseados em redes neurais estão finalmente emergindo e tomando espaço no mercado. Os benefícios de tais controladores segundo Narendra (1996) estão sendo agora usufruídos em uma larga variedade de campos. Muitos sistemas do mundo real (Chen e Billings, 1992) exibem características não lineares complexas e não podem ser tratados satisfatoriamente bem utilizando-se a teoria de sistemas lineares.

Redes neurais artificiais possuem a capacidade de aprender por exemplos e fazer interpolações do que aprenderam (Braga et al, 2000). Este aprendizado, em geral, é obtido por algoritmos de otimização numérica não-linear. Um conceito fundamental no treinamento das redes *feedforward* é o conceito da retro-propagação que aparece pela primeira vez num trabalho de Werbos (1974). Hagan e Menhaj (1994) mostram que os métodos para acelerar a convergência dos algoritmos de treinamento das redes *feedforward* podem ser divididos grosseiramente em duas categorias:

a) a primeira categoria envolve técnicas heurísticas. Tais idéias incluem variação na taxa de aprendizado e uso do momentum;

b) a segunda categoria enfoca as técnicas padronizadas de otimização numérica. A mais popular abordagem desta categoria tem utilizado os métodos do gradiente conjugado ou Quase-Newton. Além destas, citam-se a dos mínimos quadrados não-linear.

Algoritmos de segunda ordem para o treinamento das redes *Multi Layer Perceptron* (MLP) baseado no método de Newton revisado são propostos por Wang e Lin (1998a). Há vários problemas em se utilizar o método de Newton para minimizar a função erro de saída de uma rede MLP. Uma das principais dificuldades está em computar a inversa da matriz Hessiana. Wang e Lin (1998a) definem a matriz Hessiana em bloco para aproximar e simplificar a matriz Hessiana global. Shah et al (1992) propõem um algoritmo de filtragem para acelerar o aprendizado de uma rede *feedforward*.

Algoritmos genéticos segundo Blanco et al (2000) são utilizados para obter não somente uma topologia ótima para uma rede neural recorrente, mas também um número mínimo de conexões neurais necessárias para cada neurônio.

A versão da filtragem de Kalman com processamento paralelo e recursivo tem sido bastante explorada por Rios Neto (1997) no treinamento de redes neurais. Este tipo de metodologia, utilizada no treinamento das redes *feedforward*, tem as vantagens de fornecer uma taxa de convergência mais alta e insensibilidade a variações no número total de padrões de treinamento a ser considerado. Processamento paralelo local da filtragem de Kalman no nível neural, com procedimentos adaptativos, também tem sido explorado por Rios Neto (1997).

A versão da filtragem de Kalman com processamento paralelo e recursivo foi testada por Silva e Rios Neto (1999) com relação aos problemas XOR e diagnóstico do câncer de mama com resultados bastante satisfatórios.

Algoritmos de aprendizado supervisionado para treinar uma rede *Radial Basis Functions* (RBF) utilizando um par de filtros de Kalman em paralelo para atualizar seqüencialmente os pesos de saída e os centros deste tipo particular de rede são propostos por Ciocoiu (2002). Rivals e Personnaz (1998) demonstram que o filtro de Kalman estendido (EKF) converge mais rápido e para um mínimo menor do que aqueles obtidos pelos métodos recursivos e não-recursivos de predição de erros de primeira-ordem que utilizam o algoritmo *backpropagation* para computar o gradiente da função custo sobre redes com arquitetura *feedforward*. O filtro estendido de Kalman para treinar uma rede do tipo RBF para a aplicação nas áreas de processamento de sinais e reconhecimento de padrões é explorado por Simon (2002). Ruck et al (1992) comparam o filtro estendido de Kalman com a regra *backpropagation* com relação ao desempenho de treinamento da rede neural, utilizando-se de um exemplo prático referente às imagens com efeito doppler obtidas de um radar a laser. O algoritmo *backpropagation* é uma forma degenerativa do filtro estendido de Kalman, e, de forma bastante simplificada, Chandran (1994) consegue demonstrar esta importante propriedade no treinamento das redes *feedforward*.

Na literatura, apenas em nível ilustrativo da abrangência alcançada pelas aplicações possíveis das redes neurais, são encontrados inúmeros exemplos interessantes: um sistema conexionista proposto por Basak et al (1995), um modelo híbrido neural de Chiang (1998) as redes *Adaptative Resonance Theory* (ART) utilizada por Kim et al (1996) e o trabalho de Shustorovich et al (1996) estão todos relacionados diretamente ao projeto de reconhecimento de caracteres lingüísticos manuscritos e inclusive os caracteres orientais, trabalhos estes muito úteis às empresas de correio.

As aplicações são as mais diversificadas possíveis, por exemplo, Simon et al (1995) aplicam redes neurais para relacionar um subconjunto de satélites ótimos para serem utilizados em navegação. Ferreira et al (2002) utilizam a rede RBF para modelar a temperatura do ar interior de uma estufa de plantas e Tan e Saif (2000) aplicam redes recorrentes para modelar a dinâmica não-linear de motores automotivos. Bassoe (1995) através das redes neurais associativas consegue propor máquinas de diagnósticos clínicos automatizadas.

Por outro lado, estruturas de controle também podem ser utilizadas juntamente com redes neurais. Sendo assim, existem vários tipos de estruturas de controle neural encontrados na literatura internacional, e segundo Hunt et al (1992) algumas delas são: controle supervisionado, controle inverso direto, estrutura de controle *Internal Model Control* (IMC) e controle preditivo. As principais características das redes neurais na implementação em controle segundo Hunt e Sbarbaro (1991) e Hunt et al (1992) são:

1) a habilidade de representar relações arbitrárias não-lineares;

2) adaptação e aprendizagem de sistemas incertos através de atualização dos parâmetros ou pesos da rede em tempo *off-line* ou *on-line*;

3) a transformação da informação do sinal, tanto em nível qualitativo quanto em nível quantitativo, permitindo a fusão dos dados;

4) a habilidade de processamento paralelo na arquitetura das redes neurais, que permite um processamento mais rápido para muitos sistemas dinâmicos;

5) um elevado grau de robustez nas arquiteturas das redes permitindo a tolerância de erros e evitando a degradação do sinal;

6) a capacidade de se utilizar estruturas de aprendizado para representar a dinâmica direta e inversa de sistemas dinâmicos não-lineares;

7) a aplicação de forma direta e natural das redes neurais em estruturas de controle.

Balakrishnan e Weil (1996) realizam uma revisão na literatura sobre controle neural. Listam mais de 75 citações da área em aplicações experimentais e teóricas em controle neural. As redes neurais com arquitetura RBF são exploradas teoricamente nas aplicações referentes aos sistemas de controle adaptativo para sistemas não-lineares por Hunt e Sbarbaro (1991). A utilização dos modelos direto e inverso da planta é bem explorada por estes autores na estrutura de controle IMC.

Ender e Maciel Filho (2000) apresentam uma nova estratégia de controle multivariável. Nesta estratégia estes autores utilizam o método direto para obter o controlador e o método indireto para gerar o modelo neural do sistema. Tanto o controlador como o modelo neural do processo são otimizados em tempo real.

Jarmulak et al (1997) apresentam vários esquemas de controle utilizando redes neurais que podem ser aplicados no controle de sistemas não-lineares mal definidos como, por exemplo, ocorre freqüentemente na agricultura. Estes autores combinam as técnicas dos algoritmos genéticos e da regra *backpropagation* para conduzir numerosos experimentos com controles neurais.

Kasparian e Batur (1998) lidam com modelo de referência baseado em estrutura neural que pode ser utilizado em controle adaptativo de processos lineares e não-lineares. Estes autores utilizam um algoritmo rápido, que é baseado na técnica de minimização dos mínimos quadrados de Davidon, para treinar o controlador neural.

A identificação de sistemas dinâmicos através de uma rede neural RBF é explorada por Liu et al (1996). A identificação de sistemas dinâmicos é realizada em dois passos principais: o primeiro consiste em escolher um modelo de identificação apropriado e o

segundo em ajustar os parâmetros do modelo de acordo com alguma lei adaptativa. Uma vez que redes neurais possuem boa capacidade de aproximação e características adaptativas inerentes, elas provêm uma poderosa ferramenta para identificação de sistemas dinâmicos com não-linearidades desconhecidas.

Nikravesh et al (1997) discutem a análise da estabilidade da estrutura das redes *Dynamic Neural Network Control* (DNNC). Os resultados da análise da estabilidade podem ser utilizados para projetar um controlador confiável atuando, por exemplo, numa estrutura de controle IMC.

Métodos adaptativos para formar um controlador adaptativo neural são explorados por Mills et al (1994). A performance deste controlador é demonstrada por estes autores e avaliada através da simulação de dois processos realistas: o controle de nível de um tanque cônico e um controle multivariável de um evaporador industrial.

Economou, Morari e Palsson (1986) apresentam um operador formalizado que é utilizado para estender a estrutura de controle IMC de sistemas lineares para sistemas não-lineares. Uma extensiva revisão de várias aplicações utilizando redes neurais para controle de processos químicos em tempo real é abordada por Hussian (1999). Este autor divide a revisão em três categorias principais: controle preditivo, controle baseado no modelo inverso e métodos de controle adaptativos. Tais técnicas são bastante utilizadas em análises de dados de sensores, detecção de defeitos e identificação de processos não-lineares.

Uma comparação de técnicas de controle preditivo não-linear utilizando modelos de redes neurais é abordada por Botto e Costa (1998). Liu et al (1998) mostram que existem quatro métodos principais baseados no controle preditivo: *Model Algorithmic Control* (MAC) de 1987, *Dynamic Matrix Control* (DMC) de 1980, *Extended Prediction Self-Adaptative Control* (EPSAC) de 1985 e *Extended Horizon Adaptative Control* (EHAC) de 1984. Há ainda um quinto tipo denominado *Model Predictive Control* (MPC).

Soloway e Haley (1997 e 1998) apresentam uma eficiente implementação do controle preditivo generalizado neural (GPC). Estes autores utilizam uma rede neural *feedforward* como modelo da planta não-linear e usam o algoritmo de otimização Newton-Raphson com processamento em tempo real.

Rios Neto (2000) introduz a filtragem de Kalman para formular e resolver os problemas de otimização exigidos pelo controle preditivo. Silva e Rios Neto (2000) aplicam o esquema de controle preditivo neural no controle da atitude de satélites artificiais. Os algoritmos de filtragem de Kalman são utilizados não somente para treinar o modelo de rede neural *feedforward* associada, mas também para estimar as ações de controle preditivo. Desta forma, tanto o treinamento da rede neural como do controle podem ser abordados de uma forma unificada. Há ainda vários outros trabalhos relacionados à teoria e aplicação das estruturas de controle preditivo neural. Entre eles citam-se:

1) Tan e Van Cauvenberghe (1996) apresentam três diferentes métodos de otimização para o projeto de um controlador preditivo suave baseado em redes neurais recorrentes para processos não-lineares com grande tempo de atraso. Os métodos de otimização apresentados são do gradiente, Newton-Raphson e Levenberg-Marquardt;

2) Huang e Van Cauvenberghe (1998) propõem a utilização de rede neural em um controle preditivo múltiplo em *feedback* para processos industriais não-lineares dispostos em cascata;

3) Sorensen et al (1999) descrevem um método de controle para sistemas não-lineares baseado em uma estrutura de controle preditivo generalizada, com ênfase no algoritmo de Quasi-Newton, e sua performance é demonstrada sobre um servo-mecanismo pneumático;

4) Benne et al (2000) tratam da modelação não-linear da evaporação de múltiplos efeitos numa indústria de cana de açúcar com o objetivo de elaborar um controle robusto e para superar os limites dos sistemas de controle tradicionais usam esquema de controle preditivo.

Com relação aos trabalhos desenvolvidos no Instituto Nacional de Pesquisas Espaciais (INPE) e mais relacionados diretamente a este trabalho, Carrara (1997), Carrara e Rios Neto (1997) e Varotto (1997) exploram a identificação de sistemas dinâmicos através das redes *feedforward* aplicadas no aprendizado da dinâmica de atitude de um satélite com geometria variável. Carrara e Rios Neto (1998 e 1999), Carrara, Varoto e Rios Neto (1998) dão prosseguimento a esses estudos e apontam as principais dificuldades de se aplicar redes neurais ao controle da atitude de satélites artificiais. Entre elas citam-se: o número total de neurônios da camada interna da rede e a especificação da densidade do espaço das condições iniciais referentes aos padrões de treinamento da dinâmica do sistema. Silva e Rios Neto (2000 e 2001) desenvolvem, respectivamente, trabalhos referentes às aplicações de uma estrutura de controle preditivo no controle da atitude de satélites artificiais e na trajetória de vôos.

Apolloni et al (1997) aplicam uma rede neural recorrente como componente chave de um sistema de controle hábil para regular um satélite artificial cuja atitude deve ser mantida ao redor do ângulo zero com respeito a um sistema referencial inercial centrado na Terra.

Para finalizar, uma abordagem bastante importante surgiu recentemente e é a de utilizar uma rede *feedforward* dentro de uma estrutura de integração numérica, na tentativa de representar sistemas dinâmicos descritos por equações diferenciais ordinárias onde não se conhece a função de derivadas do modelo teórico do sistema original. Wang e Lin (1998b), aplicam este conceito ao integrador Runge-Kutta de quarta ordem na tentativa bem sucedida de representar sistemas dinâmicos ordinários e introduzem na literatura o termo rede neural de Runge-Kutta. Rios Neto (2001) utiliza este conceito dentro da teoria de controle neural.

He et al (2000) verificaram que as redes *feedforward* podem ser utilizadas para encontrar facilmente soluções completas e aproximadas para equações diferenciais parciais de primeira ordem mais complicadas. Esta metodologia poderá ser utilizada no projeto de uma classe de sistemas de controle não-linear, onde as soluções não-triviais das equações diferenciais parciais são difíceis de achar. Mai-Duy e Tran-Cong (2001)

utilizam redes com funções de base radial multi-quadradas para resolver equações diferenciais lineares.

1.3 Organização.

Para cumprir os objetivos propostos, esta tese está organizada como segue. No Capítulo 2, faz-se uma introdução teórica de processos estocásticos, dando-se ênfase ao filtro de Kalman aplicado ao problema de estimação de parâmetros em sistemas não-lineares. No Capítulo 3, é dada uma introdução sobre a teoria de redes neurais que utiliza o algoritmo de treinamento neural do filtro de Kalman com processamento paralelo e recursivo. No Capítulo 4 dá-se uma introdução geral sobre as principais estruturas de controle neural, dando-se ênfase à estrutura de controle preditivo. No Capítulo 5 são apresentadas as metodologias originais deste trabalho, quais sejam, a das derivadas instantâneas e a das derivadas médias para a elaboração de integradores neurais aplicados na representação da dinâmica de sistemas físicos e as suas respectivas utilizações em estruturas de controle preditivo. No Capítulo 6 os principais testes e resultados numéricos obtidos da aplicação destas duas metodologias nos problemas de transferência de órbitas e controle de atitude de satélites artificiais são apresentados. No Capítulo 7, conclusões e sugestões para trabalhos futuros são propostos. No Apêndice A é apresentada à equivalência matemática entre os métodos de treinamento neural do gradiente e da filtragem de Kalman. No Apêndice B é apresentado detalhadamente o fluxograma da filtragem de Kalman com processamento paralelo e recursivo para o treinamento neural. No Apêndice C é demonstrado matematicamente a retro-propagação em relação às saídas dos neurônios numa rede *feedforward*. No Apêndice D uma breve introdução à teoria dos integradores numéricos é dada e no Apêndice E é desenvolvido o cálculo das derivadas parciais para uma estrutura de controle preditivo trabalhando com as derivadas médias, sendo essa, outra parte original deste trabalho.

CAPÍTULO 2

ESTIMAÇÃO DE PARÂMETROS E FILTRO DE KALMAN

2.1 Introdução.

Pretende-se neste capítulo abordar, sucintamente, tópicos necessários para a formulação e desenvolvimento do filtro de Kalman. Esta ferramenta matemática (Kalman, 1960) possui a peculiaridade de apresentar ampla aplicação no desenvolvimento tecnológico de satélites, robôs e de toda tecnologia moderna em controle estocástico que vem se desenvolvendo nas últimas décadas.

Tendo isso em vista, neste capítulo pretende-se enunciar as equações de estimação de parâmetros referentes ao filtro de Kalman para o caso não-linear a partir do conceito fundamental de variáveis aleatórias. Ver-se-á que estas equações são oriundas da linearização das equações de estimação de parâmetros de um estimador Gauss-Markov, de variáveis aleatórias linearmente correlacionadas e com os erros das observações independentes dos parâmetros a serem estimados. Pretende-se também, no decorrer deste capítulo, introduzir o conceito de processamento *recursivo*.

O filtro de Kalman tem duas grandes finalidades no desenvolvimento deste trabalho: treinar uma rede neural *feedforward* do tipo não-linear (ver Capítulo 3) e resolver um problema de controle ótimo em malha fechada (ver Capítulos 4 e 5) em que a dinâmica é aproximada pela própria rede neural.

2.2 A Estimação Linear de Parâmetros.

Em um problema de estimação de parâmetros deseja-se encontrar a função de distribuição do vetor de variáveis aleatórias **W** que representa os parâmetros a serem estimados. Desta maneira, deve existir uma relação matemática entre as variáveis aleatórias $\mathbf{Z_i}$ que representam as ocorrências observadas $\mathbf{z_i}$ e o vetor aleatório **W** dos parâmetros, ou seja,

$$\mathbf{Z_i} = \mathbf{h_i(W, V_i)} \tag{1}$$

A variável aleatória $\mathbf{V_i}$ é devida ao erro que todo sistema físico a ser observado acarreta devido ao erro inerente aos instrumentos de medida (analógico ou digital) que são utilizados para observá-lo e medi-lo. Admitindo-se que os erros de observação são aditivos, as *ocorrências* da variável aleatória $\mathbf{Z_i}$ podem, então, ser representadas por:

$$\mathbf{z_i} = \mathbf{h_i(w)} + \mathbf{v_i} \qquad \mathbf{i=1, 2, ..., m} \tag{2}$$

Para cada ocorrência $\mathbf{z_i}$ existe um erro de medida $\mathbf{v_i}$ associado. É importante diferenciar as letras maiúsculas das minúsculas que representam, respectivamente, as variáveis aleatórias e as ocorrências das variáveis aleatórias. É costume na literatura básica representar as variáveis aleatórias por letras minúsculas contrariando a definição acima. Deste modo, essa convenção também será adotada neste capítulo, a menos que se mencione o contrário.

Partindo-se portanto, de um número considerável de observações das ocorrências $\mathbf{z_i}$ pode-se encontrar indiretamente uma estimativa de **w**, isto é, a função de distribuição de **w.** Os critérios utilizados para a estimação dos parâmetros **w** são dois:

1) critério da *não tendenciosidade*, ou seja, se $\hat{\mathbf{w}}$ for a estimativa de **w**, então a seguinte relação deve ser satisfeita

$$\boldsymbol{\varepsilon}\,\mathbf{[(w - \hat{w})] = 0} \tag{3.a}$$

2) critério da *mínima variância* ou de minimização do erro quadrático médio de $(\mathbf{w} - \hat{\mathbf{w}})$

$$\underset{\{\hat{\mathbf{w}}\}}{\mathbf{Min}}\ \boldsymbol{\varepsilon}\left[(\mathbf{w} - \hat{\mathbf{w}})^{\mathbf{T}} \cdot \mathbf{S} \cdot (\mathbf{w} - \hat{\mathbf{w}})\right] \tag{3.b}$$

onde, **S** é uma matriz simétrica definida positiva.

Teorema 01 (Teorema Fundamental da Estimação de Parâmetros): os critérios de não tendenciosidade e das mínimas variâncias são satisfeitos simultaneamente quando a estimativa de $\hat{\mathbf{w}}$ for dada por (e.g., Jazwinski, 1970):

$$\hat{\mathbf{w}} = \boldsymbol{\varepsilon}\left\{\mathbf{w/z}\right\} \tag{4}$$

O resultado do teorema 01 é bastante geral, pois além de independer do tipo de distribuição das variáveis aleatórias **w** e **z**, também independe de como estas variáveis estão relacionadas. Para estimar os parâmetros $\hat{\mathbf{w}}$ do vetor aleatório **w** é preciso então, determinar a esperança condicionada $\boldsymbol{\varepsilon}\{\mathbf{w/z}\}$. Para variáveis aleatórias Gaussianas basta determinar a função densidade condicionada $\mathbf{p_{w/z}(w/z)}$ para se obter os dois momentos $\boldsymbol{\varepsilon}\{\mathbf{w/z}\}$ e $\mathbf{cov(w/z)}$ que surgirão naturalmente. Aplicando a regra de Bayes (e.g., Papoulis, 1965) tem-se:

$$\mathbf{p_{w/z}(w/z)} = \frac{\mathbf{p_{z/w}(z/w)} \cdot \mathbf{p_w(w)}}{\mathbf{p_z(z)}} \tag{5}$$

As distribuições $\mathbf{p_{z/w}(z/w)}$, $\mathbf{p_w(w)}$ e $\mathbf{p_z(z)}$ devem ser determinadas para se conseguir estimar $\hat{\mathbf{w}}$. Para tanto é necessário que estas três distribuições sejam determinadas a partir de uma estimação inicial $\overline{\mathbf{w}}$ ou informação a priori da variável aleatória **w** como segue:

$$\overline{\mathbf{w}} = \mathbf{w} + \overline{\mathbf{e}} \tag{6}$$

onde,

$\bar{\mathbf{e}}$ ... ruído de média nula ou a incerteza do sistema;

$\overline{\mathbf{w}}$ ... estimativa inicial do valor médio de **w**;

w ... valor verdadeiro da variável que não se conhece e deseja-se estimar.

A principal observação, a ser lembrada em uma estimação de parâmetros, é que se deve sempre estimar a priori os parâmetros que deverão ser encontrados, ou seja, é sempre necessário conhecer alguma coisa daquilo que se deseja determinar. A distribuição de $\bar{\mathbf{e}}$ também deve ser conhecida, e, por se tratar de um erro, sua média é nula. Em outras palavras:

$$\boldsymbol{\varepsilon}\{\bar{\mathbf{e}}\} = \mathbf{0} \tag{7.a}$$

$$\boldsymbol{\varepsilon}\{\bar{\mathbf{e}} \cdot \bar{\mathbf{e}}^{\mathbf{T}}\} = \overline{\mathbf{P}} \tag{7.b}$$

A equação (7.b) serve também como estimativa a priori para a $\mathbf{cov(w)}$. Um caso particular e de bastante interesse é quando as observações são lineares em relação aos parâmetros, as variáveis aleatórias são Gaussianas e os erros das observações são independentes de **w**. Quando isto ocorre a estimação é dita linear, e como será visto mais adiante, isso conduzirá a um estimador de Gauss-Markov que pode ser colocado na forma recursiva do algoritmo de filtragem de Kalman. A seguir enunciar-se-á a estimação linear de parâmetros e, na seção subsequente, esta teoria será estendida para tratar iterativamente o caso não-linear, que é o objetivo principal deste capítulo. Neste trabalho, a estimação não-linear de parâmetros será utilizada no treinamento das redes neurais (ver Capítulo 3) e numa estrutura de controle preditivo para estimação das variáveis de controle (ver Capítulo 4).

A estimação dos parâmetros $\hat{\mathbf{w}}$ exige a determinação das funções de densidade $\mathbf{p_w(w)}$, $\mathbf{p_z(z)}$ e $\mathbf{p_{z/w}(z/w)}$. Estas três distribuições estarão completamente determinadas se for possível calcular os seus dois primeiros momentos, uma vez que se assume que elas são distribuições Gaussianas.

Para tanto, estima-se a priori os momentos de **w**, ou seja:

$$\boldsymbol{\varepsilon}\{\mathbf{w}\} = \overline{\mathbf{w}} \tag{8.a}$$

$$\mathbf{cov(w)} = \boldsymbol{\varepsilon}\{(\mathbf{w} - \overline{\mathbf{w}}) \cdot (\mathbf{w} - \overline{\mathbf{w}})^{\mathbf{T}}\} = \overline{\mathbf{P}} \tag{8.b}$$

onde,

$$\overline{\mathbf{w}} = \mathbf{w} + \overline{\mathbf{e}} \tag{8.c}$$

$$\boldsymbol{\varepsilon}\{\overline{\mathbf{e}}\,\overline{\mathbf{e}}^{\mathbf{T}}\} = \overline{\mathbf{P}} \tag{8.d}$$

Assumindo que a variável aleatória **w** é independente do erro **v**, ou seja,

$$\mathbf{cov(w, v) = 0} \tag{9}$$

E também que as variáveis aleatórias **w** e **z** são relacionadas linearmente como na equação abaixo

$$\mathbf{z} = \mathbf{H} \cdot \mathbf{w} + \mathbf{v} \tag{10}$$

Pode-se, então, demonstrar (Maybeck 1979) que os outros momentos desejados são dados por:

$$\boldsymbol{\varepsilon}\{\mathbf{z}\} = \mathbf{H} \cdot \overline{\mathbf{w}} \tag{11.a}$$

$$\mathbf{cov(z)} = \mathbf{H} \cdot \overline{\mathbf{P}} \cdot \mathbf{H}^{\mathbf{T}} + \mathbf{R} \tag{11.b}$$

$$\boldsymbol{\varepsilon}\{\mathbf{v}\mathbf{v}^{\mathbf{T}}\} = \mathbf{R} \tag{11.c}$$

$$\boldsymbol{\varepsilon}\{\mathbf{z/w}\} = \mathbf{H} \cdot \mathbf{w} \tag{11.d}$$

$$\mathbf{cov(z/w)} = \mathbf{cov(v/w)} = \mathbf{R} \tag{11.e}$$

Partindo-se das equações (11.a) à (11.e) e assumindo que todas as variáveis aleatórias estudadas aqui sejam Gaussianas (Jazwinski, 1970), chega-se, após alguns

desenvolvimentos analíticos, às seguintes expressões para as funções de densidade vetorial no espaço n-dimensional:

$$\mathbf{p_w(w) = \frac{1}{(2\pi)^n \cdot |\overline{P}|^{1/2}} \cdot exp\left[-\frac{1}{2} \cdot (w - \overline{w})^T \cdot \overline{P}^{-1} \cdot (w - \overline{w}) \right]} \tag{12.a}$$

$$\mathbf{p_z(z) = \frac{1}{(2\pi)^n \cdot |P_z|^{1/2}} \cdot exp\left[-\frac{1}{2} \cdot (z - H \cdot \overline{w})^T \cdot (P_z)^{-1} \cdot (z - H \cdot \overline{w}) \right]} \tag{12.b}$$

onde,

$$\mathbf{P_z = H \cdot \overline{P} \cdot H^T + R} \tag{12.c}$$

$$\mathbf{p_{z/w}(z/w) = \frac{1}{(2\pi)^n \cdot |R|^{1/2}} \cdot exp\left[-\frac{1}{2} \cdot (z - H \cdot w)^T \cdot R^{-1} \cdot (z - H \cdot w) \right]} \tag{12.d}$$

O símbolo $|\mathbf{A}|$ empregado nas expressões acima denota o determinante da matriz **A**. Substituindo as expressões (12.a), (12.b), (12.c) e (12.d) na relação de Bayes dada pela equação (5), obtem-se após alguns desenvolvimentos algébricos (e.g., Maybeck, 1979, Stengel, 1986), que:

$$\mathbf{p_{w/z}(w/z) = \frac{|H \cdot \overline{P} \cdot H^T + R|^{1/2}}{(2\pi)^n \cdot |\overline{P}|^{1/2} \cdot |R|^{1/2}} \cdot exp\left[-\frac{1}{2} \cdot (w - \hat{w})^T \cdot R^{-1} \cdot (w - \hat{w}) \right]} \tag{13.a}$$

onde,

$$\mathbf{\hat{w} = \varepsilon\{w/z\} = \overline{w} + \hat{P} \cdot H^T \cdot R^{-1} \cdot (z - H \cdot \overline{w})} \tag{13.b}$$

$$\mathbf{\hat{P} = cov(w/z) = (\overline{P}^{-1} + H^T \cdot R^{-1} \cdot H)^{-1}} \tag{13.c}$$

As equações (13.b) e (13.c) representam portanto as estimativas dos parâmetros do vetor **w**, em função das estimativas a priori $\overline{\mathbf{w}}$ e $\overline{\mathbf{P}}$. Como, neste caso, todas as observações

são processadas simultaneamente, então a solução dada por elas é dita estar na forma de *lotes*.

Em análise numérica algumas identidades matriciais são bastante importantes. A seguir é usada uma delas. Admita $\overline{\mathbf{P}}$, **R** e **H** serem de dimensões **n x n**, **m x m**, e **m x n**, respectivamente. Suponha $\overline{\mathbf{P}} \geq \mathbf{0}$ e $\mathbf{R} > \mathbf{0}$. Então a seguinte igualdade é verdadeira (Jazwinski, 1970):

$$(\overline{\mathbf{P}}^{-1}+\mathbf{H}^{\mathbf{T}}\cdot\mathbf{R}^{-1}\cdot\mathbf{H})^{-1}= \overline{\mathbf{P}} - \overline{\mathbf{P}}\cdot\mathbf{H}^{\mathbf{T}}\cdot(\mathbf{H}\cdot\overline{\mathbf{P}}\cdot\mathbf{H}^{\mathbf{T}}+\mathbf{R})^{-1}\cdot\mathbf{H}\cdot\overline{\mathbf{P}} \quad \text{para } \mathbf{P} > \mathbf{0} \tag{14}$$

Comparando a equação (14) acima com a equação (13.c) pode-se concluir que:

$$\hat{\mathbf{P}} = (\overline{\mathbf{P}}^{-1}+\mathbf{H}^{\mathbf{T}}\cdot\mathbf{R}^{-1}\cdot\mathbf{H})^{-1}\equiv \overline{\mathbf{P}} - \overline{\mathbf{P}}\cdot\mathbf{H}^{\mathbf{T}}\cdot(\mathbf{H}\cdot\overline{\mathbf{P}}\cdot\mathbf{H}^{\mathbf{T}}+\mathbf{R})^{-1}\cdot\mathbf{H}\cdot\overline{\mathbf{P}} \tag{15}$$

Quando não se conhece nenhuma informação a priori dos parâmetros é interessante observar, que nesta situação em particular, tudo se passa como se $\overline{\mathbf{w}} = \mathbf{0}$ e $\overline{\mathbf{P}} \to \infty$. Passando estes limites nas equações (13.b) e (13.c), tem-se:

$$\hat{\mathbf{w}} = \hat{\mathbf{P}}\cdot\mathbf{H}^{\mathbf{T}}\cdot\mathbf{R}^{-1}\cdot\mathbf{z} \tag{16.a}$$

$$\hat{\mathbf{P}} = (\mathbf{H}^{\mathbf{T}}\cdot\mathbf{R}^{-1}\cdot\mathbf{H})^{-1} \tag{16.b}$$

Substituindo a equação (16.b) em (16.a), vem:

$$\hat{\mathbf{w}} = (\mathbf{H}^{\mathbf{T}}\cdot\mathbf{R}^{-1}\cdot\mathbf{H})^{-1}\cdot\mathbf{H}^{\mathbf{T}}\cdot\mathbf{R}^{-1}\cdot\mathbf{z} \tag{17}$$

Se a matriz dos ruídos for igual a matriz identidade, resulta:

$$\hat{\mathbf{w}} = (\mathbf{H}^{\mathbf{T}}\cdot\mathbf{H})^{-1}\cdot\mathbf{H}^{\mathbf{T}}\cdot\mathbf{z} = \mathbf{H}^{\#}\cdot\mathbf{z} \tag{18}$$

onde,

$\mathbf{H}^{\#}$ ... é a matriz pseudo-inversa de **H**.

Portanto, quando não se conhece nada a priori a respeito das informações do estimador e a matriz dos ruídos for igual a matriz identidade, o método estocástico de estimação reduz-se ao da pseudo-inversa (Gelb et al, 1996). Outro teorema bastante importante de equivalência é enunciado abaixo.

Teorema 02: A estimação estocástica de parâmetros é formalmente equivalente a solução dos mínimos quadrados linear (Jazwinski, 1970), ou seja, da minimização do funcional dado por,

$$\mathbf{J} = \frac{\mathbf{1}}{\mathbf{2}} \cdot \left[(\mathbf{w} - \overline{\mathbf{w}})^{\mathbf{T}} \cdot \overline{\mathbf{P}}^{\mathbf{-1}} \cdot (\mathbf{w} - \overline{\mathbf{w}}) + (\mathbf{z} - \mathbf{H} \cdot \mathbf{w})^{\mathbf{T}} \cdot \mathbf{R}^{\mathbf{-1}} \cdot (\mathbf{z} - \mathbf{H} \cdot \mathbf{w}) \right] \tag{19}$$

A demonstração do teorema 02 é obtida através da diferenciação direta. Para tanto, deve-se fazer $\frac{\partial \mathbf{J}}{\partial \mathbf{w}} = 0$ e isolar a variável $\hat{\mathbf{w}}$. A diferença fundamental entre as estimações estocásticas e a dos mínimos quadrados é que, a primeira, define formalmente as matrizes $\overline{\mathbf{P}}$ e **R** como matrizes de covariâncias e a segunda utiliza-se de técnicas heurísticas para a determinação destas matrizes como matrizes de ponderação.

Outra grande vantagem da estimação estocástica linear é que ela determina também a matriz de covariâncias $\hat{\mathbf{P}}$ dos parâmetros estimados $\hat{\mathbf{w}}$. Não é possível determinar esta matriz à partir dos mínimos quadrados. Em outras palavras, os métodos determinísticos de otimização são sempre casos particulares e menos rigorosos dos estocásticos.

2.2.1 Estimação de Parâmetros Pelo Método Recursivo.

A determinação da matriz de covariâncias $\hat{\mathbf{P}}$ dos parâmetros do vetor aleatório **w** exige a inversão do termo matricial $(\mathbf{H} \cdot \overline{\mathbf{P}} \cdot \mathbf{H}^{\mathbf{T}} + \mathbf{R})^{\mathbf{-1}}$ presente na equação (15), cuja dimensão é **m x m**, ou seja, igual ao número total de observações. Em estimação de parâmetros utilizando redes neurais artificiais é comum utilizar mil ou mais padrões de treinamento, o que inviabiliza totalmente a solução em lotes. Para evitar a inversão matricial e

conseqüentes instabilidades numéricas (Jazwinski, 1970) propõe como solução o *processamento recursivo* de parâmetros.

No caso em que **R** é diagonal, isto é, os erros de observação são não correlacionados, o processamento recursivo permite que os termos matriciais da equação (15) sejam processados *linha a linha*, isto é, a inversão de matrizes transforma-se em *inversões sucessivas de escalares*. Para o problema de estimação estocástica linear de **n** parâmetros sobre **m** observações ($\mathbf{m \geq n}$) seja então **H(t)**, **z(t)**, **w** e **v(t)** definidos, respectivamente, por:

$\mathbf{H(t) = [h_{t1}\ h_{t2} \ldots h_{tn}]}$ ... vetor da t-ésima linha da matriz $\mathbf{H = [h_{ij}]_{m\ x\ n}}$;

z(t) ... t-ésimo elemento do vetor de observações $\mathbf{z = [z_i]_{m\ x1}}$;

w ... vetor de parâmetros $\mathbf{w^T = [\ w_1\ \ w_2\ \ \ldots w_n\ ]}$;

v(t) ... t-ésimo elemento do vetor de ruídos $\mathbf{v = [v_i]_{\ m\ x\ 1}}$.

O processamento recursivo dos parâmetros é garantido pelo teorema 03 enunciado, sem demonstração, a seguir:

Teorema 03: Se os erros das observações forem independentes entre si, isto é, $\boldsymbol{\varepsilon\{v_i\} = 0}$, $\boldsymbol{\varepsilon\{v_i \cdot v_i{}^t\} = R_i}$ e $\boldsymbol{\varepsilon\{v_i \cdot v_j{}^T\} = 0}$, então a solução pelo processamento recursivo conduz a mesma solução obtida pelo processamento em lotes. A solução recursiva poderá então, ser encontrada pelas seguintes expressões,

$$\mathbf{P(t) = \overline{P}(t) - \overline{P}(t) \cdot H^T(t) \cdot [H(t) \cdot \overline{P}(t) \cdot H^T(t) + R(t)]^{-1} \cdot H(t) \cdot \overline{P}(t)} \qquad (20.a)$$

$$\mathbf{\hat{w}(t) = \overline{w}(t) + P(t) \cdot H^T(t) \cdot R^{-1}(t) \cdot [z(t) - H(t) \cdot \overline{w}(t)]} \qquad \mathbf{t=1, 2, ..., m} \qquad (20.b)$$

$$\mathbf{\overline{w}(t+1) = \hat{w}(t), \qquad \overline{P}(t+1) = P(t)} \qquad (20.c)$$

para,

$$\overline{\mathbf{P}}(\mathbf{1}) = \overline{\mathbf{P}} \tag{20.d}$$

$$\overline{\mathbf{w}}(\mathbf{1}) = \overline{\mathbf{w}} \tag{20.e}$$

$$\hat{\mathbf{w}} = \hat{\mathbf{w}}(\mathbf{m}) \tag{20.f}$$

$$\hat{\mathbf{P}} = \mathbf{P}(\mathbf{m}) \tag{20.g}$$

onde,

P(t) ... matriz de covariâncias (**n x n**) que contêm o processamento das primeiras **t** observações;

$\hat{\mathbf{w}}(\mathbf{t})$ ... vetor (**n x 1**) dos parâmetros que contêm o processamento das primeiras **t** observações.

A análise dimensional da equação (20.a) torna evidente que as operações de inversão em cada recursão **t** são escalares e não mais matriciais. É interessante observar que na *estimação linear recursiva* as medidas que já foram processadas até a observação **t-1** estão armazenadas nas informações a priori $\overline{\mathbf{w}}(\mathbf{t})$ e $\overline{\mathbf{P}}(\mathbf{t})$. Este é um teorema bastante importante na estimação linear.

2.2.2 Introduzindo o Ganho de Kalman.

As equações (13.b), (13.c) ou (15), isto é, aquelas que resultam como solução das estimativas dos parâmetros do vetor aleatório **w** podem ser colocadas ou representadas através do ganho de Kalman **k** (Jazwinski, 1970). Introduzindo o ganho de Kalman na estimação linear estocástica resulta:

$$\mathbf{k} = \overline{\mathbf{P}} \cdot \mathbf{H}^{\mathbf{T}} \cdot [\mathbf{H} \cdot \overline{\mathbf{P}} \cdot \mathbf{H}^{\mathbf{T}} + \mathbf{R}]^{-1} \tag{21.a}$$

$$\hat{\mathbf{w}} = \overline{\mathbf{w}} + \mathbf{k} \cdot [\mathbf{z} - \mathbf{H} \cdot \overline{\mathbf{w}}] \tag{21.b}$$

$$\mathbf{P} = [\mathbf{I} - \mathbf{k} \cdot \mathbf{H}] \cdot \overline{\mathbf{P}} \qquad (21.c)$$

onde,

$\mathbf{k}_{\mathbf{n\,x\,m}}$ ... ganho de Kalman;

$\mathbf{I}_{\mathbf{n\,x\,n}}$ ... denota a matriz identidade de dimensão **n**;

m ... número total de observações;

n ... número total de parâmetros a serem estimados.

A forma como as equações de Kalman aparecem possui a vantagem de representar o incremento na estimativa como proporcional a ($\mathbf{z} - \mathbf{H} \cdot \overline{\mathbf{w}}$) onde o ganho **k** é o fator de proporcionalidade. Isto facilita a análise dos resultados quando da saturação da estimativa, fenômeno este muito comum de ocorrer no caso de sistemas de estimação não-lineares. As expressões na forma do ganho de Kalman também podem ser processadas recursivamente (Maybeck, 1979) como segue,

$$\mathbf{k(t)} = \overline{\mathbf{P}}\mathbf{(t)} \cdot \mathbf{H^T(t)} \cdot [\mathbf{H(t)} \cdot \overline{\mathbf{P}}\mathbf{(t)} \cdot \mathbf{H^T(t)} + \mathbf{R(t)}]^{-1} \qquad (22.a)$$

$$\hat{\mathbf{w}}\mathbf{(t)} = \overline{\mathbf{w}}\mathbf{(t)} + \mathbf{k(t)} \cdot [\mathbf{z(t)} - \mathbf{H(t)} \cdot \overline{\mathbf{w}}\mathbf{(t)}] \qquad (22.b)$$

$$\overline{\mathbf{w}}\mathbf{(t+1)} = \hat{\mathbf{w}}\mathbf{(t)} \qquad (22.c)$$

$$\overline{\mathbf{P}}\mathbf{(t+1)} = [\mathbf{I} - \mathbf{k(t)} \cdot \mathbf{H(t)}] \cdot \overline{\mathbf{P}}\mathbf{(t)} \qquad (22.d)$$

$$\hat{\mathbf{w}} = \hat{\mathbf{w}}\mathbf{(m)} \qquad (22.e)$$

$$\mathbf{P} = [\mathbf{I} - \mathbf{k(m)} \cdot \mathbf{H(m)}] \cdot \overline{\mathbf{P}}\mathbf{(m)} \qquad (22.f)$$

$\overline{\mathbf{w}}\mathbf{(1)} = \overline{\mathbf{w}}$ e $\overline{\mathbf{P}}\mathbf{(1)} = \overline{\mathbf{P}}$ onde, $\mathbf{t = 1, 2, ..., m}$ (22.g)

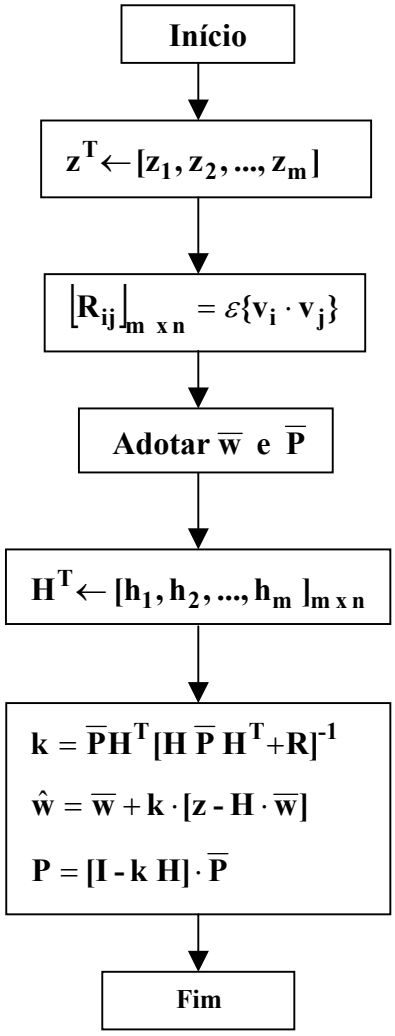


FIGURA 2.1.a – Processamento em lotes do filtro de Kalman.

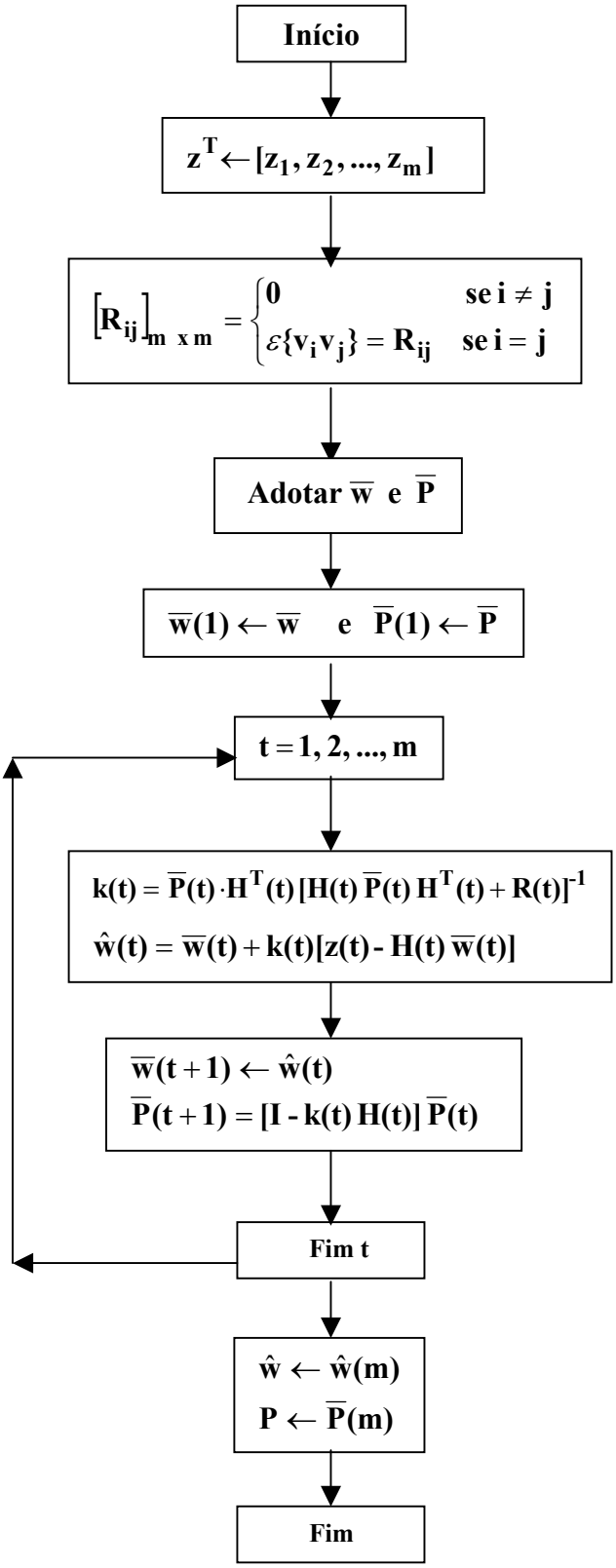


FIGURA 2.1.b – Processamento recursivo do filtro de Kalman.

Em geral, as equações mais utilizadas em estimação estocástica de parâmetros são as do ganho de Kalman na forma recursiva. Os fluxogramas para a computação das estimativas dos parâmetros, através das equações do ganho de Kalman, são apresentados nas Figuras 2.1.a e 2.1.b. Estas figuras tem o objetivo de distinguir a diferença básica existente entre o processamento em lotes e o recursivo.

2.3 A Estimação de Parâmetros no Caso Não-Linear.

Quando não é possível separar linearmente as observações dos parâmetros, na função de estimação, então tem-se o caso não-linear de estimação de parâmetros. A equação (23.a) exemplifica sucintamente esta situação.

$$\hat{\mathbf{y}}(t) = \hat{\mathbf{f}}(\mathbf{x}(t),\mathbf{w}) \neq \mathbf{H}(\mathbf{x}(t)) \cdot \mathbf{w} \qquad (23.a)$$

$$\{\mathbf{x}(t), \mathbf{y}(t)\} : t = 1, 2, ..., L \qquad (23.b)$$

onde,

$\{\mathbf{x}(t), \mathbf{y}(t)\} : t = 1, 2, ..., L$ ... são, respectivamente, os dados de entrada e saída, isto é, $\mathbf{y}(t)$ são valores observados de $\hat{\mathbf{f}}(\mathbf{x}(t),\mathbf{w})$. No caso de treinamento de redes neurais são os padrões de treinamento, tendo-se então:

$\mathbf{w}$ ... o vetor de parâmetros a ser estimado;

$\hat{\mathbf{y}}(t)$ ... o vetor de estimativas dos padrões de saída;

$\hat{\mathbf{f}}(\mathbf{x}(t), \mathbf{w})$ ... a função não-linear, representando, no caso, a rede neural, que relaciona os parâmetros com as observações.

A solução para esta classe de estimadores pode ter as mesmas características daquelas com o filtro de Kalman no caso linear. Entretanto, isto será válido somente se a equação (23.a) puder ser expandida em série de Taylor até os termos de *primeira*

ordem. A diferença fundamental entre o caso não-linear para o linear fica, então, em reconhecer que no primeiro a solução passa a ser *iterativa*, ou seja, realizam-se várias linearizações sucessivas sobre os parâmetros gerados pelo ganho de Kalman, até que seja atingida a convergência do problema. No caso não-linear, assim como no linear, é possível também processar a solução do problema tanto em *lotes* como *recursivamente* empregando-se as mesmas equações do filtro de Kalman dadas por (20.a) à (20.g).

A expansão em série de Taylor da equação (23.a) até os termos de primeira ordem, resulta em

$$\alpha(\mathbf{i}) \cdot [\mathbf{y(t)} - \bar{\mathbf{y}}(\mathbf{t,i})] \cong \hat{\mathbf{f}}_{\mathbf{w}}(\mathbf{x(t)}, \bar{\mathbf{w}}(\mathbf{i})) \cdot [\mathbf{w(i)} - \bar{\mathbf{w}}(\mathbf{i})] \tag{24}$$

onde,

$\mathbf{0} < \alpha(\mathbf{i}) \leq \mathbf{1}$ ... parâmetro empírico a ser ajustado para garantir a hipótese de linearização;

$\hat{\mathbf{f}}_{\mathbf{w}}(\mathbf{x(t)},\bar{\mathbf{w}}(\mathbf{i}))$ ... matriz das primeiras derivadas com respeito a **w** ou o *Jacobiano* da função de estimação com relação aos parâmetros ;

$\bar{\mathbf{w}}(\mathbf{i})$ ... é a estimativa a priori tomada da iteração anterior e iniciada em $\bar{\mathbf{w}}(\mathbf{1}) = \bar{\mathbf{w}}$. É o ponto em torno do qual se realiza a linearização da função $\hat{\mathbf{f}}(\mathbf{x(t),w})$;

$\bar{\mathbf{y}}(\mathbf{i}) = \hat{\mathbf{f}}(\mathbf{x}, \bar{\mathbf{w}}(\mathbf{i}))$ ... valor da função $\hat{\mathbf{f}}(\mathbf{.})$ sobre o ponto no qual se realizou a linearização;

$\mathbf{y(i)}$ ... vetor dos padrões de treinamento.

Para reduzir o problema atual para uma forma equivalente à linear é conveniente adotar as seguintes notações compactas:

$$\mathbf{z(t,i)} \equiv \alpha(\mathbf{i}) \cdot [\mathbf{y(t)} - \bar{\mathbf{y}}(\mathbf{t,i})] + \hat{\mathbf{f}}_{\mathbf{w}}(\mathbf{x(t)},\bar{\mathbf{w}}(\mathbf{i})) \cdot \bar{\mathbf{w}}(\mathbf{i}) \tag{25.a}$$

$$\mathbf{H(t,i) \equiv \hat{f}_w(x(t), \overline{w}(i))} \tag{25.b}$$

onde,

$$\overline{\mathbf{w}} = [\overline{\mathbf{w}}_1, \overline{\mathbf{w}}_2, ..., \overline{\mathbf{w}}_n]^T \tag{25.c}$$

$$\mathbf{H(t,i)} = \hat{\mathbf{f}}_w(\mathbf{x(t)}, \overline{\mathbf{w}}(\mathbf{i})) = \begin{bmatrix} \frac{\partial \overline{\mathbf{y}}(1,\mathbf{i})}{\partial \overline{\mathbf{w}}_1} & \cdots & \frac{\partial \overline{\mathbf{y}}(1,\mathbf{i})}{\partial \overline{\mathbf{w}}_n} \\ \cdots & \cdots & \cdots \\ \frac{\partial \overline{\mathbf{y}}(\mathbf{m},\mathbf{i})}{\partial \overline{\mathbf{w}}_1} & \cdots & \frac{\partial \overline{\mathbf{y}}(\mathbf{m},\mathbf{i})}{\partial \overline{\mathbf{w}}_n} \end{bmatrix}_{\mathbf{m \times n}} \tag{25.d}$$

O índice **i** acrescido às equações (24), (25.a) à (25.d) evidencia a característica *iterativa* da estimação não-linear, onde, **i=1, 2, ..., I**. A partir das equações (25.a) e (25.b) é possível formular o seguinte problema de estimação estocástica:

$$\overline{\mathbf{w}}(\mathbf{i}) = \mathbf{w(i)} + \overline{\mathbf{e}} \tag{26.a}$$

$$\mathbf{z(t,i)} = \mathbf{H(t,i)} \cdot \mathbf{w(i)} + \mathbf{v(t)} \qquad \text{para } \mathbf{t=1, 2, ..., m} \tag{26.b}$$

ou

$$\mathbf{z(i)} = \mathbf{H(i)} \cdot \mathbf{w(i)} + \mathbf{v} \tag{26.c}$$

onde,

$$\boldsymbol{\varepsilon}\{\overline{\mathbf{e}}\} = \mathbf{0}, \qquad \boldsymbol{\varepsilon}\{\overline{\mathbf{e}} \cdot \overline{\mathbf{e}}^T\} = \overline{\mathbf{P}} \tag{26.d}$$

$$\boldsymbol{\varepsilon}\{\mathbf{v(t)}\} = \mathbf{0}, \qquad \boldsymbol{\varepsilon}\{\mathbf{v(t)} \cdot \mathbf{v}^T\mathbf{(t)}\} = \mathbf{R(t)} \tag{26.e}$$

com $\overline{\mathbf{e}}$ e $\mathbf{v(t)}$ sendo não correlacionados, tendo distribuições Gaussianas. Com relação as equações (26.b) e (26.c) é interessante observar que a primeira está na forma recursiva e a segunda na forma de processamento em lotes.

Observe que a solução recursiva deve teoricamente gerar o mesmo resultado numérico que a solução em lotes. Entretanto, na prática a solução recursiva para estimadores não-lineares pode diferir ligeiramente da em lotes, uma vez que os erros computacionais podem gerar pequenas discrepâncias numéricas entre estes dois métodos.

CAPÍTULO 3

REDES NEURAIS ARTIFICIAIS

3.1 Introdução.

O desenvolvimento de sistemas neurais artificiais iniciou-se em 1943 com a formulação do primeiro modelo matemático de um neurônio artificial elementar idealizado por Warren McCulloch (psiquiatra) e Walter Pitts (matemático) num trabalho pioneiro (e.g., Kovács e Solt, 1996). Entretanto, foi somente em 1949 que Donald Hebb conseguiu propor o primeiro esquema de aprendizado para atualizar os pesos das conexões dos neurônios e que hoje é conhecido como regra de aprendizado Hebbiana. É importante observar que a regra de Hebb é ainda utilizada em vários algoritmos de treinamento. O primeiro neurônio artificial foi implementado computacionalmente em 1957 por Minsky onde os pesos das conexões eram adaptados automaticamente (Zurada, 1992) .

O modelo de neurônio artificial mais conhecido na literatura é o *perceptron* e foi proposto por Frank Rosenblat em 1958. O inconveniente deste neurônio é que ele é somente capaz de classificar padrões que sejam linearmente separáveis, ou seja, quando os padrões podem ser separados no espaço n-dimensional por um hiperplano.

A regra delta de aprendizado que é baseada no método do gradiente para minimizar o erro na saída de um neurônio com resposta linear foi proposta por Widrow e Holff em 1960 (Zurada, 1992). No início dos anos 60 novas contribuições foram feitas como o surgimento dos aparelhos chamados *Adaptive Linear Combiner* (ADALINE) e muitas Adalines (MADALINE). Uma regra de aprendizado poderosa foi proposta por Bernard Widrow e Marcian Hoff para treinamento destas redes. Esta regra conseguia minimizar a soma dos erros quadráticos durante o treinamento envolvendo modelos de classificação. As aplicações iniciais destas redes eram em controles adaptativos e reconhecimento de padrões.

O desenvolvimento teórico em redes neurais artificiais realizado até a primeira metade da década de 60 foi sumarizado num trabalho de Nilsson (1965). Até essa época, no

entanto, nenhum esquema eficiente de aprendizado existia para superar os problemas de classificação não-linear.

O fato citado acima foi o principal responsável para que o estudo de redes neurais artificiais entrasse numa fase de estagnação. É evidente também que a falta de computadores mais poderosos, nesta época, contribuiu para a interrupção parcial destas pesquisas. Para piorar ainda mais a situação (Minsky e Papert, 1969) publicaram um livro que consolidou mais dúvidas quanto ao potencial de aprendizado das redes neurais. Estes dois pesquisadores argumentaram que o *perceptron* não era capaz de detectar paridade, conectividade e simetria. Estes tipos de problemas são classificados, em geral, como não separáveis linearmente.

Devido a isso, as limitações das redes de classificação de neurônios *perceptron* foram então superadas somente em meados da década de 80. Entretanto, antes do ressurgimento definitivo das redes neurais, alguns trabalhos pioneiros foram realizados durante a década de 70 e início da década de 80. Neste período surgiram as redes *neocognitron*, *cognitron* e de Kohonen. Sistemas auto-adaptativos e redes sem pesos também têm suas origens nesta época.

Entre o período de 1982 e 1986 as publicações dos trabalhos exploravam mais o potencial de alcance das redes do que propriamente técnicas de treinamento. O renascimento das redes neurais ocorreu, propriamente dito, com a publicação de dois trabalhos. O primeiro está associado ao surgimento das redes neurais recorrentes de Hopfield em 1982 e o segundo referente a descoberta da retro-propagação em 1986 (Kovács e Solt, 1996).

Nestes dois trabalhos ficou confirmado que a visão dada em 1969 por Minsky e Papert era bastante pessimista. Os problemas difíceis de aprender começavam a ser resolvidos. Declarações mais recentes afirmam que os primeiros autores que abordaram a otimização de sistemas de rede *feedforward* multicamadas foram Bryson and Ho em 1969 e Kelley também em 1969 (Zurada, 1992). Estudos subsequentes revelaram também que a regra de retro-propagação foi proposta numa tese de doutorado de Werbos (1974). Não era utilizada, contudo, em treinamento de redes neurais.

A partir de 1986 as pesquisas na área tomaram um novo rumo com o trabalho de Rumelhart et al (1986) referente a retro-propagação. O número de trabalhos publicados, conferências e revistas aumentaram consideravelmente. As aplicações das redes neurais começaram a tomar grande praticidade. Em 1987 ocorreu em São Francisco a primeira conferência de redes neurais em tempos modernos, a *IEEE International Conference on Neural Networks*, e também foi formada a *International Neural Networks Society* (INNS). A partir destes acontecimentos decorreram a fundação do *INNS journal* em 1989, seguido do *Neural Computation* e do *IEEE Transactions on Neural Networks* em 1990.

Dois fatores têm contribuído para a difusão de trabalhos nesta área: desde 1987 muitas universidades anunciaram a formação de institutos de pesquisa e programas de educação em neuro computação; e o surgimento de *software* especializados com algoritmos de treinamento para a rede *feedforward* utilizando o conceito de retro-propagação. Outros fatores que contribuiram para o crescimento das pesquisas nesta área da ciência foram os insucessos obtidos pela escola simbolista da inteligência artificial (IA) e o crescimento bastante acentuado da microeletrônica.

3.2 Redes Neurais: Fundamentos.

Segundo Zurada (1992) as redes neurais artificiais podem ser definidas de várias formas. A seguir são dadas algumas destas definições básicas:

- redes neurais podem ser vistas essencialmente como uma notação gráfica para alguns algoritmos matemáticos desenvolvidos nas últimas décadas;

- as redes neurais podem ser interpretadas como sendo uma réplica simplificada que emula as redes neurais biológicas encontradas em organismos vivos;

- podem ser definidas também, como uma interconexão de neurônios, tal que as saídas de um neurônio são conectadas, através de pesos, para outros neurônios.

Para fins práticos, a última definição é a que melhor se enquadra nas aplicações computacionais e tecnológicas possíveis de serem alcançadas. É também a partir desta definição que se pode chegar ao conceito de neurônio como exemplificados nas Figuras 3.1 e 3.2. Todo neurônio é composto por uma entrada ponderada **s** aplicada a uma dinâmica linear na variável **x**. O resultado da dinâmica é então, aplicado a uma função não-linear **a(x)** como exemplificado na Figura 3.1.

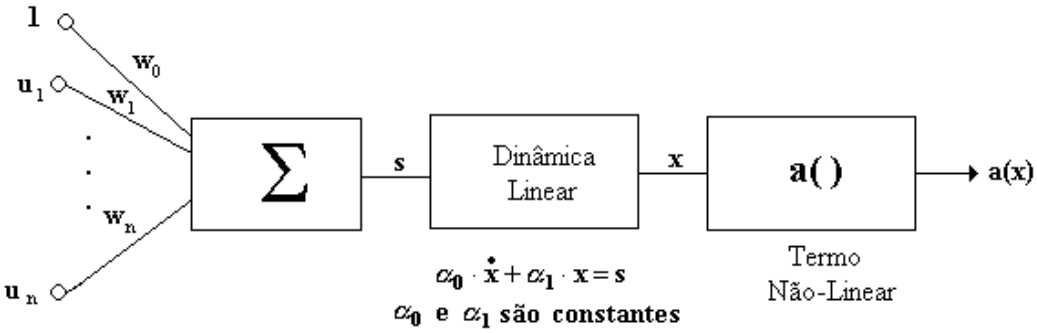


FIGURA 3.1 – Conceito de neurônio generalizado.

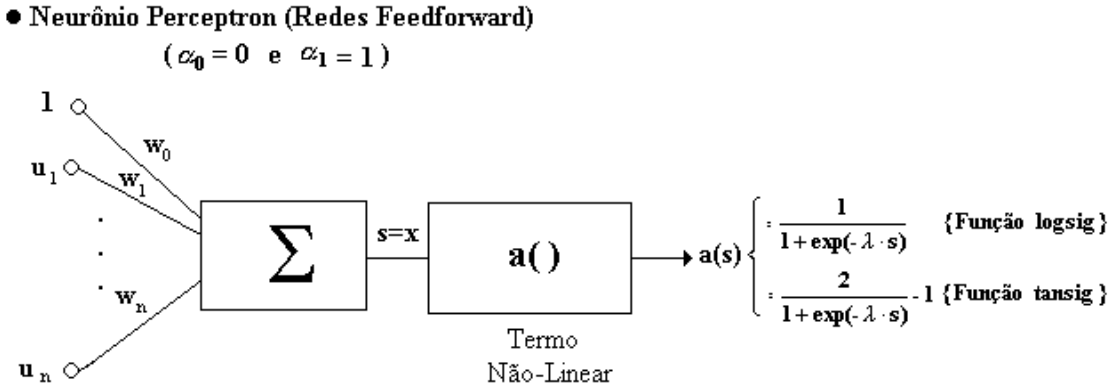


FIGURA 3.2 – O neurônio *perceptron*.

A dinâmica do neurônio é dada pela equação diferencial $\alpha_0 \cdot \dot{\mathbf{x}} + \alpha_1 \cdot \mathbf{x} = \mathbf{s}$. Para o caso particular do *perceptron*, neurônio mais utilizado nas redes *feedforward*, tem-se α_0**=0** e α_1**=1** (ver Figura 3.2).

Desta forma, um neurônio é uma unidade de processamento de informação que é fundamental para a operação de uma rede neural artificial. No caso do *perceptron* pode-se identificar três elementos básicos do modelo neural:

1) um conjunto de sinapses ou elos de conexão, cada uma caracterizada por um peso ou força própria;

2) um somador para adicionar os sinais ponderados de entrada que constitui a parte linear da rede;

3) uma função de ativação para restringir a amplitude de saída de um neurônio que constitui a parte não-linear do neurônio.

O modelo neural da Figura 3.2 inclui também um bias aplicado externamente, representado por $\mathbf{w_o}$. O bias $\mathbf{w_o}$ tem o efeito de aumentar ou diminuir a entrada líquida da função de ativação. Em termos matemáticos, pode-se descrever um neurônio a partir do seguinte par de equações:

$$\mathbf{s} = \sum_{\mathbf{j=1}}^{\mathbf{n}} \mathbf{w_j} \cdot \mathbf{u_j} + \mathbf{w_o} \tag{1.a}$$

$$\mathbf{a} = \mathbf{a\,(s)} \tag{1.b}$$

onde $\mathbf{u_1}$, $\mathbf{u_2}$, ..., $\mathbf{u_n}$ são os sinais de entrada; $\mathbf{w_1}$, $\mathbf{w_2}$, ..., $\mathbf{w_n}$ são os pesos sinápticos do neurônio; **s** é a saída do combinador linear devido aos sinais de entrada; $\mathbf{w_o}$ é o bias; **a(.)** é a função de ativação e **a(s)** é o sinal de saída do neurônio.

A utilização do bias $\mathbf{w_0}$ tem o efeito de aplicar uma transformação afim na saída do combinador linear, e como resultado desta transformação o gráfico de **a(s)** em função de **s** não passa mais pela origem. Outra questão muito importante é com relação à disposição ou interconexão destes neurônios (arquitetura da rede neural). Duas classes de redes neurais (Narendra e Parthasarathy, 1990) têm recebido considerável atenção na área de redes neurais artificiais em anos recentes: as redes *feedforward* multi-camadas e as redes recorrentes.

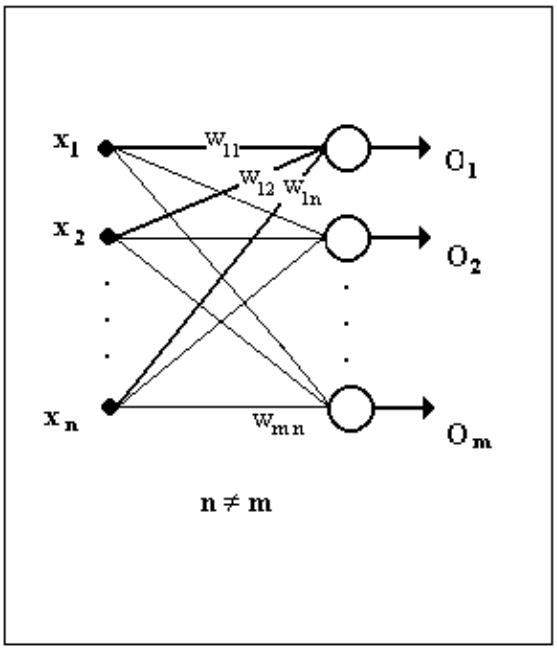
As redes *feedforward* têm tido extremo sucesso em problemas de reconhecimento de padrões, enquanto as redes recorrentes têm sido utilizadas em memórias associativas bem como para solução de problemas de otimização. As Figuras 3.3.a e 3.3.b ilustram, respectivamente, a arquitetura de cada uma dessas redes. Do ponto de vista teórico, as redes *feedforward* multi-camadas representam o mapeamento não-linear estático, enquanto que as redes recorrentes são representadas por sistemas dinâmicos não-lineares em feedback.

Segundo Narendra e Parthasarathy (1990), há muitos argumentos para afirmar que num futuro próximo sistemas neurais complexos conterão ambos os tipos de redes interagindo num sistema híbrido. É interessante observar que, assim como as funções de transferência estão para representar modelos lineares genéricos, as redes neurais estão para representar modelos não-lineares genéricos.

3.2.1 Treinamento das Redes Neurais com Arquitetura *Feedforward.*

O teorema de Kolmogorov estabelece que qualquer função contínua de **N** variáveis pode ser computada utilizando-se somente somatórios lineares de funções não-lineares continuamente crescentes e de somente uma variável. Este teorema é a base dos fundamentos teóricos das redes *feedforward* quanto ao aspecto de treinamento.

Entretanto, os resultados teóricos de Kolmogorov não podem explicar completamente os sucessos alcançados nas aplicações de redes neurais. Hornik et al (1989) demonstram que as redes *feedforward* multi-camadas são realmente capazes de aproximações universais de funções com qualquer grau de precisão que se deseja. Estes autores fazem uso do teorema de Stone-Weierstrass para estabelecer que a rede *feedforward* multi-camadas utilizando funções assintóticas com limites superior e inferior podem aproximar qualquer função de interesse, desde que provida de um número suficiente de unidades internas ou neurônios.



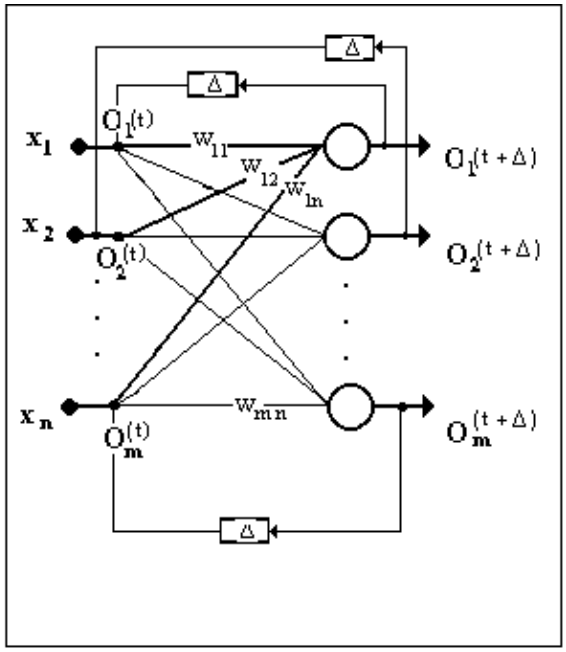


FIGURA 3.3.a – Rede *feedforward.*

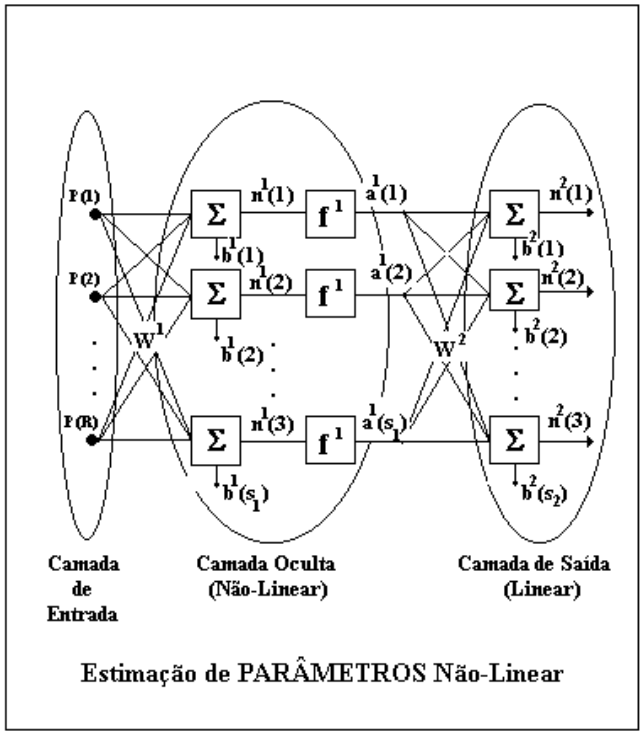
FIGURA 3.3.b – Rede recorrente.

No contexto de redes este teorema pode ser interpretado da seguinte forma: "Uma rede *feedforward* com uma única camada interna com termos não-lineares e uma camada de saída com termos lineares já é suficiente para interpolar qualquer função no espaço de $\mathbf{R^n} \rightarrow \mathbf{R^m}$." Este teorema estabelece as condições suficientes, mas não diz nada a respeito da estrutura da rede e de como chegar aos parâmetros da rede que interpolam a solução desejada em cada problema. Desta forma, treinar uma rede exige o emprego de algoritmos de otimização numérica não-lineares.

Há suporte teórico para as observações empíricas (Hunt et al,1992) que apontam que as redes com duas camadas internas aparentam possuir melhor precisão e generalização do que uma rede com uma simples camada interna, e com um custo mais baixo (menos unidade de processamento interno). Uma vez garantida a propriedade de aproximador universal de funções para as redes *feedforward*, como visto no início do Capítulo 1, então, pode-se cogitar sobre o processo de aprendizagem.

Pode-se definir aprendizagem no contexto de redes neurais como um processo pelo qual os parâmetros livres de uma rede são ajustados através da estimulação exercida pelo

ambiente externo, e o tipo de aprendizagem está intimamente relacionado na maneira pela qual a modificação dos parâmetros ocorre.



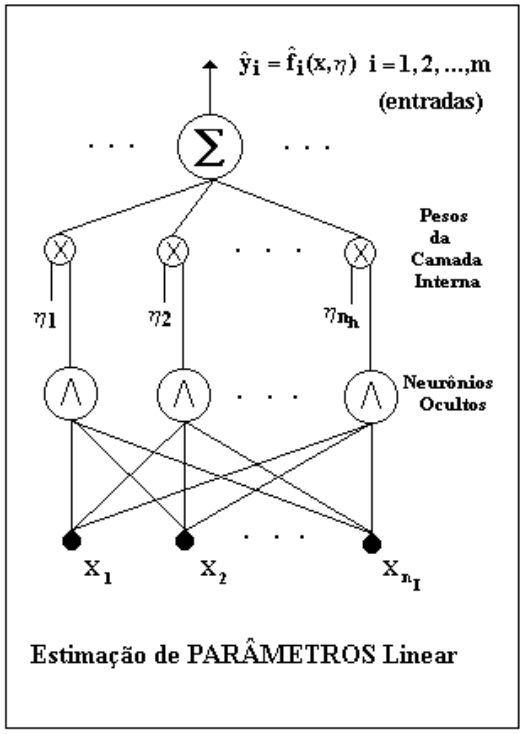


FIGURA 3.4.a – Rede *feedforward.*

FIGURA 3.4.b – Rede RBF.

No treinamento de uma rede feedforward a *aprendizagem por correção de erro* é a técnica mais utilizada para ensinar a rede a aprender uma tarefa específica. Nesta aprendizagem os vetores de valor desejado $\mathbf{y^d(t)}$ e de saída **y(t)** de uma determinada rede no instante t são considerados e o erro de saída da rede pode então ser representado por

$$\mathbf{e(t)=y^d(t) - y(t)} \qquad (2)$$

No aprendizado por correção de erro o sinal **e(t)** aciona um mecanismo de controle que produz uma seqüência de ajustes nos parâmetros da rede. Os ajustes tem a propriedade de corrigir passo a passo o sinal de saída **y(t)** em relação a resposta desejada $\mathbf{y^d(t)}$.

Este objetivo pode ser alcançado, em geral, minimizando uma função de custo ou índice de desempenho **J(t)** dado por:

$$\mathbf{J(t)} = \frac{\mathbf{1}}{\mathbf{2}} \cdot \mathbf{e}^{\mathbf{T}}\mathbf{(t)} \cdot \mathbf{e(t)} \tag{3}$$

Os ajustes são realizados passo a passo até o sistema atingir um erro aceitável. Neste instante, o processo é encerrado. Por razões óbvias o tipo de treinamento apresentado aqui recebe o nome de aprendizagem por correção sequencial de erro. Em geral, este é o tipo de aprendizagem utilizada para treinar as redes *feedforward* multicamadas.

Existem vários algoritmos para realizar o treinamento de uma rede *feedforward* empregando esta técnica de aprendizagem. De maneira geral, estes algoritmos podem ser divididos em duas categorias distintas: métodos de treinamento determinísticos e métodos de treinamento estocásticos. Entre os métodos determinísticos citam-se: do gradiente, do gradiente com momentum, de Quasi-Newton e o método de Levenberg-Marquardt. Entre os métodos estocásticos estão: a filtragem de Kalman na versão *full* ou com processamento paralelo e recursivo. Todos estes métodos possuem em comum o emprego da retro-propagação para o cálculo das derivadas parciais das variáveis de saída da rede em relação aos parâmetros de ajuste ou pesos sinápticos.

As seções seguintes tratarão da metodologia matemática empregada nos algoritmos de treinamento. Assim sendo, a próxima seção tratará exclusivamente da retro-propagação e em seguida será realizada a descrição detalhada do método da filtragem de Kalman com processamento paralelo e recursivo. O método do gradiente, algoritmo de treinamento neural mais utilizado, pode ser visto no Apêndice A, onde ele é bastante detalhado do ponto de vista matemático, sendo inclusive, comparado com o do filtro de Kalman.

3.2.2 Representação Matemática da Retro-propagação Numa Rede *Feedforward.*

A Figura 3.5 apresenta um esquema gráfico simplificado de uma camada genérica **k** em uma rede *feedforward* (Carrara, 1997). A representação matemática formal para o vetor de saída $\mathbf{x}^k$ da camada **k** será então dada pela seguinte expressão:

$$x_i^k = f^k\left(\sum_{j=1}^{n_{k-1}} w_{ij}^k \cdot x_j^{k-1} - b_i^k\right) \quad p/ \quad i = 1, 2, ..., n_k \tag{4}$$

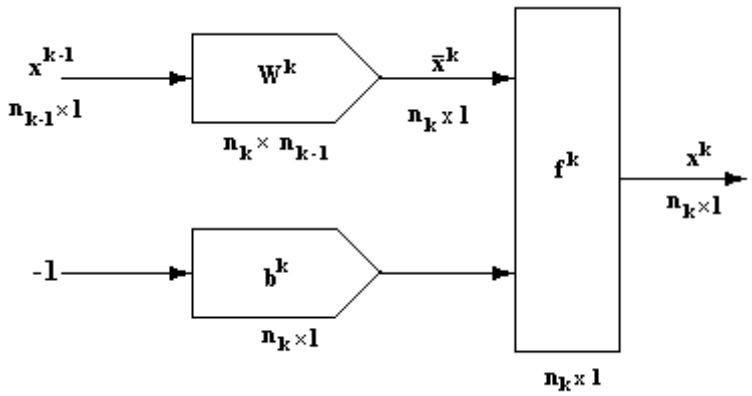


FIGURA 3.5 – Entradas e saídas de uma camada genérica k em uma rede com estrutura *feedforward* num diagrama simplificado.

O processamento analítico dado pela equação (4) pode ser colocado na forma matricial, demonstrando a característica de paralelismo apresentada pelas redes *feedforward*. Desta forma, sendo $\mathbf{W}^k$ a matriz de pesos da camada **k**, o vetor de saída desta camada será:

$$\mathbf{x}^k = \mathbf{f}^k\left(\mathbf{W}^k \cdot \mathbf{x}^{k-1}\right) \tag{5.a}$$

onde,

$$\mathbf{x}^k = [\, f^k(\bar{x}_1^k) \cdots f^k(\bar{x}_{n_k}^k) \, \text{-} \, 1]^T \tag{5.b}$$

$$\mathbf{W^k} = \begin{bmatrix} \mathbf{w^k_{11}} & \cdots & \mathbf{w^k_{1n_{k-1}}} & \mathbf{b^k_1} \\ \mathbf{w^k_{21}} & \cdots & \mathbf{w^k_{2n_{k-1}}} & \mathbf{b^k_2} \\ \vdots & \ddots & \vdots & \vdots \\ \mathbf{w^k_{n_k 1}} & \cdots & \mathbf{w^k_{n_k n_{k-1}}} & \mathbf{b^k_{n_k}} \end{bmatrix}_{\mathbf{n_k, n_{k-1}+1}} \tag{5.c}$$

Deve ser observado que na equação (5.c) foi acrescentado o vetor linha $\mathbf{b^k}$ à matriz de pesos $\mathbf{W^k}$, mas desde que não se esqueça de acrescentar o valor unitário negativo na última linha do vetor $\mathbf{x^k}$ representado pela equação (5.b).

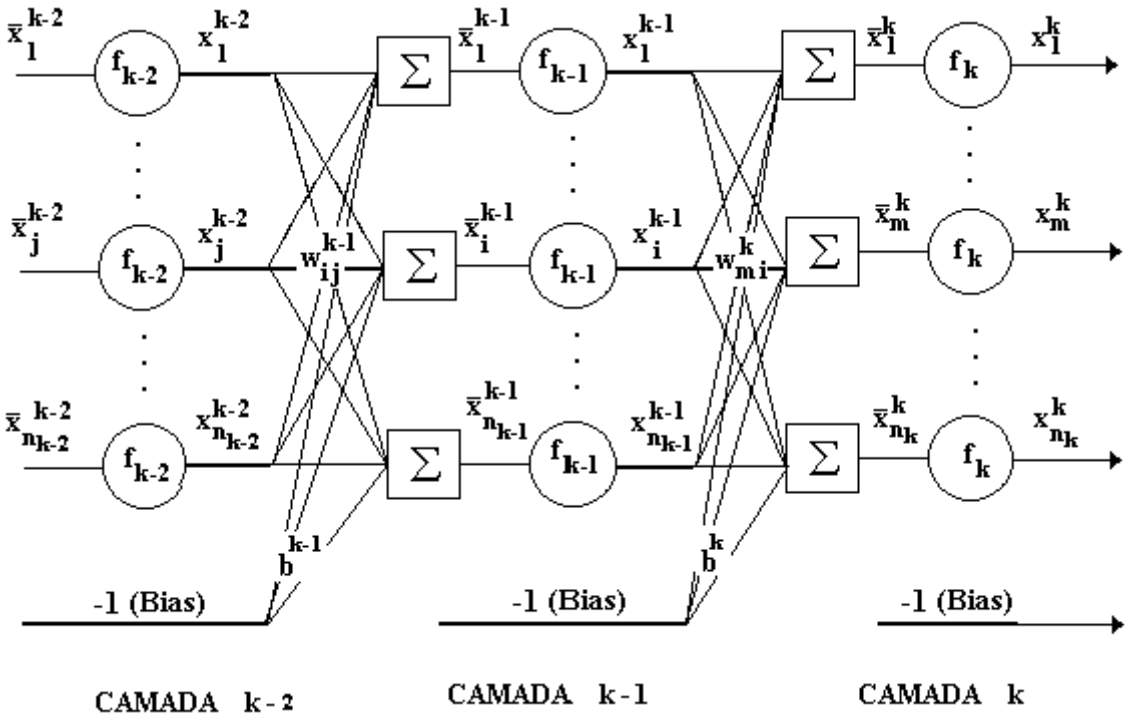


FIGURA 3.6 – Arquitetura detalhada de uma rede *feedforward*.

As derivadas das saídas $\mathbf{x^k_m}$ em relação aos pesos $\mathbf{w^k_{ni}}$ podem ser calculados da seguinte forma (ver Figura 3.6):

1) Se $\mathbf{n \neq m}$ então,

$$\frac{\partial \mathbf{x^k_m}}{\partial \mathbf{w^k_{ni}}} = 0 \tag{6.a}$$

pois neste caso a saída $\mathbf{x_m^k}$ é independente do peso $\mathbf{w_{ni}^k}$, uma vez que o peso $\mathbf{w_{ni}^k}$ só se comunica com a saída $\mathbf{x_n^k}$.

2) Se **n=m** então, tem-se que

$$\frac{\partial \mathbf{x_m^k}}{\partial \mathbf{w_{mi}^k}} = \frac{\partial \mathbf{f_k}\left(\bar{\mathbf{x}}_\mathbf{m}^\mathbf{k}\right)}{\partial \mathbf{w_{mi}^k}} = \frac{\partial}{\partial \mathbf{w_{mi}^k}} \cdot \left[\mathbf{f}_\mathbf{k} \left(\sum_{s=1}^{\mathbf{n_{k-1}}} \mathbf{x_s^{k-1}} \cdot \mathbf{w}_{\mathbf{ms}}^{k} \right) \right] = \mathbf{f_k'}(\bar{\mathbf{x}}_\mathbf{m}^\mathbf{k}) \cdot \underbrace{\frac{\partial}{\partial \mathbf{w}_{\mathbf{mi}}^{k}} \cdot \left(\sum_{\mathbf{s=1}}^{\mathbf{n_{k-1}}} \mathbf{x_s^{k-1}} \cdot \mathbf{w}_{\mathbf{ms}}^{k} \right)}_{\mathbf{x_i^{k-1}}}$$

Logo,

$$\frac{\partial \mathbf{x_m^k}}{\partial \mathbf{w}_{\mathbf{mi}}^{k}} = \mathbf{f_k'}\left(\bar{\mathbf{x}}_\mathbf{m}^\mathbf{k}\right) \cdot \mathbf{x_i^{k-1}} \tag{6.b}$$

O cálculo das derivadas das saídas genéricas da rede $\mathbf{x_m^k}$ em relação aos pesos $\mathbf{w_{ij}^{k-1}}$ exige um pouco mais de trabalho. Desta maneira, pode-se proceder da seguinte forma:

$$\frac{\partial \mathbf{x_m^k}}{\partial \mathbf{w_{ij}^{k-1}}} = \frac{\partial \mathbf{f_k}\left(\bar{\mathbf{x}}_\mathbf{m}^\mathbf{k}\right)}{\partial \mathbf{w_{ij}^{k-1}}} = \frac{\partial}{\partial \mathbf{w_{ij}^{k-1}}} \cdot \left[\mathbf{f}_\mathbf{k} \left(\sum_{\mathbf{s=1}}^{\mathbf{n_{k-1}}} \mathbf{w_{ms}^k} \cdot \mathbf{x_s^{k-1}} \right) \right]$$

$$= \mathbf{f_k'}\left(\bar{\mathbf{x}}_\mathbf{m}^\mathbf{k}\right) \cdot \frac{\partial}{\partial \mathbf{w_{ij}^{k-1}}} \left(\sum_{\mathbf{s=1}}^{\mathbf{n_{k-1}}} \mathbf{w_{ms}^k} \cdot \mathbf{x_s^{k-1}} \right)$$

$$= \mathbf{f_k'}\left(\bar{\mathbf{x}}_\mathbf{m}^\mathbf{k}\right) \cdot \sum_{\mathbf{s=1}}^{\mathbf{n_{k-1}}} \mathbf{w_{ms}^k} \cdot \frac{\partial \mathbf{x_s^{k-1}}}{\partial \mathbf{w_{ij}^{k-1}}} \tag{7}$$

O valor das derivadas $\frac{\partial \mathbf{x_s^{k-1}}}{\partial \mathbf{w_{ij}^{k-1}}}$ podem ser obtidas das expressões (6.a) e (6.b) que já foram deduzidas anteriormente. Desta forma, tem-se:

$$\sum_{\mathbf{s=1}}^{\mathbf{n_{k-1}}} \mathbf{w_{ms}^k} \cdot \frac{\partial \mathbf{x_s^{k-1}}}{\partial \mathbf{w_{ij}^{k-1}}} = \mathbf{w_{mi}^k} \cdot \frac{\partial \mathbf{x_i^{k-1}}}{\partial \mathbf{w_{ij}^{k-1}}} = \mathbf{w_{mi}^k} \cdot \mathbf{f_{k-1}'}\left(\bar{\mathbf{x}}_\mathbf{i}^\mathbf{k-1}\right) \cdot \mathbf{x_j^{k-2}} \tag{8}$$

Substituindo a equação (8) em (7), resulta finalmente que:

$$\frac{\partial \mathbf{x}_{\mathbf{m}}^{\mathbf{k}}}{\partial \mathbf{w}_{\mathbf{ij}}^{\mathbf{k-1}}} = \mathbf{f}_{\mathbf{k}}^{'}\left(\overline{\mathbf{x}}_{\mathbf{m}}^{\mathbf{k}}\right) \cdot \mathbf{w}_{\mathbf{mi}}^{\mathbf{k}} \cdot \mathbf{f}_{\mathbf{k-1}}^{'}\left(\overline{\mathbf{x}}_{\mathbf{i}}^{\mathbf{k-1}}\right) \cdot \mathbf{x}_{\mathbf{j}}^{\mathbf{k-2}} \tag{9}$$

O fato das derivadas dos pesos da camada **k** terem sidos utilizados para o cálculo das derivadas dos pesos da camada **k-1**, demonstra o caráter recursivo da retro-propagação. Observa-se que primeiro se determinam as derivadas da última camada e só depois disso é que se determinam as derivadas da camada anterior. Por essa razão o fluxo do cálculo destas derivadas caminha no sentido contrário ao fluxo dos sinais de saída da rede. Desta forma, supondo que **l** seja a última camada da rede, então pode-se obter a seguinte fórmula recorrente:

$$\frac{\partial \mathbf{x}_{\mathbf{m}}^{\mathbf{l}}}{\partial \mathbf{w}_{\mathbf{ij}}^{\mathbf{k}}} = \delta_{\mathbf{m,i}}^{\mathbf{k}} \cdot \mathbf{x}_{\mathbf{j}}^{\mathbf{k-1}} \tag{10.a}$$

onde os escalares $\delta_{\mathbf{m,i}}^{\mathbf{k}}$ podem ser obtidos a partir de:

$$\delta_{\mathbf{m,i}}^{\mathbf{k}} = \mathbf{f}_{\mathbf{k}}^{'}\left(\overline{\mathbf{x}}_{\mathbf{i}}^{\mathbf{k}}\right) \cdot \sum_{\mathbf{s=1}}^{\mathbf{n_{k+1}}} \delta_{\mathbf{m,s}}^{\mathbf{k+1}} \cdot \mathbf{w}_{\mathbf{si}}^{\mathbf{k+1}} \tag{10.b}$$

Com a condição inicial sendo dada por:

$$\delta_{\mathbf{m,i}}^{\mathbf{l}} = \begin{cases} \mathbf{f}_{\mathbf{l}}^{'}\left(\overline{\mathbf{x}}_{\mathbf{i}}^{\mathbf{l}}\right), \ \mathbf{se} \ \ \mathbf{i = m} \\ \\ \mathbf{0} \quad\quad , \ \mathbf{se} \ \ \mathbf{i \neq m} \end{cases} \tag{10.c}$$

Ainda existe a versão matricial para as equações (10.a), (10.b) e (10.c). Entretanto por envolverem matrizes de dimensões muito altas e com elementos nulos fora da diagonal principal elas não possuem muita praticidade computacional. Para maiores detalhes da aplicação destas equações diretamente no método do gradiente de treinamento neural ver o Apêndice A.

3.2.3 A Filtragem de Kalman com Processamento Paralelo Aplicada ao Treinamento das Redes *Feedforward.*

O treinamento de uma rede feedforward é usualmente feito por *treinamento supervisionado* do mapeamento dos conjuntos de dados:

$$\left\{ \left(\mathbf{x(t), y(t)}\right) \mathbf{: y(t) = f}\left(\mathbf{x(t)}\right), \mathbf{t = 1, 2, ..., L} \right\} \tag{11}$$

Os parâmetros de pesos devem ser ajustados para, aproximadamente, adaptar o sistema neural artificial, correspondente ao modelo computacional, para estes dados de entrada/saída. O processamento por treinamento do sistema neural artificial dos dados de entrada **x(t)**, para produzir saídas $\hat{\mathbf{y}}\mathbf{(t)}$, pode ser visto como um mapeamento parametrizado:

$$\hat{\mathbf{y}}\mathbf{(t)} = \hat{\mathbf{f}}\left(\mathbf{x(t), w}\right) \tag{12}$$

Na expressão acima **w** é o vetor dos parâmetros de pesos. A abordagem usual para resolver o problema de treinamento supervisionado de sistemas neurais *feedforward* é minimizar, com respeito ao vetor de pesos w, o funcional:

$$\begin{aligned} \mathbf{J} = \frac{\mathbf{1}}{\mathbf{2}} \cdot [& \left(\mathbf{w} - \overline{\mathbf{w}}\right)^{\mathbf{T}} \overline{\mathbf{P}}^{\mathbf{-1}} \left(\mathbf{w} - \overline{\mathbf{w}}\right) + \\ & \sum_{\mathbf{t=1}}^{\mathbf{L}} \left(\mathbf{y(t)} - \hat{\mathbf{f}}\mathbf{(x(t), w)}\right)^{\mathbf{T}} \mathbf{R}^{\mathbf{-1}}\mathbf{(t)} \left(\mathbf{y(t)} - \hat{\mathbf{f}}\mathbf{(x(t), w)}\right)] \end{aligned} \tag{13}$$

Para a estimativa de **w** gerar bons resultados devem ser dados os valores de entrada/saída **{x(t), y(t): t=1,2,...,L}**, uma estimativa a priori de $\overline{\mathbf{w}}$ e as matrizes de peso $\overline{\mathbf{P}}^{\mathbf{-1}}$ e $\mathbf{R}^{-1}\mathbf{(t)}$. Para resolver a equação (13) uma perturbação linear é adotada para aproximar o funcional **J** num típico processo iterativo. Desta forma, tem-se:

$$\alpha\mathbf{(i)} \cdot \left[\mathbf{y(t)} - \overline{\mathbf{y}}\mathbf{(t, i)}\right] \cong \hat{\mathbf{f}}_{\mathbf{w}} \left(\mathbf{x(t)}, \overline{\mathbf{w}}\mathbf{(i)}\right) \cdot \left[\mathbf{w(i)} - \overline{\mathbf{w}}\mathbf{(i)}\right] \tag{14}$$

onde,

$\mathbf{i = 1, 2, ..., I}$ são as iterações típicas do método;

$\overline{\mathbf{w}}\mathbf{(i)}$ é a estimação a priori de **w** vinda de uma iteração anterior iniciada com $\overline{\mathbf{w}}\mathbf{(1)} = \overline{\mathbf{w}}$;

$\overline{\mathbf{y}}\mathbf{(t,i)} = \hat{\mathbf{f}}\mathbf{(x(t),}\overline{\mathbf{w}}\mathbf{(i))}$ são os valores de saída da rede *feedforward;*

$\hat{\mathbf{f}}_{\mathbf{w}}\mathbf{(x(t),}\overline{\mathbf{w}}\mathbf{(i))}$ é a matriz de primeiras derivadas parciais com respeito a w;

$\mathbf{0} < \alpha\mathbf{(i)} < \mathbf{1}$ é o parâmetro de linearização a ser ajustado para garantir a hipótese de perturbação linear.

A matriz $\overline{\mathbf{P}}$ pode ser escolhida como diagonal, porque os termos da diagonal são as variâncias dos erros da informação a priori avaliadas em geral empiricamente; **R** é a matriz de covariâncias dos erros dos dados usados para treinamento (ver equação 11).

O resultado da aproximação dada pela equação (14) e aplicada sobre o funcional **J** da equação (13) é:

$$\mathbf{J(w(i))} = \frac{\mathbf{1}}{\mathbf{2}} \cdot \mathbf{[}\left(\mathbf{w(i)} - \overline{\mathbf{w}}\right)^{\mathbf{T}} \overline{\mathbf{P}}^{\mathbf{-1}} \left(\mathbf{w(i)} - \overline{\mathbf{w}}\right) + \sum_{\mathbf{t=1}}^{\mathbf{L}} \left(\mathbf{z(t,i)} - \mathbf{H(t,i)} \cdot \mathbf{w(i)}\right) \mathbf{R}^{\mathbf{-1}}\mathbf{(t)} \left(\mathbf{z(t,i)} - \mathbf{H(t,i)} \cdot \mathbf{w(i)}\right) \mathbf{]} \tag{15.a}$$

onde a seguinte notação compacta foi adotada:

$$\mathbf{z(t,i)} \equiv \alpha\mathbf{(i)} \cdot \left[\mathbf{y(t)} - \overline{\mathbf{y}}\mathbf{(t,i)}\right] + \hat{\mathbf{f}}_{\mathbf{w}}\mathbf{(x(t),}\overline{\mathbf{w}}\mathbf{(i))} \cdot \overline{\mathbf{w}}\mathbf{(i)} \tag{15.b}$$

$$\mathbf{H(t,i)} \equiv \hat{\mathbf{f}}_{\mathbf{w}}\mathbf{(x(t),}\overline{\mathbf{w}}\mathbf{(i))} \tag{15.c}$$

A solução de minimizar o funcional da equação (15.a) é formalmente equivalente (Rios Neto, 1997) ao seguinte problema de estimação linear estocástico:

$$\overline{\mathbf{w}} = \mathbf{w(i)} + \overline{\mathbf{e}} \tag{16.a}$$

$$\mathbf{z(t,i) = H(t,i) \cdot w(i) + v(t)} \tag{16.b}$$

$$\mathbf{E[\bar{e}] = 0, \qquad E[ee^T] = \bar{P}} \tag{16.c}$$

$$\mathbf{E[v(t)] = 0, \quad E[v(t)\, v^T(t)] = R(t)} \tag{16.d}$$

Uma observação importante é que se assume que os valores de $\bar{\mathbf{e}}$ e **v(t)** são não correlacionados e com distribuições Gaussianas. Observa-se que existe uma equivalência entre os métodos dos mínimos quadrados e o estocástico também para o caso de estimação não-linear. No Capítulo 2 foi verificada essa equivalência apenas para o caso de estimação linear. Para maiores detalhes sobre os estimadores lineares e não-lineares recomenda-se ver o Capítulo 2.

As equações (16.a) à (16.d) são a formulação do problema de estimação estocástica aplicada ao treinamento das redes neurais. A solução deste problema, como vista nas seções 2.2 e 2.3, é dada pela filtragem de Kalman. Entretanto, deve ser observado que existem duas possíveis soluções (Rios Neto, 1997): a versão *full* e a com processamento paralelo. Ainda com relação a estas duas soluções pode-se afirmar que em ambas pode-se aplicar o processamento em lotes ou recursivo. Desta forma, a seguir será dada rapidamente a solução *full* do problema em questão e logo em seguida o desenvolvimento da solução em paralelo. A solução em paralelo da filtragem de Kalman aplicada ao treinamento das redes *feedforward* é a parte principal deste capítulo, pois foi a ferramenta computacional de treinamento neural utilizada nesta tese.

a) Solução *Full* em lotes.

Um algoritmo de filtragem de Kalman em lotes como solução do problema de estimação de parâmetros, como visto no Capítulo 2, é dado por:

$$\mathbf{\hat{w}(i) = \bar{w} + k(i)[z(i) - H(i) \cdot \bar{w}]} \tag{17.a}$$

$$\mathbf{k(i) = \bar{P} \cdot H^T(i)\left[H(i) \cdot \bar{P} \cdot H^T(i) + R\right]^{-1}} \tag{17.b}$$

$$\overline{\mathbf{w}}(\mathbf{i}+\mathbf{1}) = \hat{\mathbf{w}}(\mathbf{i}), \qquad \alpha(\mathbf{i}) \leftarrow \alpha(\mathbf{i}+\mathbf{1}) \tag{17.c}$$

Todos os valores de **t=1, 2, ..., L** foram considerados para definir o vetor estendido **z(i)**, a matriz **H(i)** e o vetor erro **V** com matriz de covariância **R**. A solução *off line* depois das iterações **i=1,2, ..., I** é dada por:

$$\hat{\mathbf{w}} = \hat{\mathbf{w}}(\mathbf{I}), \ \mathbf{P(I)} = [\mathbf{I} - \mathbf{K(I)} \cdot \mathbf{H(I)}]\overline{\mathbf{P}} \tag{18}$$

Se α **(i)** for suficientemente pequeno e se existir redundância suficiente nos dados de treinamento, então a teoria garante que **P(I)** é uma aproximação para a matriz de covariâncias de estimação do erro, que é:

$$\mathbf{P(I)} \cong \mathbf{E}[(\mathbf{w} - \hat{\mathbf{w}})(\mathbf{w} - \hat{\mathbf{w}})^{\mathbf{T}}] \tag{19}$$

As seguintes observações para a solução *off-line* se aplicam:

1) Se a matriz **R** for diagonal, então o processamento recursivo pode ser aplicado para evitar a inversão da matriz presente na equação (17.b);

2) Para o cálculo da matriz **H(i)** pode-se utilizar a retro-propagação, mas este algoritmo não alcança o processamento paralelo embora seja um método ainda mais sofisticado do que aquele do gradiente, como descrito no Apêndice A;

3) Para evitar divergência numérica ou perda da capacidade de ter aprendizagem distribuída para todos os dados pode-se considerar (Jazwinski, 1970) técnicas adaptativas ou de fatorização.

b) Solução com Processamento Paralelo.

Os algoritmos com a estrutura de filtragem de Kalman, que coincide com aquelas dos mínimos quadrados recursivos, podem ser simplificados para produzir versões que preservem a capacidade de processamento paralelo local de redes neurais artificiais (e. g., Rios Neto, 1997). Pode-se provar que o algoritmo de filtragem de Kalman conduz

para um processamento paralelo local ainda mais geral e sofisticado do que o usual *backpropagation.*

Seja então, a seguinte equação:

$$\alpha(i)\cdot\left[y(t)-\bar{y}(t,i)\right]=\bar{f}_{w}(x(t),\overline{w}(i))\cdot\left[w(i)-\overline{w}(i)\right]+v(t) \tag{20}$$

Analisando a expressão acima, pode ser visto que na i-ésima iteração do conjunto de dados de entrada/saída pode ser localmente processado para obter-se uma estimativa do vetor de pesos $\mathbf{w_{kl}(i)}$ do l-ésimo neurônio na k-ésima camada se for considerado que:

1) para as conexões dos parâmetros de peso $\mathbf{w_a(i)}$ das camadas do sistema neural, depois de onde o processamento já está sendo feito, existem avaliados os valores estimados $\hat{\mathbf{w}}_{\mathbf{a}}(\mathbf{i})$ e o erro associado $\mathbf{e_a(i)}$ da distribuição conhecida;

2) para os parâmetros $\mathbf{w_s(i)}$, dos pesos das conexões dos outros neurônios da mesma camada, há uma estimativa a priori $\overline{\mathbf{w}}_{\mathbf{s}}(\mathbf{i})$, com erro $\bar{\mathbf{e}}_{\mathbf{s}}(\mathbf{i})$ de distribuição conhecida, que pode ser tomada como uma aproximação para $\mathbf{w_s}$;

3) para os parâmetros $\mathbf{w_e(i)}$ correspondentes às conexões dos neurônios das camadas anteriores há também uma estimativa a priori $\overline{\mathbf{w}}_{\mathbf{e}}(\mathbf{i})$ com erro $\bar{\mathbf{e}}_{\mathbf{e}}(\mathbf{i})$ da distribuição conhecida que pode ser tomada como uma aproximação para $\mathbf{w_e(i)}$.

Com estas suposições, o problema de obter uma estimativa para o vetor de pesos $\mathbf{w_{kl}(i)}$ do l-ésimo neurônio da k-ésima camada é reduzido ao problema de estimação local na i-ésima iteração e para $\mathbf{t=1,2,...,L}$:

$$\overline{\mathbf{w}}_{\mathbf{kl}}(\mathbf{i})=\mathbf{w_{kl}(i)}+\bar{\mathbf{e}}_{\mathbf{kl}} \tag{21.a}$$

$$\begin{aligned}&\alpha(i)\cdot\left[y(t)-\bar{y}(t,i)\right]-\hat{f}_{wa}(x(t),\overline{w}(i))\cdot\left[\hat{w}_a(i)-\overline{w}_a(i)\right]=\\&\hat{f}_{w_{kl}}(x(t),\overline{w}(i))\cdot\left[w_{kl}(i)-\overline{w}_{kl}(i)\right]+\hat{f}_{w_a}(x(t),\overline{w}(i))\cdot e_a(i)\\&+\hat{f}_{w_s}(x(t),\overline{w}(i))\cdot\bar{e}_s(i)+\hat{f}_{w_e}(x(t),\overline{w}(i))\cdot\bar{e}_e(i)+v(t)\end{aligned} \tag{21.b}$$

Numa notação mais compacta pode-se ter para todos os dados correspondentes a **t=1,2,...,L**:

$$\overline{\mathbf{w}}_{\mathbf{kl}}(\mathbf{i}) = \mathbf{w}_{\mathbf{kl}}(\mathbf{i}) + \overline{\mathbf{e}}_{\mathbf{kl}} \tag{22.a}$$

$$\overline{\mathbf{z}}_{\mathbf{kl}}(\mathbf{i}) = \mathbf{H}_{\mathbf{kl}}(\mathbf{i}) \cdot \mathbf{w}_{\mathbf{kl}}(\mathbf{i}) + \overline{\mathbf{v}}_{\mathbf{kl}}(\mathbf{i}) \tag{22.b}$$

Nas expressões acima $\overline{\mathbf{e}}_{\mathbf{kl}}$ é a partição correspondente de $\overline{\mathbf{e}}$ na equação (16.a) e numa i-ésima iteração este problema pode ser localmente e recursivamente resolvido com o algoritmo de filtragem de Kalman, iniciado com a matriz $\overline{\mathbf{P}}$ diagonal . Os componentes dos erros $\mathbf{e}_{\mathbf{a}}(\mathbf{i}), \overline{\mathbf{e}}_{\mathbf{s}}(\mathbf{i})$ e $\overline{\mathbf{e}}_{\mathbf{e}}(\mathbf{i})$ associados aos parâmetros de diferentes neurônios não são correlacionados (Rios Neto, 1997). Entretanto, um algoritmo com processamento paralelo local mais simples pode ser realizado através de mais duas condições de simplificação:

4) Ignorar os termos fora da diagonal da matriz de covariâncias $\mathbf{R}_{\mathbf{kl}}(\mathbf{i})$ do erro $\overline{\mathbf{V}}_{\mathbf{kl}}(\mathbf{i})$, que permite processar as componentes do vetor $\overline{\mathbf{Z}}_{\mathbf{kl}}(\mathbf{i})$ uma a uma, evitando dessa forma a necessidade de inversão da matriz. Assumir tomar esta aproximação corresponde considerar $\mathbf{e}_{\mathbf{a}}(\mathbf{i})$, $\overline{\mathbf{e}}_{\mathbf{s}}(\mathbf{i})$ e $\overline{\mathbf{e}}_{\mathbf{e}}(\mathbf{i})$ na equação (21.b) não correlacionados um com o outro;

5) Ignorar a informação sobre o nível de precisão no conhecimento anterior de $\mathbf{w}_{\mathbf{a}}(\mathbf{i}), \mathbf{w}_{\mathbf{s}}(\mathbf{i})$ e $\mathbf{w}_{\mathbf{e}}(\mathbf{i})$, e definir estes valores como se eles fossem:

$$\mathbf{w}_{\mathbf{a}}(\mathbf{i}) = \overline{\mathbf{w}}_{\mathbf{a}}(\mathbf{i}), \quad \mathbf{w}_{\mathbf{s}}(\mathbf{i}) = \overline{\mathbf{w}}_{\mathbf{s}}(\mathbf{i}), \quad \mathbf{w}_{\mathbf{e}}(\mathbf{i}) = \overline{\mathbf{w}}(\mathbf{i}) \tag{23}$$

A combinação das condições acima com a equação (22.a) resulta num problema de estimação simplificado dos dados correspondentes para **t=1, 2, ..., L**:

$$\overline{\mathbf{w}}_{\mathbf{kl}}(\mathbf{i}) = \mathbf{w}_{\mathbf{kl}}(\mathbf{i}) + \overline{\mathbf{e}}_{\mathbf{kl}} \tag{24.a}$$

$$\mathbf{z}_{\mathbf{kl}}(\mathbf{i}) = \mathbf{H}_{\mathbf{kl}}(\mathbf{i}) \cdot \mathbf{w}_{\mathbf{kl}}(\mathbf{i}) + \mathbf{v} \tag{24.b}$$

Esta última versão simplificada do filtro de Kalman é ainda mais sofisticada do que o usual algoritmo de retro-propagação e ainda pode ser demonstrada a equivalência deste algoritmo com o método de Newton (Rios Neto, 1997). A questão básica aqui é que a filtragem de Kalman pode agora ser aplicada em nível neural e não mais em nível de camada. Desta forma, o paralelismo computacional dos neurônios pode ser alcançado, embora nesta trabalho ele é implementado apenas no nível de *software*. No Apêndice B pode ser encontrado o fluxograma detalhado da filtragem de Kalman com processamento paralelo e recursivo no caso do treinamento de uma rede *feedforward*.

Há ainda questões abertas que não foram mencionadas aqui. Uma delas, muito importante, é com o fato de que pode ocorrer um mau comportamento numérico na filtragem de Kalman e divergências podem acontecer quando muitos dados são processados sequencialmente. A causa disto é que o algoritmo aprende muito bem informações erradas (Jazwinski, 1970), perdendo a capacidade de manter a aprendizagem quando os novos dados são processados. O que acontece é uma excessiva diminuição na estimativa da dispersão dos erros nas estimativas calculadas. Isto corresponde à situação de se ter a matriz de covariâncias próxima de zero.

Para evitar este mau comportamento e tentar manter a capacidade de aprendizagem uniforme é comum utilizar técnicas com fator de esquecimento ou mais efetivamente técnicas de estimação de estados adaptativas como proposto por Jazwinski (1970) e modificada por Rios Neto e Kuga (1985).

3.2.4 Técnicas Heurísticas Para Acelerar o Treinamento da Rede.

Existem muitas técnicas heurísticas para acelerar a convergência dos algoritmos de treinamento neural. O próprio algoritmo do gradiente com *momentum* é um exemplo clássico (e. g., Zurada, 1992). Aqui serão considerados métodos heurísticos para acelerar a convergência do algoritmo de treinamento que utiliza a filtragem de Kalman.

Inicialmente será discutido a normalização dos padrões de treinamento. Em geral, deve-se normalizar os padrões de treinamento de uma rede neural por dois motivos básicos: acelerar a convergência da rede durante seu treinamento e principalmente evitar os *round-off errors*, ou seja, erros numéricos devido a limitação de *hardware* que os computadores possuem ao realizarem operações matemáticas entre números muito grandes e outros muito pequenos. Seja a Matriz de Treinamento dos Padrões de Saída da rede (**MTPS**) dada por:

$$\mathbf{MTPS} = \begin{bmatrix} \mathbf{y}_{11} & \mathbf{y}_{12} & \cdots & \mathbf{y}_{1p} \\ \mathbf{y}_{21} & \mathbf{y}_{22} & \cdots & \mathbf{y}_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{y}_{n1} & \mathbf{y}_{n2} & \cdots & \mathbf{y}_{np} \end{bmatrix} = \left[\mathbf{y}_{kj}\right]_{n,p} \tag{25}$$

A normalização dos padrões de treinamento é obtida a partir da determinação do fator de normalização $\mathbf{p}^*$ dado por:

$$\mathbf{p}^* = \mathbf{max} \left| \mathbf{MTPS} \right| \tag{26}$$

O valor de $\mathbf{p}^*$ é portanto o máximo dos elementos da matriz **MTPS** em módulo. A normalização dos padrões de treinamento será então, dada por:

$$\mathbf{MTPS_n} = \frac{\mathbf{1}}{\mathbf{p}^*} \cdot \begin{bmatrix} \mathbf{y}_{11} & \mathbf{y}_{12} & \cdots & \mathbf{y}_{1p} \\ \mathbf{y}_{21} & \mathbf{y}_{22} & \cdots & \mathbf{y}_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{y}_{n1} & \mathbf{y}_{n2} & \cdots & \mathbf{y}_{np} \end{bmatrix} \tag{27}$$

onde,

$\mathbf{MTPS_n}$ ... Matriz de treinamento dos padrões de saída normalizada.

De maneira análoga pode-se normalizar a Matriz de Treinamento dos Padrões de Entrada (**MTPE**). A equação (27) só apresenta um incoveniente: quando a ordem de grandeza dos números de uma determinada linha da matriz considerada for muito maior em relação a qualquer outra linha desta mesma matriz, então a normalização obtida pelo

valor $1/p^*$ gerará números muito próximos de zero, para a linha que possuir os menores valores numéricos. Quando isso ocorrer, então dever-se-a normalizar separadamente cada linha da matriz **MTPS**, ou seja, a primeira linha possuirá o fator de normalização $1/p_1^*$, a segunda linha o fator de normalização $1/p_2^*$ e assim por diante. Onde,

$$p_i^* = \max [\,|y_{i1}|\ |y_{i2}| \ldots |y_{ip}|\,]_{1\,x\,p} \tag{28}$$

Voltando agora a questão do algoritmo da filtragem de Kalman, as principais heurísticas que podem ser aplicadas a esse método de treinamento são três: correção da informação a priori das matrizes de covariância $\overline{\mathbf{P}}$, correção do valor de α e a alteração dos valores da matriz de ruído **R**. Para explicar a primeira desta heurísticas considere, por exemplo a Figura 3.7. O neurônio *perceptron* possui **n** pesos em sua entrada. No método de treinamento neural empregando o filtro de Kalman com processamento paralelo, ou seja, em nível neural, existe também uma matriz a priori das covariâncias destes pesos. Como se sabe, a informação a priori desta matriz é, em geral, dada pela matriz identidade de dimensão **nxn** quando se inicia o treinamento. Entretanto, com o decorrer das iterações do método de treinamento pode acontecer que os valores dos pesos cresçam muito e desenvolvam uma dispersão em seus valores muito maior que a unidade.

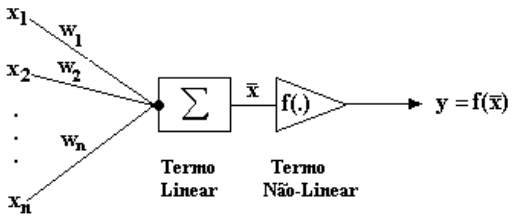


FIGURA 3.7 – Neurônio *perceptron*.

Neste caso, seria conveniente aumentar os valores da diagonal principal da matriz de covariâncias a priori. Uma heurística bastante utilizada é a seguinte:

$$\overline{p}_{ii} = 1 + \beta \cdot \mathbf{w_i} \qquad \text{para } \mathbf{i=1, 2, ..., n} \tag{28}$$

onde,

$\mathbf{0} < \beta << \mathbf{1}$ ... porcentagem associada ao valor do peso $\mathbf{w_i}$;

$\overline{p}_{ii}$ ... valor genérico da diagonal principal da matriz de covariâncias a priori associada aos pesos de entrada do neurônio *perceptron*.

Outras heurísticas bastante utilizadas são a alteração do valor de α e a diminuição do valor dos termos da diagonal da matriz **R**, $\mathbf{r_{ii}}$, com o decorrer das iterações. Neste último caso, começa-se com um valor alto para o valor de $\mathbf{r_{ii}}$ e com o processamento sucessivo das iterações diminui-se esporadicamente e empiricamente os valores de $\mathbf{r_{ii}}$. Deve ser observado que esta técnica depende muito de cada tipo de problema, e em geral, a porcentagem de diminuição deste valor é pelo método da tentativa e erro.

CAPÍTULO 4

ESTRUTURAS DE CONTROLE NEURAL

4.1 Introdução.

Neste capítulo, com o objetivo de contextualizar a metodologia adotada na execução deste trabalho, serão abordadas as principais estruturas de controle neural. Serão analisadas a possibilidades de se utilizar as redes neurais diretamente em estratégias de controle não-linear. Neste contexto o conhecimento da dinâmica de uma planta poderá ser armazenado implicitamente dentro de uma rede neural do tipo *feedforward*. A habilidade das redes para aproximar mapeamentos não-lineares é a propriedade fundamental para sua utilização em controle. Da mesma maneira que funções de transferências provêm um esquema genérico para representar modelos lineares, as redes neurais artificias provêm a representação e generalização de modelos não-lineares.

Neste capítulo inicialmente serão vistas as representações dos modelos diretos e inversos da planta dinâmica e em seguida as principais estruturas de controle neural. Apesar de serem detalhadas as principais estruturas de controle a ênfase será dada à estrutura de controle preditivo, pois esta foi a estrutura utilizada na abordagem desenvolvida neste trabalho.

4.2 Identificação de Sistemas Dinâmicos Diretos e Inversos.

É importante identificar os modelos direto e inverso da planta do sistema físico a ser estudado. Estes modelos, como será visto mais tarde, são o ponto chave de partida para se utilizar qualquer uma das possíveis estruturas de controle.

a) Modelo direto

O procedimento de treinar uma rede neural para representar a dinâmica de um sistema será referido aqui como modelação direta. Uma estrutura para alcançar isso é mostrada na Figura 4.1. Como pode ser visto, o modelo da rede neural é colocado em paralelo

com o sistema e o erro entre o sistema e as saídas da rede é utilizado como o sinal de treinamento da rede. Esta estrutura de aprendizagem é um clássico problema de aprendizado onde a necessidade de um professor (padrões de treinamento) é necessária.

Uma questão no contexto de controle bastante importante é a natureza dinâmica dos sistemas a serem estudados. É assumido que o sistema é governado (Norgaard et al, 2000) pela seguinte equação não-linear de diferenças discretas no tempo.

$$\mathbf{y^p(t+1) = f[y^p(t), ..., y^p(t-n+1); u(t), ..., u(t-m+1)]} \quad (1)$$

Desta forma, a saída do sistema $\mathbf{y^p}$ no tempo **t+1** depende dos **n** valores passados e dos **m** valores de entrada **u.** Uma abordagem óbvia para o modelamento do sistema é escolher as estruturas de entrada/saída da rede neural como sendo as mesmas daquela do sistema. Denotando a saída da rede como $\mathbf{y^m}$ tem-se então

$$\mathbf{y^m(t+1) = \hat{f}[y^p(t), ..., y^p(t-n+1); u(t), ..., u(t-n+1), w]} \quad (2)$$

Aqui, $\hat{\mathbf{f}}$ representa o mapeamento não-linear de entrada/saída da rede. Note que a entrada para a rede inclui os "valores passados" do sistema de saída real. Esta dependência sobre o sistema de saída não está incluída no esquema da Figura 4.1 por questões de simplicidade.

Se for assumido que após um aceitável período de treinamento a rede fornece uma boa representação da planta (isto é $\mathbf{y^m \cong y^p}$) então, a rede poderá ser utilizada independentemente da planta. Tal modelo neural poderá ser descrito por:

$$\mathbf{y^m(t+1) = \hat{f}[y^m(t), ..., y^m(t-n+1); u(t); ..., u(t-m+1), w]} \quad (3)$$

O modelo direto representa a dinâmica do sistema físico por meio de uma rede neural, no caso de redes *feedfoorward*, projetada como um modelo com entradas atrasadas discretizadas..

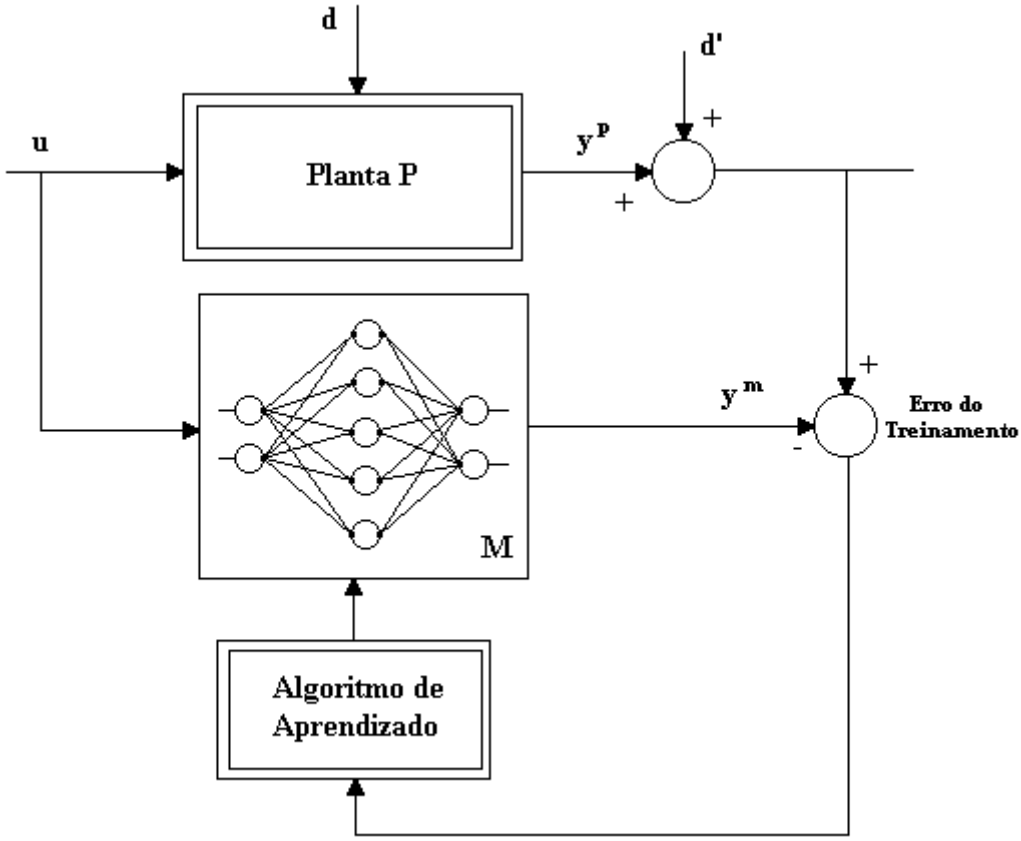


FIGURA 4.1 – Identificação de sistemas dinâmicos através de redes neurais.

Desta forma, a partir de um conjunto particular de estados do sistema dinâmico original que servirão para montar os padrões de treinamento, a rede neural será capaz de aprender todas as características do sistema físico dentro de um domínio de interesse.

b) Modelo Inverso

O modelo inverso da planta é exatamente o contrário do modelo direto como ilustra a Figura 4.2. Neste caso, dados os valores atrasados e atuais dos estados e para onde se deseja ir, a saída da rede fornecerá o controle **u(t)** que tornará viável esta trajetória de controle. É importante observar que existem dois métodos básicos para se treinar o modelo inverso da planta: a *aprendizagem especializada* e a *aprendizagem generalizada* (Hunt et al, 1992).

A desvantagem da *aprendizagem especializada* em relação a *aprendizagem generalizada* é que a primeira exige a presença do modelo direto (já treinado anteriormente) para realizar o treinamento do modelo inverso (ver Figura 4.2) enquanto que a segunda não. Na *aprendizagem especializada* a retro-propagação dos pesos deve ser passada do modelo direto para o indireto e somente os pesos deste último devem ser atualizados durante o treinamento do modelo inverso.

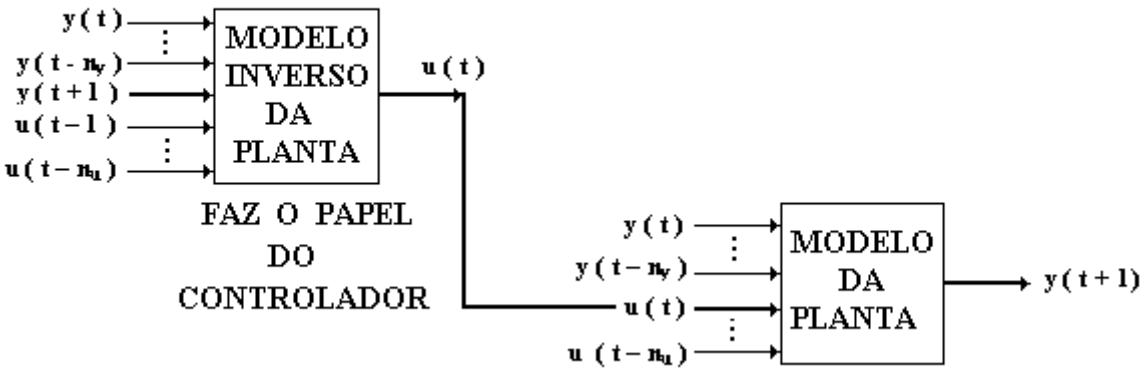


FIGURA 4.2 - Modelos direto e inverso da planta representados através de duas rede neurais do tipo *feedforward.*

As características básicas do *aprendizado generalizado* do modelo inverso são:

1) O aprendizado não pode ser conduzido apenas numa região de interesse do domínio dos controles **u`s**, mas tem de ser conduzido extensa e sistematicamente em todo esse domínio;

2) Em geral ela exige mais padrões de treinamento que a *aprendizagem especializada*;

3) Se o mapeamento não linear do sistema não for *um para um* então uma relação inversa incorreta pode ser obtida.

As características básicas do *aprendizado especializado* do modelo inverso são:

1) O controlador é ensinado já com os sinais especializados correspondentes às saídas desejadas $\mathbf{y}^{\mathbf{d}}$, diminuindo consideravelmente o esforço de treinamento;

2) O modelo neural do sistema é utilizado para facilitar o cálculo de $\partial \mathbf{Y^p} / \partial \mathbf{u}$ que é aproximado por $\partial \mathbf{Y^m} / \partial \mathbf{u}$; se o modelo estiver bem identificado, pode-se realimentar y^m diretamente ao invés de y^p.

3) Se o mapeamento não for *um para um*, uma inversa particular será determinada.

Matematicamente o que se pretende com esta rede é que:

$$\hat{\mathbf{u}}(t) = \hat{\mathbf{f}}^{-1}(\mathbf{y^d}(t+1), \mathbf{y^p}(t), ..., \mathbf{y^p}(t - \mathbf{n_y}); \mathbf{u}(t-1), ..., \mathbf{u}(t - \mathbf{n_u}), \mathbf{w}) \quad (4)$$

Em geral o modelo inverso da planta, após ser cuidadosamente treinado e testado, funcionará como o *controlador* em uma estrutura de controle neural.

4.3 Controle Neural: Fundamentos.

As propriedades de aprendizado de funções não-lineares são a condição central para a utilização de redes neurais em controle. Treinar uma rede neural utilizando dados de entrada/saída de uma planta não-linear pode ser considerado como um problema de aproximação de uma função não-linear.

As redes *feedforward* multicamadas do tipo *perceptron* podem aproximar arbitrariamente bem uma função contínua (e.g., Cybenko, 1988). Uma rede *feedforward* com somente uma simples camada interna, onde cada unidade da camada interna é representada por uma função sigmoidal continua, é suficiente para representar qualquer sistema dinâmico.

Modelos de sistemas dinâmicos e suas inversas têm utilidade imediata em controle. Na literatura de controle um número grande de trabalhos bem estabelecidos já existe. Na literatura sobre as arquiteturas das redes neurais para controle, um grande número de estruturas de controle têm sido propostas e utilizadas.

As principais características das redes neurais na implementação em controle segundo Hunt e Sbarbaro (1991) e Hunt et al (1992) são:

1) a habilidade de representar relações arbitrárias com características não-lineares;

2) adaptação e aprendizagem de sistemas incertos através de atualização dos parâmetros ou pesos da rede em tempo *off-line* ou *on-line*;

3) a transformação da informação do sinal, tanto no nível qualitativo quanto no nível quantitativo, é transformada permitindo a fusão dos dados;

4) a habilidade de processamento paralelo na arquitetura das redes neurais, que permite um processamento mais rápido para uma grande escala de sistemas dinâmicos;

5) um elevado grau de robustez nas arquiteturas das redes permitindo a tolerância de erros e evitando a degradação do sinal;

6) a capacidade de se utilizar estruturas de aprendizado para representar a dinâmica direta e inversa de sistemas dinâmicos não-lineares.

Entre as estruturas de controle mais relacionadas na literatura internacional (Hunt et al, 1992; Varoto, 1997) estão: controle supervisionado, controle inverso direto, controle por modelo de referência, estrutura de controle *Internal Model Control* (IMC), controle adaptativo e controle preditivo. A seguir serão examinadas apenas algumas destas estruturas, dando-se ênfase, a estrutura de controle preditivo.

4.3.1 Estrutura de Controle IMC.

Já treinados os modelos direto e inverso da planta então, respectivamente, servirão como modelo **M** e controlador **C** de uma estrutura de controle IMC como ilustra a Figura 4.3. É demonstrado que a estrutura IMC pode ser utilizada em sistemas não-lineares (Economou et al, 1986). O controlador IMC é uma estrutura de controle

conveniente que incorpora diretamente o modelo da planta do sistema físico e sua inversa correspondente.

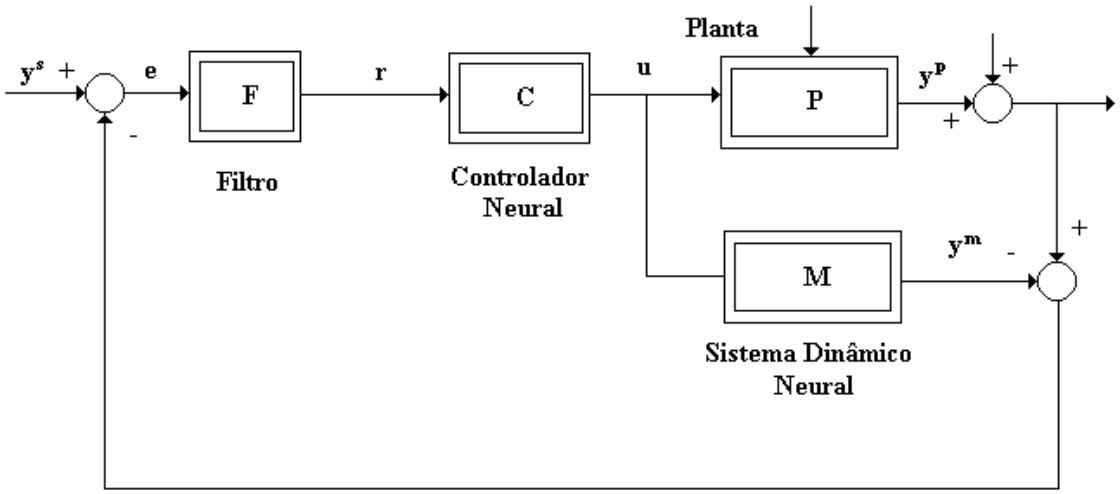


FIGURA 4.3 – Estrutura de Controle IMC.

As características relevantes ao projeto de um sistema de controle com estrutura IMC são:

1) Análise da estabilidade do sistema IMC;

2) Análise das condições de inversabilidade de sistemas dinâmicos não-lineares;

3) Algoritmos para construir os modelos direto e inverso da planta.

Partindo-se dos diagramas de bloco de uma estrutura IMC, pode-se demonstrar três propriedades interessantes (Economou et al, 1986):

1) Se a planta **P** e o controlador **C** são estáveis nas entradas/saídas e o modelo **M** é uma perfeita representação da planta, então o sistema em *feedback* da estrutura IMC é estável na entrada $\mathbf{y_s}$ e saída ($\mathbf{y_p}$**-d'**);

2) Se a inversa de **M** ($\mathbf{M^{-1}}$) existe e o sistema em *feedback* com estrutura IMC é estável na entrada e na saída e com o controlador dado por $\mathbf{C=M^{-1}}$, então o controle será perfeito, isto é**,** $\mathbf{y=y_s}$.

3) Assumindo que a inversa das variáveis de estado do modelo exista, que as variáveis de estado do controlador seja igual a essa inversa e que o sistema em *feedback* seja estável na entrada e na saída com este controlador, então o *offset* do controle é alcançado assintoticamente pelas constantes de entrada.

A presença do filtro na estrutura IMC possui duas finalidades: aliviar os problemas de sensibilidade sobre as incertezas do modelo e projetar o sinal **e** em um espaço de entrada apropriado para o controlador. Pode acontecer também da rede do modelo inverso funcionar como um filtro, e desta forma, este elemento poderá eventualmente ser omitido da malha de controle.

4.3.2 Controle Preditivo.

Os algoritmos de filtragem de Kalman com ou sem processamento paralelo podem ser utilizados não só para treinar as redes *feedforward* que possuem a capacidade inerente de representar a dinâmica de sistemas físicos, mas também para determinar a política de controle suave numa estrutura de controle preditivo em malha fechada. Nesta seção, adotando-se uma abordagem heurística e teórica, apresentar-se-á o projeto e análise de convergência de um método de controle preditivo neural.

Este método consiste em resolver um problema de otimização de um índice quadrático de desempenho, cujo vínculo é a rede já treinada com a dinâmica do sistema desejado (Mills et al, 1994). Por se tratar de um problema de otimização não-linear a solução é obtida iterativamente para as ações discretas de controle através de sucessivas linearizações. Para o método se tornar viável a estimação dos controles deverá ser obtida necessariamente em tempo real. Com a evolução dos processadores atuais isto não constitui nenhum problema que impeça sua utilização.

Ao contrário de uma estrutura de controle do tipo IMC, num esquema de controle preditivo não é necessário empregar a dinâmica do *modelo inverso* da planta, e desta forma, evita-se um treinamento neural. Por outro lado, no controle preditivo neural

deve-se a todo instante resolver um problema de otimização envolvendo o modelo neural da dinâmica da planta.

A Figura 4.4 apresenta um esquema simplificado da estrutura de controle preditivo. Como pode ser visto, uma rede neural é colocada em paralelo com a planta na tentativa de aprendê-la. Quando o aprendizado alcançado estiver dentro de um erro ou tolerância aceitáveis, então a determinação dos controles suaves poderá ser obtida pelo algoritmo da filtragem de Kalman como a solução de um problema de otimização vinculado a rede e que rastreará também a trajetória de referência **r(t)**. Pode ser demonstrado (Rios Neto, 2000) que os algoritmos de filtragem de Kalman fornecem soluções que convergem para soluções suaves para as variáveis de controle e que rastreiam a trajetória de referência.

No esquema da Figura 4.4 os problemas associados ao treinamento da rede neural *feedforward* e da determinação da política de controle suave são ambos vistos e tratados de uma maneira integrada como problemas de estimação linear estocástica de parâmetros. O tipo de abordagem elaborada aqui permite ver o problema de controle ótimo em uma estrutura estocástica mais geral e derivar versões de algoritmos de controle com processamento paralelo ou não (Rios Neto, 2000) que são formalmente equivalentes às versões do filtro de Kalman derivadas e utilizadas para o problema de treinamento da rede neural *feedforward* (Rios Neto, 1997).

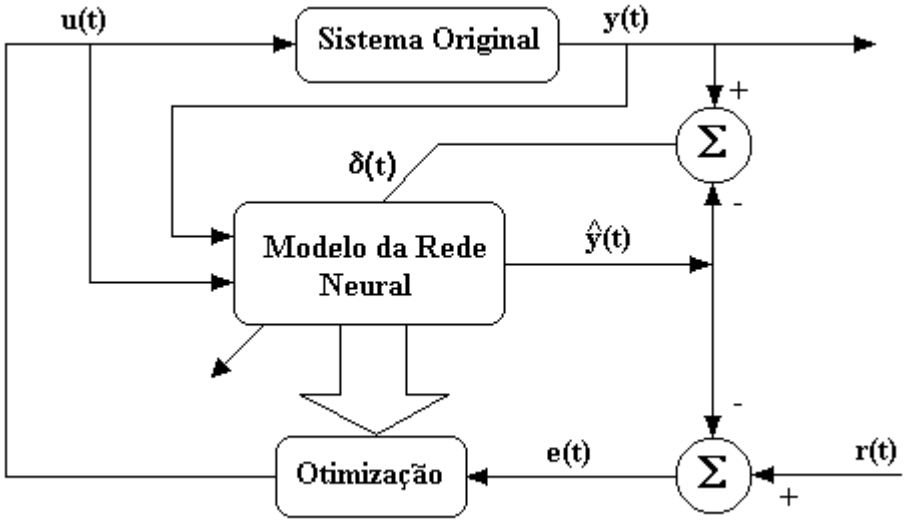


FIGURA 4.4 – Esquema de otimização neural para a determinação da política de controle suave u(t) que rastreará a trajetória de referência r(t).

O problema que se deseja resolver é o de *controlar* o sistema dinâmico dado por,

$$\dot{\mathbf{x}} = \mathbf{f(x,u)} \tag{5}$$

onde um modelo de entrada/saída não-linear discretizado no tempo é utilizado para predizer respostas aproximadas do sistema dado em (5):

$$\hat{\mathbf{y}}(\mathbf{t_j}) = \mathbf{f}(\mathbf{y}(\mathbf{t_{j-1}}), ..., \mathbf{y}(\mathbf{t_{j-n_y}}); \mathbf{u}(\mathbf{t_{j-1}}), ..., \mathbf{u}(\mathbf{t_{j-n_u}}), \mathbf{w}) \tag{6.a}$$

onde,

$$\mathbf{t_j} = \mathbf{t} + \mathbf{j} \cdot \Delta\mathbf{t} \tag{6.b}$$

O esquema de controle preditivo neural utiliza-se de uma rede *feedforward* que possui a habilidade de aprender com precisão desejada um mapeamento como aquele representado por (6.a) para modelar o sistema dinâmico da equação (5). O modelo interno representado pela rede será então o modelo de resposta que poderá ser utilizado para determinar as ações de controle suaves que rastrearão a trajetória de referência por minimizar um índice de performance preditivo quadrático (Hunt et al, 1992) ou

(Norgaard et al, 2000) em uma estrutura de controle preditivo. O índice de performance ou funcional desta estrutura de controle é dado por:

$$\mathbf{J} = [\sum_{j=1}^{n} [\mathbf{y_r(t_j)} - \hat{\mathbf{y}}(\mathbf{t_j})]^{\mathbf{T}} \cdot \mathbf{r_y^{-1}(t)} \cdot [\mathbf{y_r(t_j)} - \hat{\mathbf{y}}(\mathbf{t_j})] + \sum_{j=0}^{n-1} [\mathbf{u(t_j)} - \mathbf{u(t_{j-1})}]^{\mathbf{T}} \cdot \mathbf{r_u^{-1}(t)} \cdot [\mathbf{u(t_j)} - \mathbf{u(t_{j-1})}]]/2 \quad (7)$$

onde,

$\mathbf{y_r(t_j)}$ ... trajetória de referência no instante $\mathbf{t_j}$;

n ... horizonte em que as ações de controle e trajetórias de referência são consideradas;

$\mathbf{r_y(t_j)}$ e $\mathbf{r_u(t_j)}$ ... matrizes de pesos definidas positivas;

$\hat{\mathbf{y}}(\mathbf{t_j})$ ... saída da rede *feedforward* treinada para aproximar o modelo do sistema dinâmico.

A saída da rede *feedforward* $\hat{\mathbf{y}}(\mathbf{t_j})$ é representada pela expressão (6.a). Os parâmetros ou pesos **w** desta rede já devem ter passado por um treinamento que produza uma saída na rede com um erro dentro de uma tolerância aceitável. O primeiro termo do funcional da equação (7) está associado ao rastreio da trajetória de referência e o segundo termo na determinação de uma política de controle suave. Quando este funcional é minimizado espera-se que estas duas condições sejam satisfeitas simultaneamente. Para maiores detalhes sobre a construção matemática do funcional dado pela equação (7) ver (Clarke et al, 1987a e 1987b).

Em uma estrutura de controle preditivo basicamente visa-se rastrear continuamente uma trajetória de referência e, portanto trabalha-se em *malha fechada*. Por outro lado, a trajetória de referência pode ser determinada, por exemplo, por técnicas numéricas de controle ótimo numérico visando-se encontrar trajetórias de mínimo tempo ou mínimo

combustível. Deste modo, a determinação da trajetória de referência, em geral, se dá em *malha aberta*.

O problema de determinação das ações de controle preditivo pode ser tratado também como uma estimação linear ótima de parâmetros permitindo, assim, a derivação e utilização de um algoritmo do tipo Filtro de Kalman. Este método estocástico assume inicialmente que o problema de determinação do controle sobre o funcional da equação (7) pode ser visto como o seguinte problema de estimação de parâmetros estocásticos (Rios Neto, 2000):

$$\mathbf{y_r(t_j) = \hat{f}(\hat{y}(t_{j-1}), ..., \hat{y}(t_{j-n_y}); u(t_{j-1}), ..., u(t_{j-n_u}), \hat{w}) + v_y(t_j)} \tag{8.a}$$

$$\mathbf{0 = u(t_{j-1}) - u(t_{j-2}) + v_u(t_{j-1})} \tag{8.b}$$

$$\mathbf{E[v_y(t_j)] = 0;\ E[v_y(t_j) \cdot v_y^T(t_j)] = r_y(t_j)} \tag{8.c}$$

$$\mathbf{E[v_u(t_j)] = 0;\ E[v_u(t_j) \cdot v_u^T(t_j)] = r_u(t_j)} \tag{8.d}$$

para,

$$\mathbf{j = 1, 2, ..., n} \tag{8.e}$$

onde,

$\mathbf{\hat{y}(t_j) = f(y(t_{j-1}), ..., y(t_{j-n_y}); u(t_{j-1}), ..., u(t_{j-n_u}), w)}$ é a saída da rede neural;

$\mathbf{\hat{y}(t), ..., \hat{y}(t_{1-n_y})}$ e $\mathbf{u(t_{-1}), ..., u(t_{1-n_u})}$ são, respectivamente, as respostas do sistema e as ações de controle já ocorridas e conhecidas;

$\mathbf{v_y(t_j)}$ e $\mathbf{v_u(t_j)}$ são as componentes não-correlacionadas de ruído para diferentes valores de $\mathbf{t_j}$.

A equação (8.a) estabelece que a trajetória de referência do estado do sistema $\mathbf{y_r(t_j)}$ no instante futuro é igual a estimação realizada pela rede em relação aos instantes atrasados mais uma incerteza $\mathbf{v_y(t_j)}$. A equação (8.b) traduz a característica suave das ações de controle, ou seja, duas atuações sucessivas devem ser estimadas de tal forma que a diferença entre elas esteja o mais próximo possível da média nula.

A equação (8.b) é uma relação recursiva que pode ser expressa na forma de uma informação a priori como segue:

$$\hat{\mathbf{u}}(\mathbf{t_{-1}}) = \mathbf{u}(\mathbf{t_{j-1}}) + \sum_{\mathbf{k=0}}^{\mathbf{j-1}} \mathbf{v_u}(\mathbf{t_k}) \tag{9}$$

onde o valor a priori $\hat{\mathbf{u}}(\mathbf{t_{-1}})$ é o valor estimado de um controle no instante $\mathbf{t_{-1}} = \mathbf{t} - \Delta\mathbf{t}$.

Desta forma, a minimização do funcional dado pela equação (7) é modelada como satisfazendo as observações da equação (8.a) sujeita a informação a priori da equação (9). A primeira conseqüência do tratamento deste problema numa estrutura estocástica mais geral é que as matrizes de pesos presentes na função objetivo da equação (7), ou seja, $\mathbf{r_y^{-1}(t_j)}$ e $\mathbf{r_u^{-1}(t_j)}$ têm agora o significado de matrizes de covariância associadas as variáveis aleatórias cujos os desvios padrões modelam, respectivamente, a precisão no rastreio da trajetória de referência e a dispersão do incremento do controle suave. Isto facilita bastante o entendimento de suas definições.

Para tratar reiterativamente o problema representado pelas equações (8.a) e (8.b) como um de estimação linear de parâmetros é necessário tomar uma aproximação linearizada da equação (8.a) como segue:

$$\begin{aligned} &\boldsymbol{\alpha}(\mathbf{i}) \cdot [\mathbf{y_r(t_j)} - \bar{\mathbf{y}}(\mathbf{t_j}, \mathbf{i})] = \\ &\sum_{\mathbf{k=0}}^{\mathbf{j-1}} [\partial \hat{\mathbf{y}}(\mathbf{t_j}) / \partial \mathbf{u}(\mathbf{t_k})]_{\{\bar{\mathbf{u}}(\mathbf{t_k};\mathbf{i})\}} \cdot [\mathbf{u}(\mathbf{t_k}, \mathbf{i}) - \bar{\mathbf{u}}(\mathbf{t_k}, \mathbf{i})] + \mathbf{v_y(t_j)} \end{aligned} \tag{10}$$

onde,

$\mathbf{0 < \alpha(i) \leq 1}$ é uma constante que deve ser ajustada para garantir a hipótese de aproximação da perturbação linear.

As derivadas parciais que aparecem na equação (10) são calculadas utilizando-se a regra da retro-propagação em relação *às saídas dos neurônios* da rede *feedforward*. Tem-se também que:

$$\alpha(\mathbf{i}) \cdot [\hat{\mathbf{u}}(\mathbf{t_{-1}}) - \overline{\mathbf{u}}(\mathbf{t_l}, \mathbf{i}))] = [\mathbf{u}(\mathbf{t_l}, \mathbf{i}) - \overline{\mathbf{u}}(\mathbf{t_l}, \mathbf{i})] + \sum_{\mathbf{k=0}}^{\mathbf{l}} \mathbf{v_u}(\mathbf{t_k}) \tag{11.a}$$

para,

$$\mathbf{l = 0, 1, ..., n - 1} \text{ e } \mathbf{i = 1, 2, ..., I} \tag{11.b}$$

onde,

$\hat{\mathbf{u}}(\mathbf{t_{-1}})$ ... é a solução estimada do último passo de controle;

$\alpha(\mathbf{i}) \leftarrow \alpha(\mathbf{i+1})$;

$\overline{\mathbf{u}}(\mathbf{t, i+1}) = \hat{\mathbf{u}}(\mathbf{t_l, i})$ ... é o valor estimado aproximado de $\mathbf{u}(\mathbf{t_l})$ na i-ésima iteração e para **i=1** são utilizados os valores estimados ou extrapolados do último passo de controle.

Para $\mathbf{j = 1, 2, ..., n}$ e $\mathbf{l = 0, 1, ..., n - 1}$ o problema das equações (10) e (11.a) é o de estimação linear estocástica de parâmetros. Este problema pode ser representado numa notação mais compacta e de mais fácil entendimento. Seja então as seguintes definições:

$$\mathbf{U_l(t, i) \equiv u(t_l, i)} \tag{12.a}$$

$$\hat{\mathbf{U}}_\mathbf{l}(\mathbf{t_{-1}}) \equiv \hat{\mathbf{u}}(\mathbf{t_{-1}}) \tag{12.b}$$

Assim, o problema pode ser equivalentemente expresso como:

$$\alpha(\mathbf{i}) \cdot [\hat{\mathbf{U}}(\mathbf{t_{-1}}) - \overline{\mathbf{U}}(\mathbf{t, i})] = \mathbf{U(t, i)} - \overline{\mathbf{U}}(\mathbf{t, i}) + \mathbf{V_u(t)} \tag{13.a}$$

$$\alpha(\mathbf{i}) \cdot \overline{\mathbf{Z}}^{\mathbf{u}}(\mathbf{t},\mathbf{i}) = \mathbf{H}^{\mathbf{u}}(\mathbf{t},\mathbf{i}) \cdot [\mathbf{U}(\mathbf{t},\mathbf{i}) - \overline{\mathbf{U}}(\mathbf{t},\mathbf{i})] + \mathbf{V}_{\mathbf{y}}(\mathbf{t}) \tag{13.b}$$

O significado das variáveis compactas nas equações acima é obtido por comparação direta das equações (13.a) e (13.b) com as equações (10) e (11.a), respectivamente. A seguir é apresentada uma solução heurística e sem demonstração das expressões do filtro de Kalman como solução típica de uma i-ésima iteração deste problema:

$$\begin{aligned}\hat{\mathbf{U}}(\mathbf{t},\mathbf{i}) = \overline{\mathbf{U}}(\mathbf{t},\mathbf{i}) + \alpha(\mathbf{i}) \cdot [\hat{\mathbf{U}}(\mathbf{t}_{-1}) - \overline{\mathbf{U}}(\mathbf{t},\mathbf{i})] + \\ \mathbf{k}(\mathbf{t},\mathbf{i}) \cdot \alpha(\mathbf{i}) \cdot [\overline{\mathbf{Z}}^{\mathbf{u}}(\mathbf{t},\mathbf{i}) - \mathbf{H}^{\mathbf{u}}(\mathbf{t},\mathbf{i}) \cdot [\hat{\mathbf{U}}(\mathbf{t}_{-1}) - \overline{\mathbf{U}}(\mathbf{t},\mathbf{i})]]\end{aligned} \tag{14.a}$$

$$\begin{aligned}\mathbf{k}(\mathbf{t},\mathbf{i}) = \mathbf{R}_{\mathbf{u}}(\mathbf{t}) \cdot \mathbf{H}^{\mathbf{u}^{\mathbf{T}}}(\mathbf{t},\mathbf{i}) \cdot [\mathbf{H}^{\mathbf{u}}(\mathbf{t},\mathbf{i}) \cdot \mathbf{R}_{\mathbf{u}}(\mathbf{t}) \cdot \mathbf{H}^{\mathbf{u}^{\mathbf{T}}}(\mathbf{t},\mathbf{i}) + \mathbf{R}_{\mathbf{y}}(\mathbf{t})]^{-1} \equiv \\ [\mathbf{R}_{\mathbf{u}}^{-1}(\mathbf{t}) + \mathbf{H}^{\mathbf{u}^{\mathbf{T}}}(\mathbf{t},\mathbf{i}) \cdot \mathbf{R}_{\mathbf{y}}^{-1}(\mathbf{t}) \cdot \mathbf{H}^{\mathbf{u}}(\mathbf{t},\mathbf{i})]^{-1} \cdot \mathbf{H}^{\mathbf{u}^{\mathbf{T}}}(\mathbf{t},\mathbf{i}) \cdot \mathbf{R}_{\mathbf{y}}^{-1}(\mathbf{t})\end{aligned} \tag{14.b}$$

$$\overline{\mathbf{U}}(\mathbf{t},\mathbf{i}+\mathbf{1}) = \hat{\mathbf{U}}(\mathbf{t},\mathbf{i}) \,;\; \alpha(\mathbf{i}) \leftarrow \alpha(\mathbf{i}+\mathbf{1}) \tag{14.c}$$

$$\hat{\mathbf{U}}(\mathbf{t}) = \hat{\mathbf{U}}(\mathbf{t},\mathbf{I}) \,;\; \hat{\mathbf{R}}_{\mathbf{u}}(\mathbf{t},\mathbf{I}) = [\mathbf{I}_{\mathbf{u}} - \mathbf{K}(\mathbf{t},\mathbf{I}) \cdot \mathbf{H}^{\mathbf{u}}(\mathbf{t},\mathbf{I})] \cdot \mathbf{R}_{\mathbf{u}}(\mathbf{t}) \tag{14.d}$$

para,

$$\mathbf{i} = \mathbf{1}, \mathbf{2}, ..., \mathbf{I} \tag{14.e}$$

onde,

$\mathbf{R}_{\mathbf{u}}(\mathbf{t})$, $\mathbf{R}_{\mathbf{y}}(\mathbf{t})$ e $\hat{\mathbf{R}}(\mathbf{t},\mathbf{I})$... são, respectivamente, as matrizes de covariância de $\mathbf{V}_{\mathbf{u}}(\mathbf{t})$, $\mathbf{V}_{\mathbf{y}}(\mathbf{t})$ e $(\hat{\mathbf{U}}(\mathbf{t},\mathbf{I}) - \mathbf{U}(\mathbf{t}))$;

$\mathbf{I}_{\mathbf{u}}$ ... matriz identidade.

O controle calculado com este algoritmo é o mínimo do funcional:

$$
\begin{aligned}
\mathbf{J}(\alpha,\mathbf{i}) = & [[\alpha(\mathbf{i})\cdot\overline{\mathbf{Z}}^{\mathbf{u}}(\mathbf{t},\mathbf{i}) - \mathbf{H}^{\mathbf{u}}(\mathbf{t},\mathbf{i})\cdot[\mathbf{U}(\mathbf{t},\mathbf{i}) - \overline{\mathbf{U}}(\mathbf{t},\mathbf{i})]]^{\mathbf{T}}\cdot\mathbf{R}_{\mathbf{y}}^{-\mathbf{1}}(\mathbf{t})\cdot \\
& [\alpha(\mathbf{i})\cdot\overline{\mathbf{Z}}^{\mathbf{u}}(\mathbf{t},\mathbf{i}) - \mathbf{H}^{\mathbf{u}}(\mathbf{t},\mathbf{i})\cdot[\mathbf{U}(\mathbf{t},\mathbf{i}) - \overline{\mathbf{U}}(\mathbf{t},\mathbf{i})]] + \\
& [\mathbf{U}(\mathbf{t},\mathbf{i}) - \overline{\mathbf{U}}(\mathbf{t},\mathbf{i}) - \alpha(\mathbf{i})\cdot[\hat{\mathbf{U}}(\mathbf{t}_{-\mathbf{1}}) - \overline{\mathbf{U}}(\mathbf{t},\mathbf{i})]]^{\mathbf{T}}\cdot\mathbf{R}_{\mathbf{u}}^{-\mathbf{1}}(\mathbf{t})\cdot \\
& [\mathbf{U}(\mathbf{t},\mathbf{i}) - \overline{\mathbf{U}}(\mathbf{t},\mathbf{i}) - \alpha(\mathbf{i})\cdot[\hat{\mathbf{U}}(\mathbf{t}_{-\mathbf{1}}) - \overline{\mathbf{U}}(\mathbf{t},\mathbf{i})]]
\end{aligned}
\tag{15}
$$

Desta forma, a convergência para um controle suave $\hat{\mathbf{U}}(\mathbf{t})$ que rastreará a trajetória de referência $\mathbf{y_r(t)}$ é garantida uma vez que a rede neural *feedforward* $\hat{\mathbf{y}}(\mathbf{t})$ tem a capacidade de representar a dinâmica do sistema da equação (5) e de permitir uma aproximação linearizada (Chen and Billings, 1992) em uma i-ésima típica iteração, quando se emprega um valor suficientemente pequeno para $\alpha(\mathbf{i})$.

Por fim, para a determinação da retro-propagação, utilizada para determinar a matriz Jacobiana **H(t,i)** para o caso particular de um horizonte de previsão **n=1**, pode-se formular o seguinte conjunto de equações *iterativas* (ver Apêndice C) para determinar as derivadas das saídas da rede em relação às entradas dos neurônios das camadas anteriores:

$$
\frac{\partial \mathbf{y}^{\mathbf{l}}}{\partial \overline{\mathbf{y}}^{\mathbf{k}}} = \frac{\partial \mathbf{y}^{\mathbf{l}}}{\partial \overline{\mathbf{y}}^{\mathbf{k+1}}}\cdot\mathbf{W}^{\mathbf{k+1}}\cdot\mathbf{I}_{\mathbf{f}'(\overline{\mathbf{y}}^{\mathbf{k}})} \qquad \text{para } \mathbf{k=l-1,\ l-2} \tag{16.a}
$$

onde,

$$
\frac{\partial \mathbf{y}^{\mathbf{l}}}{\partial \overline{\mathbf{y}}^{\mathbf{l}}} = \mathbf{I}_{\mathbf{f}'(\overline{\mathbf{y}}^{\mathbf{l}})} \tag{16.b}
$$

$$
\mathbf{I}_{\mathbf{f}'(\overline{\mathbf{y}}^{\mathbf{l}})} = \begin{bmatrix}
\mathbf{f}'(\overline{\mathbf{y}}_{\mathbf{1}}^{\mathbf{l}}) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{f}'(\overline{\mathbf{y}}_{\mathbf{2}}^{\mathbf{l}}) & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{f}'(\overline{\mathbf{y}}_{\mathbf{n_k}}^{\mathbf{l}})
\end{bmatrix} \tag{16.c}
$$

4.4 O Algoritmo Computacional do Controle Preditivo Utilizando Redes Neurais do Tipo *Feedforward.*

Numa estrutura de controle preditivo, os parâmetros a serem estimados são os próprios controles que manterão o sistema dinâmico em torno de uma trajetória de referência dentro de um horizonte de controle desejado. Nesta seção apresentar-se-á um algoritmo em linguagem humana do filtro de Kalman com o objetivo de facilitar a implementação computacional deste problema, que é um de programação dinâmica não-linear. Como é difícil construir um algoritmo genérico serão assumidos valores particulares para **n** , $\mathbf{n_y}$ e $\mathbf{n_u}$. Neste caso, se adotará **n=5** (horizonte de estimação das variáveis de controle) e $\mathbf{n_y=n_u=3}$ (número de entradas atrasadas das variáveis de estado e de controle na entrada da rede neural).

Como o problema de estimação proposto é não linear, então o algoritmo possuirá uma característica iterativa, ou seja, partirá de uma informação a priori das variáveis de controle, e em seguida, estimará novos valores destas variáveis até que a saída dos estados do sistema obtida pela rede neural convirja para a trajetória de referência conhecida, dentro de um erro aceitável para o horizonte adotado, como ilustra a Figura 4.5.

Ainda com relação a Figura 4.5 observe que os valores das variáveis de estado **y(t-3)**, **y(t-2)** e **y(t-1)** e das variáveis de controle **u(t-3)**, **u(t-2)** e **u(t-1)** devem ser conhecidos assim como a trajetória de referência e a informação a priori dos controles a serem atuados dentro do horizonte de previsão desejado. Deste modo, o algoritmo para a solução do problema proposto pode ser sumarizado nos seguintes passos essenciais:

1) Gerar uma informação a priori dos controles **u(t)** dentro do horizonte desejado. Como a política de controle desejada deverá ser suave como imposta pelo funcional **J** da equação (7), nada mais natural do que iniciá-la com o vetor nulo,

$$\overline{\mathbf{U}}(\mathbf{t,i}) = [\,\overline{\mathbf{u}}(\mathbf{t_o,i}) : \overline{\mathbf{u}}(\mathbf{t_1,i}) : \overline{\mathbf{u}}(\mathbf{t_2,i}) : \overline{\mathbf{u}}(\mathbf{t_3,i}) : \overline{\mathbf{u}}(\mathbf{t_4,i})\,]^{\mathbf{T}} \equiv \mathbf{0} \quad \text{para} \quad \mathbf{i=1} \qquad (17)$$

onde,

$\bar{\mathbf{u}}(\mathbf{t_j},\mathbf{i})$ ... vetor das variáveis de controle no instante $\mathbf{t_j}$ na i-ésima iteração.

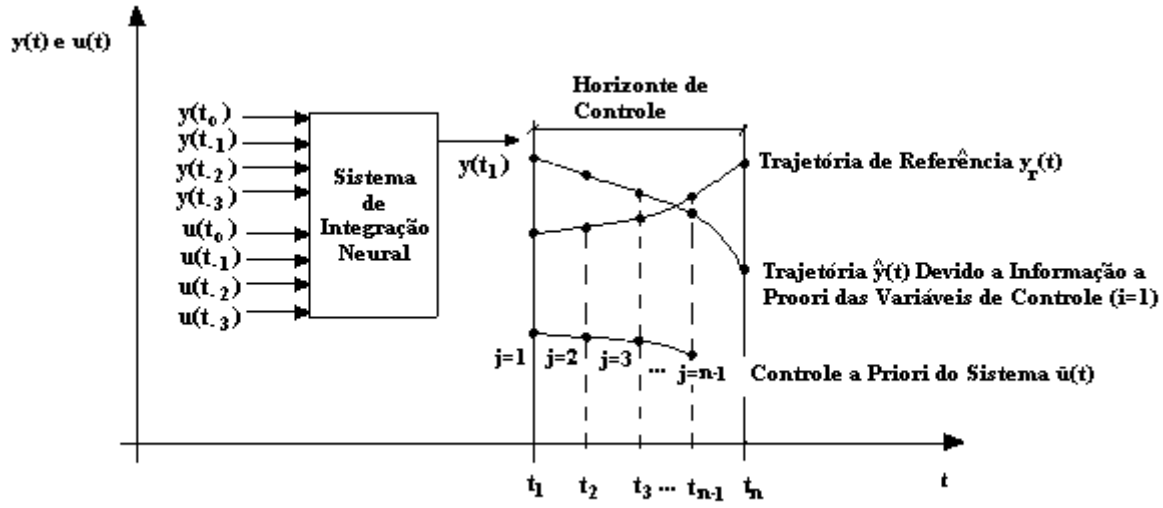


FIGURA 4.5 – Esquema gráfico das condições iniciais das variáveis de controle u(t) que deverão fazer a resposta do sistema dinâmico convergir para a trajetória de referência conhecida através da resolução de um problema de estimação estocástica não-linear.

2) Calcular as derivadas parciais $\dfrac{\partial \hat{\mathbf{y}}(\mathbf{t_j})}{\partial \mathbf{u}(\mathbf{t_k})}$ presentes na equação (10) através das expressões de (16.a) à (16.c) da retro-propagação. A tabela 4.1 ilustra o caso particular para $\mathbf{n=5}$ e $\mathbf{n_y=n_u=3}$.

As derivadas presentes na Tabela 4.1 apresentam duas dificuldades para serem calculadas. A primeira ocorre quando o instante de tempo associado às variáveis de controle **u** for muito atrasado, em relação ao instante de tempo associado às variáveis de estado **y,** a ponto do número de entradas atrasadas presentes na entrada da rede neural não ser suficiente para permitir sua determinação a partir das equações (16.a) à (16.c). Para evitar este inconveniente deve-se dispor em *série* um número suficiente de redes neurais de tal forma que torne possível o cálculo da derivada parcial desejada, como ilustra a Figura 4.6. Nesta figura está esquematizado o cálculo da derivada

$\frac{\partial \hat{y}(t_4)}{\partial u(t_o)}\Big|_{\bar{u}(t_o,i)}$ e a seta interna indica como se deve processar o cálculo da retro-propagação.

TABELA 4.1 – Derivadas parciais $\frac{\partial \hat{y}(t_j)}{\partial u(t_k)}$ necessárias para estimar os controles u(t) dentro de um horizonte de controle com n=5 e entradas atrasadas da rede neural iguais a $n_y=n_u=3$.

J	j-1	Derivadas Parciais (Retro-propagação)
1	0	$\frac{\partial \hat{y}(t_1)}{\partial u(t_0)}\Big\|_{\bar{u}(t_o,i)}$
2	1	$\frac{\partial \hat{y}(t_2)}{\partial u(t_0)}\Big\|_{\bar{u}(t_o,i)}$ $\frac{\partial \hat{y}(t_2)}{\partial u(t_1)}\Big\|_{\bar{u}(t_1,i)}$
3	2	$\frac{\partial \hat{y}(t_3)}{\partial u(t_0)}\Big\|_{\bar{u}(t_o,i)}$ ... $\frac{\partial \hat{y}(t_3)}{\partial u(t_2)}\Big\|_{\bar{u}(t_2,i)}$
4	3	$\frac{\partial \hat{y}(t_4)}{\partial u(t_0)}\Big\|_{\bar{u}(t_o,i)}$ ... $\frac{\partial \hat{y}(t_4)}{\partial u(t_3)}\Big\|_{\bar{u}(t_3,i)}$
5	4	$\frac{\partial \hat{y}(t_5)}{\partial u(t_0)}\Big\|_{\bar{u}(t_o,i)}$ ... $\frac{\partial \hat{y}(t_5)}{\partial u(t_4)}\Big\|_{\bar{u}(t_4,i)}$

A segunda dificuldade está no fato de se ter que utilizar a regra da cadeia combinada com a retro-propagação para calcular as derivadas parciais que se encontram no caso discutido na Figura 4.6 e/ou quando a rede é projetada com mais de uma entrada atrasada. Por exemplo, para se computar o valor da derivada parcial $\frac{\partial \hat{y}(t_4)}{\partial u(t_o)}\Big|_{\bar{u}(t_o,i)}$ deve-se inicialmente aplicar a regra da cadeia sobre o conjunto de equações de (18.a) à (18.d) e só depois aplicar a retro-propagação como deduzida na seção anterior.

$$\hat{y}(t_4) = \hat{f}[\hat{y}(t_3), \hat{y}(t_2), \hat{y}(t_1), \hat{u}(t_3), \hat{u}(t_2), \hat{u}(t_1), \hat{w}] \tag{18.a}$$

onde,

$$\hat{y}(t_3) = \hat{f}[\hat{y}(t_2), \hat{y}(t_1), y(t_0), \hat{u}(t_2), \hat{u}(t_1), u(t_0), \hat{w}] \tag{18.b}$$

$$\hat{y}(t_2) = \hat{f}[\hat{y}(t_1), y(t_0), y(t_{-1}), \hat{u}(t_1), u(t_0), u(t_{-1}), \hat{w}] \quad (18.c)$$

$$\hat{y}(t_1) = \hat{f}[y(t_0), y(t_{-1}), y(t_{-2}), u(t_0), u(t_{-1}), u(t_{-2}), \hat{w}] \quad (18.d)$$

Como pode-se perceber, quando se opta por um horizonte **n** muito grande o cálculo das derivadas parciais presentes na Tabela 4.1 torna-se extremamente complicado, pois, por exemplo, $\mathbf{u(t_o)}$ aparece em todas as equações anteriores dificultando em muito a determinação analítica de $\frac{\partial \hat{y}(t_4)}{\partial u(t_o)}\Big|_{\bar{u}(t_o,i)}$. Para evitar este tipo de dificuldade é conveniente adotar um horizonte **n** bem pequeno.

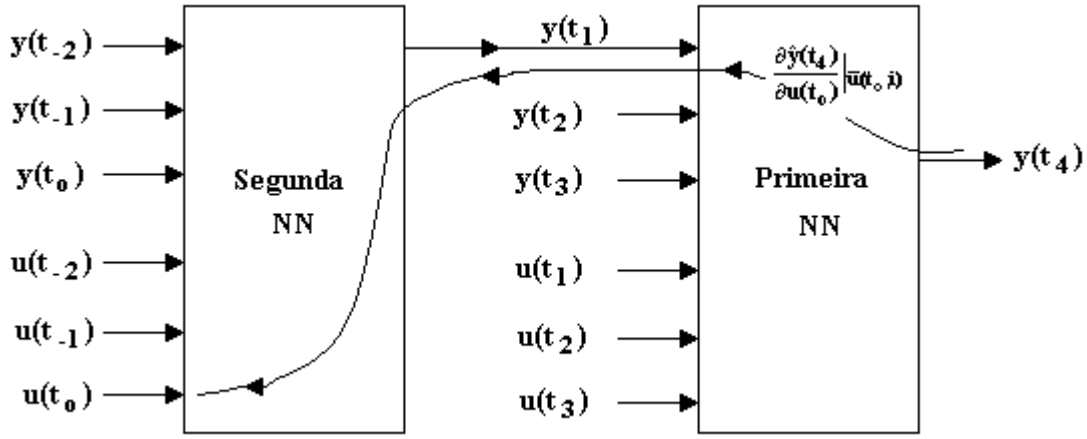


FIGURA 4.6 - Disposição em série das redes neurais que permite computar os valores da derivada parcial $\frac{\partial \hat{y}(t_4)}{\partial u(t_o)}\Big|_{\bar{u}(t_o,i)}$.

3) Montar a matriz $\mathbf{H^u(t,i)}$ e o vetor $\mathbf{\bar{Z}^u(t,i)}$ presentes na expressão $\alpha(i)\cdot\bar{Z}^u(t,i) = H^u(t,i)\cdot[U(t,i) - \bar{U}(t,i)] + V_y(t)$. Deve-se construir a matriz $\mathbf{H^u(t,i)}$ à partir da Tabela 4.1. Para simplificar a notação que está sendo utilizada é conveniente adotar a seguinte definição:

$$A_{k,j} = \frac{\partial \hat{y}(t_k)}{\partial u(t_j)}\Big|_{\bar{u}(t_j,i)} \quad (19)$$

Com base nas equações (19) e (10) e na tabela 4.1 a matriz se reduz a:

$$\mathbf{H^u(t,i)} = \begin{bmatrix} \mathbf{A_{1,0}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{A_{2.0}} & \mathbf{A_{2,1}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{A_{3,0}} & \mathbf{A_{3,1}} & \mathbf{A_{3,2}} & \mathbf{0} & \mathbf{0} \\ \mathbf{A_{4,0}} & \mathbf{A_{4,1}} & \mathbf{A_{4,2}} & \mathbf{A_{4,3}} & \mathbf{0} \\ \mathbf{A_{5,0}} & \mathbf{A_{5,1}} & \mathbf{A_{5,2}} & \mathbf{A_{5,3}} & \mathbf{A_{5,4}} \end{bmatrix} \tag{20}$$

Deste modo expandindo-se a expressão $\alpha(\mathbf{i}) \cdot \overline{\mathbf{Z}}^{\mathbf{u}}(\mathbf{t,i}) = \mathbf{H^u(t,i)} \cdot [\mathbf{U(t,i)} - \overline{\mathbf{U}}(\mathbf{t,i})] + \mathbf{V_y(t)}$, tem-se:

$$\underbrace{\begin{bmatrix} \mathbf{y_r(t_1)} - \overline{\mathbf{y}}(\mathbf{t_1,i}) \\ \mathbf{y_r(t_2)} - \overline{\mathbf{y}}(\mathbf{t_2,i}) \\ \mathbf{y_r(t_3)} - \overline{\mathbf{y}}(\mathbf{t_3,i}) \\ \mathbf{y_r(t_4)} - \overline{\mathbf{y}}(\mathbf{t_4,i}) \\ \mathbf{y_r(t_5)} - \overline{\mathbf{y}}(\mathbf{t_5,i}) \end{bmatrix}}_{\overline{\mathbf{Z}}^{\mathbf{u}}(\mathbf{t,i})} \cdot \alpha(\mathbf{i}) = \underbrace{\begin{bmatrix} \mathbf{A_{1,0}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{A_{2.0}} & \mathbf{A_{2,1}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{A_{3,0}} & \mathbf{A_{3,1}} & \mathbf{A_{3,2}} & \mathbf{0} & \mathbf{0} \\ \mathbf{A_{4,0}} & \mathbf{A_{4,1}} & \mathbf{A_{4,2}} & \mathbf{A_{4,3}} & \mathbf{0} \\ \mathbf{A_{5,0}} & \mathbf{A_{5,1}} & \mathbf{A_{5,2}} & \mathbf{A_{5,3}} & \mathbf{A_{5,4}} \end{bmatrix}}_{\mathbf{H^u(t,i)}} \cdot \underbrace{\begin{bmatrix} \mathbf{u_r(t_0,i)} - \overline{\mathbf{u}}(\mathbf{t_0,i}) \\ \mathbf{u_r(t_1,i)} - \overline{\mathbf{u}}(\mathbf{t_1,i}) \\ \mathbf{u_r(t_2,i)} - \overline{\mathbf{u}}(\mathbf{t_2,i}) \\ \mathbf{u_r(t_3,i)} - \overline{\mathbf{u}}(\mathbf{t_3,i}) \\ \mathbf{u_r(t_4,i)} - \overline{\mathbf{u}}(\mathbf{t_4,i}) \end{bmatrix}}_{\mathbf{U(t,i)} - \overline{\mathbf{U}}(\mathbf{t,i})} + \underbrace{\begin{bmatrix} \mathbf{V_0(t)} \\ \mathbf{V_1(t)} \\ \mathbf{V_2(t)} \\ \mathbf{V_3(t)} \\ \mathbf{V_4(t)} \end{bmatrix}}_{\mathbf{V_y(t)}}$$

(21)

onde,

$\overline{\mathbf{U}}(\mathbf{t,i})$ ... vetor de informações a priori das variáveis de controle adotado no passo 1;

$\mathbf{H^u(t,i)}$ ... matriz de derivadas parciais obtida no passo 2 através da retro-propagação e montada no passo 3;

$\mathbf{U(t,i)}$ ... vetor que deverá ser estimado;

$\mathbf{0} < \alpha(\mathbf{i}) \leq \mathbf{1}$ ... valor que deve ser adotado empiricamente para garantir a hipótese de linearização;

$\mathbf{V_y(t)}$ ... vetor de ruídos de média nula e matriz de covariância $\mathbf{r_y(t)}$.

4) Estimar $\hat{\mathbf{U}}(\mathbf{t,i})$ através das seguintes expressões iterativas do filtro de Kalman (Rios Neto, 2000):

$$\mathbf{k(t,i) = R_u(t) \cdot H^{u^T}(t,i) \cdot [H^u(t,i) \cdot R_u(t) \cdot H^{u^T}(t,i) + R_y(t)]^{-1} \equiv}$$

$$\mathbf{[R_u^{-1}(t) + H^{u^T}(t,i) \cdot R_y^{-1}(t) \cdot H^u(t,i)]^{-1} \cdot H^{u^T}(t,i) \cdot R_y^{-1}(t)} \qquad (22.a)$$

$$\mathbf{\hat{U}(t,i) = \overline{U}(t,i) + \alpha(i) \cdot [\hat{U}(t_{-1}) - \overline{U}(t,i)] +}$$

$$\mathbf{k(t,i) \cdot \alpha(i) \cdot [\overline{Z}^u(t,i) - H^u(t,i) \cdot [\hat{U}(t_{-1}) - \overline{U}(t,i)]]} \qquad (22.b)$$

onde,

$\mathbf{R_u(t)}$ e $\mathbf{R_y(t)}$ ... matrizes de covariâncias, respectivamente, de $\mathbf{V_u(t)}$ e $\mathbf{R_y(t)}$;

$\mathbf{\hat{U}(t_{-1})}$ ... estimativa anterior das variáveis de controle dentro de uma precisão aceitável.

Observe que o processamento das equações (22.a) e (22.b) pode ser efetuado tanto em lotes como recursivamente.

5) Implementar a variável **i** de **i=i+1**, fazer $\mathbf{\overline{U}(t,i+1) = \hat{U}(t,i)}$ e repetir os passos (1), (2),(3) e (4) até que **y(t)** esteja suficientemente próximo de $\mathbf{y_r(t)}$ dentro de um erro desejado. Pode-se utilizar o critério do erro quadrático médio para determinar a última iteração **I** para a estimação atual. Quando isso ocorrer fazer:

$$\mathbf{\hat{R}_u(t,I) = [I_u - k(t,I) \cdot H^u(t,I)] \cdot R_u(t)} \qquad (23)$$

onde,

$\mathbf{\hat{R}_u(t,I)}$ ... estimação da matriz de covariâncias de $\mathbf{[\hat{U}(t,I) - U(t)]}$.

6) Quando a estimação atual convergir para o valor desejado $\mathbf{y_r(t)}$ *avançar apenas um instante de tempo para frente*, mesmo que o horizonte de previsão adotado **n** seja maior que um, e repetir todos os passos anteriores para a nova estimação. Não esquecer que agora ter-se-á $\hat{\mathbf{U}}(\mathbf{t_{-1}}) = \hat{\mathbf{U}}(\mathbf{t,I})$.

4.5 Comentários Finais e Conclusões Sobre a Estrutura de Controle Preditivo.

A utilização do filtro de Kalman como uma ferramenta para a estimação ótima de parâmetros conduz ao projeto de um método para resolver problemas de controle preditivo neural. Vendo a solução do problema de otimização da determinação da ação de controle como um problema de estimação estocástica de parâmetros reduz este problema para um formalmente equivalente àquele de estimação de pesos de uma rede neural *feedforward* em um treinamento supervisionado. Isto permite um tratamento integrado de ambos os problemas, utilizando essencialmente os algoritmos do filtro de Kalman. Em resumo, para a solução de um problema de controle preditivo neural é necessário:

a) escolher uma rede neural *feedforward* com apropriada arquitetura e tamanho, de forma que em um processo envolvendo treinamento supervisionado em tempo real ou não, a rede poderá aprender a dinâmica discretizada no tempo a partir dos conjuntos de dados de entrada/saída obtidos do sistema dinâmico original;

b) resolver com respeito as ações de controle, em tempo real e em pequenas frações de tempo $\Delta\mathbf{t}$, o problema de programação não-linear de minimizar uma função objetivo vinculada à rede neural *feedforward* já treinada para determinar as ações de controle suave que rastreiam a trajetória de referência.

Desta forma, pode-se avaliar a ampla aplicabilidade dos algoritmos envolvendo o filtro de Kalman na solução de problemas que envolvem estruturas de controle preditivo e representação de sistemas dinâmicos não-lineares.

CAPÍTULO 5

REDES NEURAIS EM ESTRUTURA DE INTEGRADORES NUMÉRICOS COMO MODELO INTERNO EM CONTROLE PREDITIVO

5.1 Introdução.

Neste capítulo será apresentada uma metodologia alternativa para representar sistemas dinâmicos não-lineares através de uma rede neural *feedforward* trabalhando em conjunto com integradores numéricos de passo simples ou múltiplos passos. A dinâmica do sistema assim construída poderá ser utilizada também numa estrutura de controle preditivo. Este tipo de abordagem tem a vantagem de reduzir a dimensão da rede neural, e portanto, facilitar seu treinamento. É importante observar que a implementação, teste e ajuste desta metodologia constitue parte da originalidade deste trabalho. Outra parte original deste trabalho, apresentada também no final deste capítulo, trata da questão de representar sistemas dinâmicos através de redes neurais projetadas com as derivadas médias do sistema dinâmico original e inseridas internamente a uma estrutura de integração do tipo Euler. Esta última metodologia pode ser vista como uma contribuição adicional inspirada no trabalho de Rios Neto (2001).

5.2 Modelagem Dinâmica com Redes Neurais em Estruturas de Integradores Numéricos.

Nesta seção uma abordagem teórica e heurística é tomada com o propósito de mostrar explicitamente como utilizar integradores numéricos de sistemas de equações diferenciais ordinárias em esquemas de controle onde um modelo de rede neural interno discreto para frente é necessário (Rios Neto, 2001). A estrutura interna dos algoritmos de integração numérica é aproveitada para obter um modelo de rede neural com arquitetura *feedforward* que necessite somente aprender a função de derivadas do modelo original representado por sistemas de equações diferenciais ordinárias. Nesta metodologia os integradores numéricos e a rede neural *feedforward* trabalharão em

conjunto para representar a dinâmica discretizada do sistema original. A Figura 5.1 ilustra brevemente as diferenças entre a metodologia tradicional e a nova metodologia que utiliza integradores numéricos para representação de sistemas dinâmicos através de redes neurais e sua posterior aplicação numa estrutura de controle preditivo.

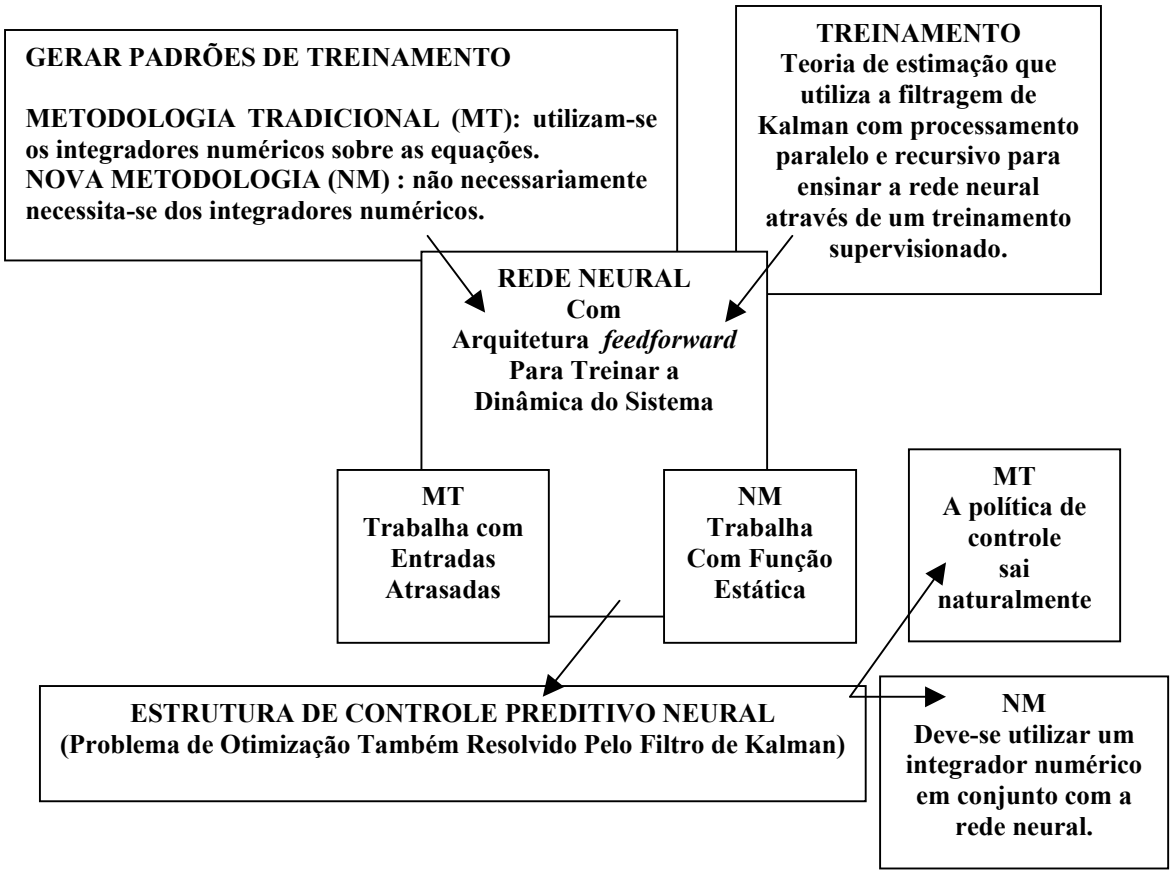


FIGURA 5.1 - Distinção entre as Metodologias Tradicional (MT) e a Nova Metodologia (NM) empregadas para representar sistemas Dinâmicos através de uma rede com arquitetura *feedforward.*

O modelo de simulação computacional provido por um integrador numérico de sistemas de equações diferenciais ordinárias é naturalmente um modelo discreto para frente de sistemas dinâmicos que por si só pode ser utilizado como um modelo interno em esquemas de controle. Estes integradores numéricos possuem características que são bastante relevantes para um modelo de sistemas dinâmicos a ser utilizado em controle, uma vez que eles permitem: processamento paralelo de todas as componentes dos estados do sistema dinâmico, o erro local e a precisão do integrador podem ser avaliados aplicando-se métodos que automaticamente variam a ordem e o tamanho do

passo de integração do integrador e a estimação dos erros globais acumulados podem também ser feita (e.g., Rios Neto e Rama Rao,1990).

Quando uma estrutura de um modelo de integrador numérico é utilizada, é possível ter uma rede neural com arquitetura *feedforward* para aproximar a função de derivadas do modelo matemático de equações diferenciais da dinâmica do sistema. Neste tipo de abordagem, a dificuldade em lidar com muitas entradas no treinamento da rede neural é aliviada, uma vez que é necessário somente aprender uma função algébrica e estática onde as entradas da rede são somente as ocorrências das variáveis de estado e de controle em seus respectivos domínios de atuação. Lembrando-se de que na abordagem convencional às vezes é necessário lidar com muitas entradas atrasadas das variáveis de estado e controle, mas nesta nova metodologia isso não será mais necessário.

Considere um sistema dinâmico e suponha que o modelo matemático ou o modelo de uma rede neural artificial representando a função de derivadas deste sistema são conhecidos. Assim, tem-se:

$$\dot{\mathbf{x}} = \mathbf{f(x,u)} \cong \hat{\mathbf{f}}\mathbf{(x,u,\hat{w})} \tag{1}$$

onde,

$\mathbf{x} \in \mathbf{R^n}$ ... é o vetor das variáveis de estados;

$\mathbf{u} \in \mathbf{R^n}$ ... é o vetor das variáveis de controle;

$\hat{\mathbf{w}}$ ... são os pesos aprendidos pela rede neural;

$\mathbf{f(x,u)}$ ... é a função de derivadas;

$\hat{\mathbf{f}}\mathbf{(x,u,\hat{w})}$ ... representa o treinamento da rede neural para aproximar a função de derivadas.

Considere agora um integrador numérico para obter uma aproximação discreta do sistema da equação (1):

$$\mathbf{x(t+\Delta t) \cong f_n(x(t), x(t-\Delta t), ..., x(t-n_o\Delta t); u(t), ..., u(t-n_o\Delta t); \Delta t)} \quad (2)$$

onde,

$\mathbf{n_o}$ ... é o número de entradas atrasadas do integrador numérico ou a ordem de aproximação.

No Apêndice D há uma classificação dos tipos principais de integradores numéricos (Rao,1984) em função do valor numérico de $\mathbf{n_o}$. Se $\mathbf{n_o}$ é igual a zero tem-se os integradores de passo simples, por exemplo, os integradores de Euler ou de Runge-Kutta. Se $\mathbf{n_o}$ for maior que zero tem-se os integradores de múltiplos passos ou de diferenças finitas, como por exemplo, os integradores de Adams-Bashforth.

O integrador numérico na equação (2) pode ser utilizado como o modelo preditivo discreto do sistema dinâmico dado pela equação (1) em um esquema de controle de modelo interno. O erro em cada passo pode ser controlado variando-se o tamanho do passo ou a ordem do integrador numérico. O resultado numérico do algoritmo pode ser processado em paralelo para cada componente do estado do sistema dinâmico.

Sendo possível aproximar um sistema dinâmico, através de uma estrutura de integrador numérico de equações diferenciais ordinárias trabalhando em conjunto com uma rede neural artificial que necessita somente aprender a função de derivadas do sistema dinâmico, então resta saber agora qual a vantagem de se representar a função de derivadas através de uma rede neural artificial, sendo que a própria função de derivadas, em geral, já possui uma forma analítica conhecida que poderá ser utilizada diretamente nas estruturas do integrador numérico. Não se estaria fazendo um trabalho desnecessário?

Para responder a esta pergunta veja o esquema proposto na Figura 5.2. Como se sabe, todo modelo teórico representado, em geral, através de sistemas de equações diferenciais possui uma margem de erro em relação ao sistema real que ele pretende representar matematicamente, pois quase sempre existirá pelo menos uma variável de estado de difícil modelagem matemática em sistemas muito complexos, acarretando

erro de modelagem. Tendo o modelo teórico em mãos, ainda assim, é necessário resolve-lo através de algum método numérico quando este é não-linear. A solução assim obtida será representada pelo vetor de estados $\mathbf{x}^{m'}$ como ilustra a Figura 5.2.

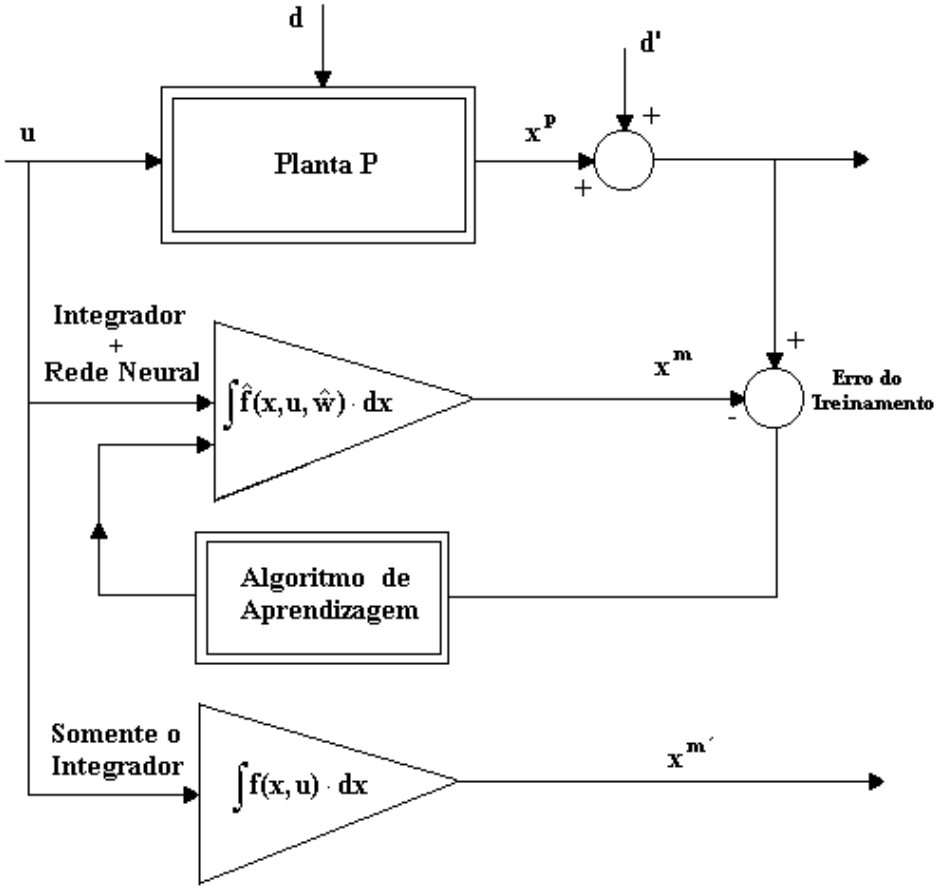


FIGURA 5.2 – Esquema ilustrativo para avaliar as vantagens de se representar à função de derivadas de um sistema dinâmico através de uma rede neural artificial com arquitetura do tipo *feedforward*.

Entretanto, como ilustra a Figura 5.2 não é possível corrigir o modelo teórico em relação ao sistema real quando se utiliza uma estrutura constituída puramente por um integrador numérico para resolver numericamente o modelo teórico original. Por outro lado, quando se utiliza uma rede neural para representar a função de derivadas do modelo teórico do sistema dinâmico ainda será possível continuar o treinamento da rede neural que modela a função de derivadas em tempo real, para que o modelo teórico se aproxime cada vez mais do real. A solução obtida por esse último método é

representada pelo vetor de estados $\mathbf{x^m}$ da Figura 5.2. Nesta última metodologia a diferença do erro de $\mathbf{(x^p-x^m)}$ tende a diminuir com o passar do tempo de funcionamento da planta. No primeiro modelo a diferença do erro $\mathbf{(x^p-x^{m'})}$ tende a crescer.

Por outro lado, se for adicionado e ajustado um processo de Gauss-Markov ao modelo dinâmico original para se compensar constantemente em tempo real os erros de modelagem, isso também poderia ser suficiente para corrigir o modelo teórico em relação ao real da planta. Entretanto, neste caso, certamente não se teria a capacidade de aprendizagem da solução proposta com redes neurais e integradores.

Uma vantagem mostrada no esquema da Figura 5.2 é que não é necessário conhecer previamente a função de derivadas do sistema dinâmico para utilizar a estrutura de treinamento composta por um integrador numérico e uma rede neural. Caso se conheça somente um número suficientemente grande de entradas/saídas da planta, por exemplo, obtidas através de sensores, então através de um treinamento supervisionado a estrutura composta pelo conjunto integrador numérico e rede neural poderá ser treinada para aprender a dinâmica do sistema. É interessante perceber que, mesmo neste caso, a rede continuará fazendo um papel semelhante ao da função de derivadas e, portanto ela poderá ser projetada com uma arquitetura que, em geral, exigirá menos camadas e menos neurônios.

Mesmo quando se conhece a priori a função de derivadas, há duas formas de treinamento supervisionado que esta metodologia permite realizar. Como ilustra a Figura 5.3, as duas maneiras distintas de realizar este treinamento são:

a) a primeira metodologia é simplesmente gerar aleatoriamente valores suficientes da função de derivadas dentro de um domínio de interesse e treinar diretamente a rede neural para aprender a função de derivadas do modelo teórico. Terminado o treinamento, a simulação dos resultados deverá ser feita aplicando-se a função de derivadas treinada pela rede no lugar da teórica sobre a estrutura do integrador. Para a análise da compatibilidade dos erros obtidos pela rede e pelo integrador, a

solução numérica do integrador deverá ser obtida pela aplicação das duas funções de derivadas, a treinada e a original, para que os resultados da propagação da solução sejam comparados. Nesta metodologia, o integrador numérico só será utilizado na simulação dos resultados;

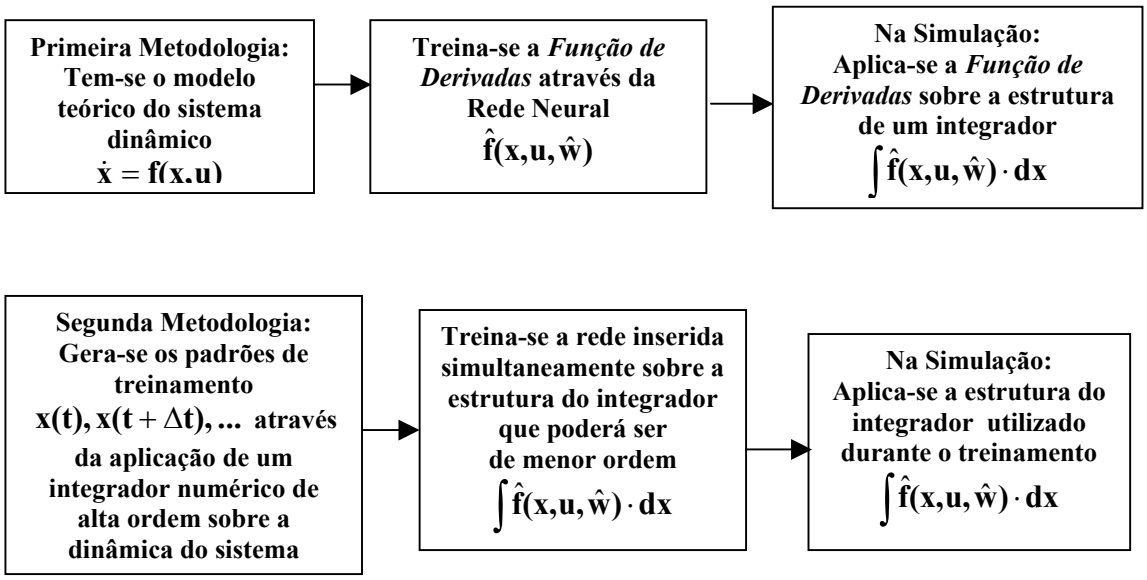


FIGURA 5.3 – Duas maneiras distintas de se treinar uma rede neural inserida na estrutura de um integrador numérico.

b) na segunda metodologia geram-se as soluções temporais do sistema dinâmico $x(t), x(t+\Delta t), ..., x(t+n\cdot\Delta t)$ para um número suficiente de condições iniciais e valores de controle através da estrutura de um integrador, de preferência, de alta ordem para garantir a alta precisão dos padrões de treinamento. O aprendizado de função com o papel de função de derivadas pela rede neural só será possível se a rede for treinada amarrada à estrutura do integrador. É interessante observar que durante o treinamento poder-se-á diminuir a ordem do integrador em relação àquele utilizado para gerar os padrões de treinamento. A análise do erro obtido pelo conjunto integrador e rede neural é realizada também pela comparação das soluções numéricas propagadas pelas funções de derivadas original e treinada. Nesta metodologia, o integrador numérico é utilizado na geração dos padrões de treinamento e é requerido durante o treinamento e a simulação da rede.

O Erro do Integrador (**EI**) numérico é dividido em quatro partes (ver Apêndice D): erro de truncamento local (τ), erro global (υ), erro local (λ) e *round-off error* (**re**). Na primeira metodologia de treinamento, apresentada na Figura 5.3, o erro quadrático medito (MEQ) cometido pela rede será propagado pelo integrador. Na segunda metodologia tem que se explorar melhor se este erro será totalmente propagado pelo integrador. A primeira vista, parece ser mais vantajoso utilizar o segundo método.

O detalhe mais interessante mostrado na Figura 5.3 é que a segunda metodologia de treinamento pode permitir reduzir a ordem do integrador, sem diminuir a precisão da solução obtida para o sistema dinâmico, quando se utilizam redes neurais para interpolar as funções de derivadas do sistema dinâmico. O raciocínio é relativamente simples, por exemplo, geram-se os padrões de treinamento através de um integrador Runge-Kutta de quarta ordem e treina-se a rede com um integrador de baixa ordem como, por exemplo, Euler até os pesos da rede gerarem soluções na mesma magnitude de erro daquele gerado pelo integrador de alta-ordem. Entretanto, no caso particular mencionado neste parágrafo, o integrador com estrutura Euler, que é utilizado durante o treinamento da rede, fará com que a rede neural associada a ele aprenda a *função de derivadas médias* e não propriamente a função de derivadas exatas. Isto ficará claro em uma próxima seção.

5.3 Integrador Neural em Controle Preditivo.

Sendo possível representar sistemas dinâmicos através do conjunto integrador numérico e redes neurais resta saber como projetar este esquema sobre uma estrutura de controle preditivo. Neste caso, o algoritmo esquematizado na seção 4.4 do Capítulo 4 continuará valendo, com a exceção que mudará a forma de calcular as derivadas parciais presentes na equação (10) do Capítulo 4. No caso presente, deverá ser empregada a regra da cadeia sobre a estrutura do integrador combinada com a retro-propagação da rede neural. Desta forma, quanto maior a ordem do integrador mais complexo se tornará o cálculo destas derivadas. Por ora, é importante salientar que o problema continua sendo determinar um controle suave que rastreie a trajetória de referência, minimizando-se o índice de performance quadrático do tipo:

$$J = [\sum_{j=1}^{n}[y_r(t_j) - y_n(t_j)]^T \cdot r_y^{-1}(t) \cdot [y_r(t_j) - y_n(t_j)] + \sum_{j=0}^{n-1}[u(t_j) - u(t_{j-1})]^T \cdot r_u^{-1}(t) \cdot [u(t_j) - u(t_{j-1})]]/2 \quad (3.a)$$

para,

$$t_j = t + j \cdot \Delta t \quad (3.b)$$

onde,

$y_r(t_j)$ ... é a trajetória de referência;

n ... horizonte de atuação do controle suave;

$r_y(t_j)$ e $r_u(t_j)$ ... matrizes de pesos positiva definida;

$\hat{y}(t_j)$ ... é a saída aproximada do sistema dinâmico da equação (1).

A saída aproximada para o sistema dinâmico da equação (1) pode ser formalmente representada por:

$$y_n(t_j) \cong g[x(t_j)] \cong g[f_n(x(t_{j-1}), ..., x(t_{j-1-n_o}); u(t_{j-1}), ..., u(t_{j-1-n_o}); \Delta t; \hat{w})] =$$

$$g_n(x(t_{j-1}), ..., x(t_{j-1-n_o}); u(t_{j-1}), ..., u(t_{j-1-n_o}); \hat{w}) \quad (4)$$

onde,

$\hat{w}$ ... pesos de treinamento estimados da rede neural que representa a função de derivadas do sistema dinâmico;

$y_n(t_j)$ ... resposta do sistema dinâmico discretizado pelo conjunto integrador e rede.

A possibilidade de ajustar o nível de aproximação do integrador numérico e da rede neural garante a precisão necessária para $\mathbf{y_n(t_j)}$ ao longo do horizonte preditivo. A solução do problema de programação não-linear do índice de desempenho dado pela equação (3.a), por qualquer que seja o método, envolverá a necessidade de calcular uma aproximação do gradiente de saída do sistema dinâmico, para obter uma aproximação linearizada em cada passo de busca:

$$\mathbf{y_n(t_j) = \bar{y}(t_j) + \sum_{k=0}^{j-1} [\partial y_n(t_j)/\partial u(t_k)]_{\{\bar{u}(t_k)\}} \cdot [u(t_k) - \bar{u}(t_k)]} \qquad (5)$$

onde $\mathbf{y_n(t_j)}$ é também uma função de $\mathbf{u(t_{j-2}), ..., u(t_{j-n_o+1})}$ através de $\mathbf{x_n(t_{j-1}), ..., x_n(t_{j-n_o})}$. As derivadas parciais são calculadas, como mencionado já anteriormente, utilizando-se a regra da cadeia.

Note que em vez de se utilizar uma rede neural para prover um modelo discreto para frente para a dinâmica do sistema, pode-se utilizar diretamente o algoritmo do integrador numérico como o modelo discreto para frente para aproximar o modelo matemático da dinâmica do sistema da equação (1). No caso do esquema de controle preditivo, o modelo do integrador numérico da equação (2) pode ser diretamente utilizado para calcular as derivadas parciais da equação (5).

5.4 Método de Redução da Ordem do Integrador ou das Derivadas Médias: Fundamentos Teóricos.

Nesta seção será inicialmente desenvolvida a teoria matemática que permite representar sistemas dinâmicos não-lineares através de integradores numéricos de baixa ordem, em particular o integrador com estrutura de Euler, com uma rede *feedforward* em sua estrutura interna que aprenderá a função de derivadas médias. Demonstrar-se-á que a precisão alcançada por este integrador de baixa ordem será tão próxima quanto se queira de qualquer outro integrador de mais alta ordem. Isto é possível graças à capacidade de

treinamento supervisionado das redes *feedforward* e das propriedades qualitativas dos sistemas de equações diferenciais ordinárias de primeira ordem não-lineares autônomas. A seguir são enunciados e demonstrados alguns teoremas bastante importantes, do ponto de vista teórico, que serão utilizados na seção 5.5 para desenvolver o algoritmo associado ao método de representação de sistemas dinâmicos não-lineares apresentado aqui.

Seja o sistema autônomo de equações diferenciais ordinárias não-lineares,

$$\dot{\mathbf{y}} = \mathbf{f(y)} \tag{6.a}$$

onde,

$$\mathbf{y} = [\mathbf{y_1} \quad \mathbf{y_2} \quad \mathbf{\dots} \quad \mathbf{y_n}]^{\mathbf{T}} \tag{6.b}$$

$$\mathbf{f(y)} = [\mathbf{f_1(y)} \quad \mathbf{f_2(y)} \quad \mathbf{\dots} \quad \mathbf{f_n(y)}]^{\mathbf{T}} \tag{6.c}$$

Seja também, por definição, $\mathbf{y_j^i = y_j^i(t)}$ para **j=1, 2, ..., n** *uma trajetória* da *família de soluções* do sistema de equações diferenciais não-lineares $\dot{\mathbf{y}} = \mathbf{f(y)}$ que passa por $\mathbf{y_j^i(t_o)}$ no instante inicial $\mathbf{t_o}$, partindo de um domínio de interesse $\mathbf{[y_j^{min}(t_o), y_j^{max}(t_o)]^n}$ onde $\mathbf{y_j^{min}(t_o)}$ e $\mathbf{y_j^{max}(t_o)}$ são finitos. Se for possível afirmar que há condições iniciais em todos os pontos interiores deste domínio, então implicará que para qualquer discretização **i,** $\mathbf{y_j^i(t_o)}$ é um conjunto de condições iniciais possíveis em $\mathbf{t_o}$ do sistema de equações diferenciais não-linear dado por (6.a) dentro de um domínio de interesse em $\mathbf{[y_j^{min}, y_j^{max}]^n}$ para **j=1, 2, ..., n**.

É conveniente introduzir a seguinte notação vetorial a respeito dos possíveis conjuntos de condições iniciais e das soluções de (6.a):

$$\mathbf{y_o^i = y^i(t_o) = [y_1^i(t_o) \; y_2^i(t_o) \; \dots y_n^i(t_o)]^T} \tag{7.a}$$

$$\mathbf{y^i = y^i(t) = [y_1^i(t)\ y_2^i(t)\ ...\ y_n^i(t)]^T} \qquad (7.b)$$

onde **i= 1, 2, ..., ∞;** e ∞ indica que na malha de discretização o número de pontos pode ser tão grande quanto se queira. A Figura 5.4 é um esboço ilustrativo para o caso unidimensional na variável **j** de uma família de soluções de $\dot{\mathbf{y}}_1 = \mathbf{f(y_1)}$ que parte das condições iniciais $\mathbf{y_1^i(t_o)}$ contidas dentro de uma região de interesse $\mathbf{[y_j^{min}(t_o), y_j^{max}(t_o)]^1}$.

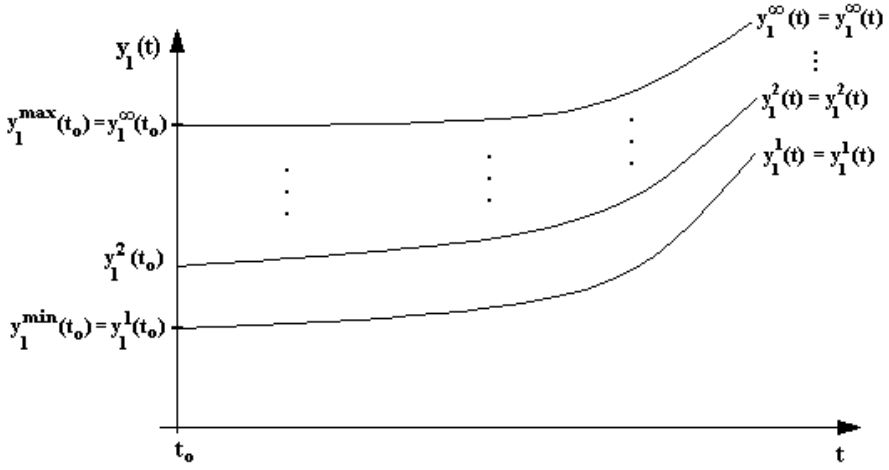


FIGURA 5.4 – Família de curvas $\mathbf{y_1^i = y_1^i(t)}$ soluções de $\dot{\mathbf{y}}_1 = \mathbf{f(y_1)}$ do domínio de interesse $\mathbf{[y_1^{min}(t_o), y_1^{max}(t_o)]^1}$ em $\mathbf{t_o}$.

É possível demonstrar dois importantes resultados (e.g., Braun, 1983) a respeito das soluções do sistema de equações diferenciais (6.a). O primeiro diz respeito a existência e unicidade das soluções e o segundo lida com a existência de soluções estacionárias de (6.a).

Teorema 1 (T1)- *Admita-se que cada uma das funções* $\mathbf{f_1(y_1, y_2, ..., y_n), ..., f_n(y_1, y_2, ..., y_n)}$ *tenha derivadas parciais contínuas com respeito a* $\mathbf{y_1, ..., y_n}$. *Então, o problema de valor inicial* $\dot{\mathbf{y}} = \mathbf{f(y)}$, $\mathbf{y(t_o)}$ *no domínio de interesse* $\mathbf{[y_j^{min}, y_j^{max}]^n}$ *para* ***j=1, 2, ..., n,*** *em* $\mathbf{t_o}$, *tem uma e uma única solução*

$\mathbf{y^i}=\mathbf{y^i(t)}$, *em* $\mathbf{R^n}$, *a partir de cada* $\mathbf{y^i(t_o)}$. *Se duas soluções* $\mathbf{y} = \phi\mathbf{(t)}$ *e* $\mathbf{y} = \varphi\mathbf{(t)}$ *têm um ponto em comum, então elas devem ser idênticas.*

Propriedade 1 (P1)- *Se* $\mathbf{y} = \phi\mathbf{(t)}$ *é uma solução de (6.a), então* $\mathbf{y} = \phi\mathbf{(t + c)}$ *é também uma solução de (6.a) sendo* ***c*** *uma constante real qualquer.*

A **P1** não é verdadeira se a função (45.a) depender explicitamente do tempo (Braun, 1983) pois, neste caso, supor que $\mathbf{y} = \phi\mathbf{(t + c)}$ seja também uma solução do sistema não-autônomo $\dot{\mathbf{y}} = \mathbf{f(t, y)}$ implicará numa outra equação diferencial do tipo $\dot{\mathbf{y}} = \mathbf{f(t + c, y)}$ levando a uma contradição.

Como, em geral, $\dot{\mathbf{y}}^\mathbf{i} = \mathbf{f(y^i)}$ não possui solução analítica, então é comum apenas conhecer para $\mathbf{y^i}=\mathbf{y^i(t)}$ um conjunto discreto de pontos $\mathbf{[y^i(t + k \cdot \Delta t)\ \ y^i[t + (k + 1) \cdot \Delta t]\ ...\ y^i[t + (k + n) \cdot \Delta t]\ ]} \equiv \mathbf{[{}^k y^i\ \ {}^{k+1}y^i\ ...\ {}^{k+n}y^i\ ]}$ para ${}^\mathbf{k}\mathbf{y^i}=\mathbf{y^i(t + k \cdot \Delta t)}$ no horizonte $\mathbf{[t_k\ , t_{k+n}]}$ e $\Delta\mathbf{t} = \mathbf{(t_{k+n} - t_k)/n}$ uma constante.

Propriedade 2 (P2)- *Como* $\dot{\mathbf{y}}^\mathbf{i} = \mathbf{f(y^i)}$ *é dada por (6.a) segue-se de imediato que, se* ${}^\mathbf{k}\mathbf{y^i} = \mathbf{y^i(t + k \cdot \Delta t)}$ *é conhecido, então* ${}^\mathbf{k}\dot{\mathbf{y}}^\mathbf{i} = \dot{\mathbf{y}}^\mathbf{i}\mathbf{(t + k \cdot \Delta t)}$ *também será.*

Esta propriedade, apesar de bastante óbvia, será bastante útil quando da aplicação dos teoremas do valor médio diferencial e integral sobre os conjuntos de pontos ${}^\mathbf{k}\mathbf{y^i}$ e ${}^\mathbf{k}\dot{\mathbf{y}}^\mathbf{i}$ no espaço $\mathbf{R^n}$ para determinar propriedades importantes a respeito da tangente da secante entre dois pontos consecutivos ${}^\mathbf{k}\mathbf{y^i}$ e ${}^{\mathbf{k+1}}\mathbf{y^i}$ sobre a curva $\mathbf{y^i(t)}$ que é uma solução particular de $\dot{\mathbf{y}} = \mathbf{f(y)}$ que parte de $\mathbf{y^i(t_o)}$. Na verdade, dizer que a partir da solução ${}^\mathbf{k}\mathbf{y^i}$ da equação diferencial (6.a) pode-se obter os valores de ${}^\mathbf{k}\dot{\mathbf{y}}^\mathbf{i}$, não tem sentido do ponto de vista da teoria das equações diferenciais, mas quando se pretende modelar uma rede neural, com a dinâmica dada por (6.a), através de um aprendizado *supervisionado,* passa a ter.

Por definição (e.g., Munem e Foulis, 1978) a secante entre dois pontos $^{k}\mathbf{y}^{i}$ e $^{k+1}\mathbf{y}^{i}$ pertencentes a curva $\mathbf{y^i(t)}$ é o segmento de reta que une estes dois pontos. Logo, as tangentes das secantes entre os pontos $^{k}\mathbf{y}_{1}^{i}$ e $^{k+1}\mathbf{y}_{1}^{i}$, $^{k}\mathbf{y}_{2}^{i}$ e $^{k+1}\mathbf{y}_{2}^{i}$, **...**, $^{k}\mathbf{y}_{n}^{i}$ e $^{k+1}\mathbf{y}_{n}^{i}$ são definidas como:

$$\mathbf{tan}_{\Delta t}\alpha^{i}(t + k \cdot \Delta t) = \mathbf{tan}_{\Delta t}{}^{k}\alpha^{i} = [\ \mathbf{tan}_{\Delta t}{}^{k}\alpha_{1}^{i}\ \ \mathbf{tan}_{\Delta t}{}^{k}\alpha_{2}^{i}\ \ \mathbf{...}\ \ \mathbf{tan}_{\Delta t}{}^{k}\alpha_{n}^{i}\ ]^{T} \quad (8.a)$$

Com,

$$\mathbf{tan}_{\Delta t}{}^{k}\alpha_{j}^{i} = \frac{^{k+1}\mathbf{y}_{j}^{i} - {}^{k}\mathbf{y}_{j}^{i}}{\Delta t}, \text{ para} \qquad \mathbf{j=1, 2, ..., n} \quad (8.b)$$

onde,

α_{j}^{i} ... ângulo da secante que une os dois pontos $^{k}\mathbf{y}_{j}^{i}$ e $^{k+1}\mathbf{y}_{j}^{i}$ pertencentes a curva $\mathbf{y}_{j}^{i}(t)$.

Propriedade 3 (P3)- *Se* $^{k}\mathbf{y}^{i}$ *é uma discretização da solução de* $\dot{\mathbf{y}}^{i} = \mathbf{f(y^i)}$ *e* $\Delta t \neq 0$ *então,* $\mathbf{tan}_{\Delta t}{}^{k}\alpha^{i}$ *existe e é única.*

Demonstração: O **T1** garante a existência e unicidade de $\mathbf{y}_{j}^{i}(t)$ e, portanto, dos $^{k}\mathbf{y}_{j}^{i}$ e $^{k+1}\mathbf{y}_{j}^{i}$, dada uma discretização $\Delta t \neq 0$, acarretando a existência e unicidade de $\mathbf{tan}_{\Delta t}{}^{k}\alpha^{i}$.□

Propriedade 4 (P4)- *Dado o vetor de estados* $\mathbf{y^i(t_o)}$ *no instante* $\mathbf{t_o}$ *do sistema de equações diferenciais não-linear* $\dot{\mathbf{y}} = \mathbf{f(y)}$ *e se o vetor* $[\ ^{k}\mathbf{y}^{i}\ \ ^{k+1}\mathbf{y}^{i}\ \mathbf{...}\ ^{k+n}\mathbf{y}^{i}\ ]$ *é a discretização de uma trajetória solução tal que para* $^{k}\mathbf{y}^{i} = \mathbf{y^i(t + k \cdot \Delta t)}$ *é uma solução dele então, as* $\mathbf{tan}_{\Delta t}{}^{k+l}\alpha^{i}$ *para* ***l = 0, 1, ..., (n-1)*** *existem e são únicas.*

Demonstração: por indução e aplicação sucessiva de **P3** as $\mathbf{tan}_{\Delta t}{}^{k+l}\alpha^{i}$ para **l=0, 1, ..., (n-1)** existem e são únicas.□

Outros dois teoremas bastante importantes que relacionam os valores de $\mathbf{tan}_{\Delta t}{}^{k}\alpha^{i}$ e $\mathbf{tan}_{\Delta t}{}^{k}\dot{\alpha}^{i}$, respectivamente, com as derivadas médias calculadas a partir de $[\,{}^{k}y^{i}\ {}^{k+1}y^{i}\ \ldots\ {}^{k+n}y^{i}\,]$ e de $[\,{}^{k}\dot{y}^{i}\ {}^{k+1}\dot{y}^{i}\ \ldots\ {}^{k+n}\dot{y}^{i}\,]$ são os teoremas do valor médio diferencial e integral (e.g., Wilson (1958), Munem e Foulis (1978), Sokolnikoff e Redheffer (1966)) que são enunciados a seguir sem demonstração:

Teorema 2 (T2)- *(Teorema do valor médio diferencial). Se uma função* $\mathbf{y_j^i(t)}$ *para* ***j=1, 2, ..., n*** *é definida e contínua no intervalo fechado* $[t_k, t_{k+1}]$ *e diferenciável no intervalo aberto* (t_k, t_{k+1})*, então existe pelo menos um número* t_k^* *com* $t_k < t_k^* < t_k + \Delta t$ *tal que*

$$\dot{y}_j^i(t_k^*) = \frac{{}^{k+1}y_j^i - {}^{k}y_j^i}{\Delta t} \tag{9}$$

O teorema **T2** afirma (e.g., Munem e Foulis (1978), Kaplan (1998)) que dada uma secante ao gráfico $\mathbf{y^i(t)}$ de uma curva diferenciável, é sempre possível encontrar um ponto do gráfico situado entre os dois pontos ${}^{k+1}y^i$ e ${}^{k}y^i$ de interseção da secante com a curva $\mathbf{y^i(t)}$ em t_k e t_{k+1}, tal que a reta tangente ao ponto $\mathbf{y^i(t_k^*)}$ seja paralela à secante. A essa propriedade geométrica interessante de $\mathbf{\dot{y}^i(t_k^*)}$ dá-se o nome de *derivada média* da função $\mathbf{y^i(t)}$ sobre o intervalo fechado $[t_k, t_{k+1}]$.

Teorema 3 (T3)- *(Teorema do valor médio integral). Se uma função* $\mathbf{y_j^i(t)}$ *para* ***j=1, 2, ..., n*** *é uma função contínua no intervalo fechado* $[t_k, t_{k+1}]$*, então existe pelo menos um número* t_k^x *interno a* $[t_k, t_{k+1}]$ *tal que*

$$\mathbf{y(t_k^x)\cdot \Delta t = \int_{t_k}^{t_{k+1}} y^i(t)\cdot dt} \qquad (10)$$

Em geral $\mathbf{t_k^*}$ é diferente de $\mathbf{t_k^x}$ e é importante observar também que os teoremas do valor médio nada dizem como determinar os valores de $\mathbf{t_k^*}$ e $\mathbf{t_k^x}$. Os teoremas simplesmente afirmam que $\mathbf{t_k^*}$ e $\mathbf{t_k^x}$ estão contidos no intervalo $\mathbf{[t_k, t_{k+1}]}$.

Propriedade 5 (P5)- *Aplicar o teorema **T3** sobre a curva* $\mathbf{\dot{y}^i(t)}$ *é equivalente a aplicar o teorema **T2** sobre a curva* $\mathbf{y^i(t)}$ *ambos sobre o mesmo intervalo fechado* $\mathbf{[t_k, t_{k+1}]}$*, ou seja,* $\mathbf{\dot{y}^i(t_k^x) = \dot{y}^i(t_k^*)}$.

Demonstração: do teorema **T3** aplicado a curva $\mathbf{\dot{y}^i(t)}$ resulta que existe um $\mathbf{\dot{y}^i(t_k^x)}$ para $\mathbf{t_k < t_k^x < t_{k+1}}$ tal que $\mathbf{\dot{y}^i(t_k^x)\cdot \Delta t = \int_{t_k}^{t_{k+1}} \dot{y}^i(t)\cdot dt = {}^{k+1}y^i - {}^{k}y^i}$ pelo teorema fundamental do cálculo. Assim, $\mathbf{\dot{y}^i(t_k^x) = \frac{{}^{k+1}y^i - {}^{k}y^i}{\Delta t} = tan_{\Delta t}{}^{k}\alpha^i}$. Por outro lado, a aplicação do teorema **T2** sobre a curva $\mathbf{y^i(t)}$ implica que existe um $\mathbf{\dot{y}^i(t_k^*)}$ para $\mathbf{t_k < t_k^* < t_{k+1}}$ tal que $\mathbf{\dot{y}^i(t_k^*) = \frac{{}^{k+1}y^i - {}^{k}y^i}{\Delta t} = tan_{\Delta t}{}^{k}\alpha^i}$. Logo, $\mathbf{\dot{y}^i(t_k^x) = \dot{y}^i(t_k^*)}$.□

Este teorema não permite afirmar, entretanto, que $\mathbf{t_k^x = t_k^*}$.

Propriedade 6 (P6)- *A derivada média* $\mathbf{\dot{y}^i(t_k^x)}$ *sobre o gráfico de* $\mathbf{y^i(t)}$ *no intervalo fechado* $\mathbf{[t_k, t_{k+1}]}$ *é igual a* $\mathbf{tan_{\Delta t}{}^{k}\alpha^i}$*, como conseqüência imediata da própria definição de derivadas médias*.

Propriedade 7 (P7)- *Observe que se* $\dot{y}^i(t_k^x) = \dot{y}^i(t_k^*)$ *(P5) e* $\dot{y}^i(t_k^*)$ *é a derivada média de* $y^i(t)$ *(teorema T2) no intervalo fechado* $[t_k, t_{k+1}]$ *então,* $\dot{y}^i(t_k^x)$ *também será numericamente igual a derivada média de* $y^i(t)$ *para o mesmo intervalo fechado.*

Isto justifica porque se utilizou o índice **x** em t_k^x na formulação do teorema anterior e não o índice *. Deste modo, obedecendo a notação adotada neste trabalho, tem-se que $\ddot{y}^i(t_k^*)$ é aplicação do teorema **T2** sobre a curva $\dot{y}^i(t)$, $\dddot{y}^i(t_k^*)$ é a aplicação do teorema **T2** sobre a curva $\ddot{y}^i(t)$ e assim por diante. Por outro lado, $\ddot{y}^i(t_k^x)$ é a aplicação do teorema **T3** sobre a curva $\ddot{y}^i(t)$ e assim por diante, ou seja, o índice * está sempre associado a aplicação do teorema **T2** e o índice **x** está sempre associado a aplicação do teorema **T3**. A aplicação do teorema **T2** sobre a curva $\dot{y}^i(t)$ implica que existe pelo menos um t_k^* em $[t_k, t_{k+1}]$ tal que $\ddot{y}(t_k^*) = \tan_{\Delta t}^k \dot{\alpha}^i$.

As equivalências numérica e geométrica entre $\dot{y}^i(t_k^x)$ e $\dot{y}^i(t_k^*)$ podem ser interpretadas como ilustra a Figura 5.5. Observe que há três interpretações geométricas possíveis para a grandeza $\dot{y}^i(t_k^x) = \dot{y}^i(t_k^*) = \tan_{\Delta t}^k \alpha^i$: a primeira interpretação é que ela é realmente a derivada média de $y^i(t)$ no intervalo fechado $[t_k, t_{k+1}]$, a segunda é que $\Delta t \cdot \tan_{\Delta t}^k \alpha^i$ para $\Delta t = t_{k+1} - t_k$ é a área sobre a curva $\dot{y}^i(t)$ sobre o mesmo intervalo $[t_k, t_{k+1}]$ e a terceira interpretação é que $\tan_{\Delta t}^k \alpha^i$ é realmente a derivada exata de pelo menos um ponto interior ao intervalo $[t_k, t_{k+1}]$ da função $y^i(t)$ como ilustram as duas retas tangentes sobre $y^i(t)$ na Figura 5.5 inferior. A **P6** sugere que se existir mais de um valor para $\dot{y}^i(t_k^x)$, então eles deverão ser iguais a $\tan_{\Delta t}^k \alpha^i$.

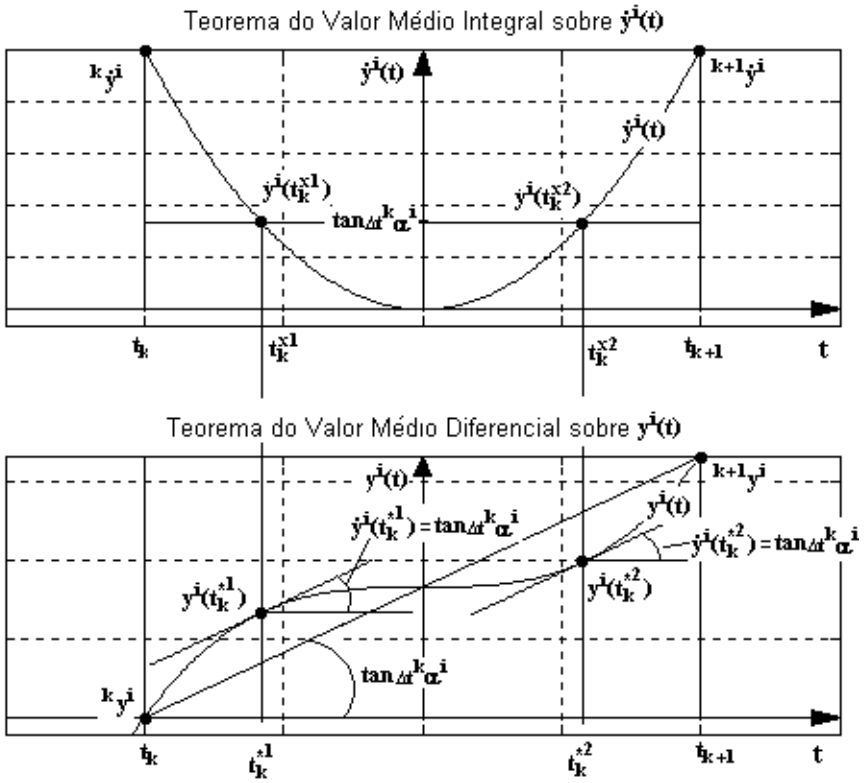


FIGURA 5.5 – Interpretações geométricas de $\dot{y}^i(t_k^x)$ e $\dot{y}^i(t_k^*)$.

A Figura 5.5 acima ilustra esta situação, e como pode ser visto, $\dot{y}^i(t_k^{x1}) = \dot{y}^i(t_k^{x2}) = \tan_{\Delta t}{}^k\alpha^i$ mesmo para o caso onde t_k^{x1} é diferente de t_k^{x2}. Os teoremas **T2** e **T3** também podem ser utilizados para garantir a existência de $\tan_{\Delta t}{}^k\alpha^i$, mas não garantem que elas são únicas, para isso é necessário realmente utilizar o teorema **T1**. A Figura 5.5 também *sugere* que $t_k^{x1} = t_k^{*1}$ e $t_k^{x2} = t_k^{*2}$, mas isso não pode ser demonstrado com base nos teoremas apresentados aqui.

Teorema 4 (T4)- *A solução* $^{k+1}y_j^i$ *para* ***j=1, 2, ..., n*** *do sistema de equações diferenciais não-linear* $\dot{y}^i = f(y^i)$ *pode ser determinada através da relação* $^{k+1}y_j^i = \tan_{\Delta t}{}^k\alpha^i \cdot \Delta t + {}^ky_j^i$ *para um dado* $^ky^i$ *e* Δt.

Demonstração: Se $\dot{y}^i = \frac{dy^i}{dt} = f(y^i)$, então $\int_{{}^k y^i}^{{}^{k+1} y^i} dy^i = \int_{t_k}^{t_{k+1}} f(y^i) \cdot dt$. Assim,

$$ {}^{k+1}y^i = \int_{t_k}^{t_{k+1}} f(y^i) \cdot dt + {}^k y^i \tag{11} $$

A aplicação do teorema do valor médio integral **T3** sobre a curva $\dot{y}^i(t)$ no intervalo fechado $[t_k, t_{k+1}]$ implica que existe um pelo menos um número t_k^x em $[t_k, t_{k+1}]$ tal que

$$ \dot{y}(t_k^x) \cdot \Delta t = \int_{t_k}^{t_{k+1}} \dot{y}^i(t) \cdot dt = \int_{t_k}^{t_{k+1}} f(y^i) \cdot dt \tag{12} $$

Pela **P6** $\dot{y}^i(t_k^x) = \tan_{\Delta t}{}^k \alpha^i$. Logo, substituindo-se a grandeza $\dot{y}^i(t_k^x)$ na equação (12) e esta na equação (11), resulta que:

$$ {}^{k+1}y^i = \tan_{\Delta t}{}^k \alpha^i \cdot \Delta t + {}^k y^i \quad \square \tag{13} $$

Corolário 1 (C1)- *A solução* ${}^{k+m}y_j^i$ *para* ***j=1, 2, ..., n*** *do sistema de equações diferenciais não-linear* $\dot{y}^i = f(y^i)$ *pode ser determinada para um dado* ${}^k y^i$ *através da relação:*

$$ {}^{k+m}y_j^i = \sum_{l=0}^{m-1} \tan_{\Delta t}{}^{k+l} \alpha^i \cdot \Delta t + {}^k y_j^i \tag{14} $$

Demonstração: pela aplicação sucessiva do teorema **T4** tem-se que ${}^{k+1}y^i = \tan_{\Delta t}{}^k \alpha^i \cdot \Delta t + {}^k y^i$, ${}^{k+2}y^i = \tan_{\Delta t}{}^{k+1} \alpha^i \cdot \Delta t + {}^{k+1} y^i$, ... ,

$\mathbf{{}^{k+m}y^{i} = tan_{\Delta t}{}^{k+m-1}\alpha^{i}\cdot\Delta t + {}^{k+m-1}y^{i}}$. Somando-se todas essas expressões resulta imediatamente que: $\mathbf{{}^{k+m}y^{i} = \sum_{l=0}^{m-1} tan_{\Delta t}{}^{k+l}\alpha^{i}\cdot\Delta t + {}^{k}y^{i}}$.□

Corolário 2 (C2)- *Para o sistema de equações diferenciais* $\mathbf{\dot{y}^{i} = f(y^{i})}$ *é válida a relação* $\mathbf{tan_{m\cdot\Delta t}{}^{k}\alpha_{j}^{i} = \frac{1}{m}\cdot\sum_{l=0}^{m-1} tan_{\Delta t}{}^{k+l}\alpha_{j}^{i}}$ *para* ***j=1, 2, ..., n****.*

Demonstração: do teorema **T4** resulta de imediato que,

$$\mathbf{{}^{k+m}y_{j}^{i} = \left(tan_{m\cdot\Delta t}{}^{k}\alpha_{j}^{i}\right)\cdot m\cdot\Delta t + {}^{k}y_{j}^{i}} \tag{15}$$

Do corolário **C1** tem-se,

$$\mathbf{{}^{k+m}y_{j}^{i} = \sum_{l=0}^{m-1} tan_{\Delta t}{}^{k+l}\alpha_{j}^{i}\cdot\Delta t + {}^{k}y_{j}^{i}} \tag{16}$$

Somando-se as equações resulta que,

$$\mathbf{tan_{m\cdot\Delta t}{}^{k}\alpha_{j}^{i} = \frac{1}{m}\cdot\sum_{l=0}^{m-1} tan_{\Delta t}{}^{k+l}\alpha_{j}^{i}}\ \square \tag{17}$$

Note-se que se para o caso em que o sistema (6.a) é autônomo, $\mathbf{y^{i1}(t_1) = y^{i2}(t_2)}$ para $\mathbf{i_1 \neq i_2}$ e $\mathbf{t_1 \neq t_2}$ implica que, $\mathbf{\dot{y}^{i1}(t_1) = \dot{y}^{i2}(t_2)}$. De fato, de (6.a) tem-se que $\mathbf{\dot{y}_{j}^{i} = f(y^{i})}$. Logo,

$$\mathbf{\dot{y}^{i1}(t_1) = f[y^{i1}(t_1)]} \tag{18.a}$$

$$\mathbf{\dot{y}^{i2}(t_2) = f[y^{i2}(t_2)]} \tag{18.b}$$

Como, por hipótese, $\mathbf{y^{i1}(t_1) = y^{i2}(t_2)}$ resulta de (18.a) e (18.b) que $\mathbf{\dot{y}^{i1}(t_1) = \dot{y}^{i2}(t_2)}$. Esta propriedade estabelece que duas curvas, que partem de duas condições iniciais distintas $\mathbf{y^{i1}(t_o)}$ e $\mathbf{y^{i2}(t_o)}$ para $\mathbf{i_1 \neq i_2}$, da família de soluções do sistema $\mathbf{\dot{y} = f(y)}$ possuirão derivadas iguais se $\mathbf{y^{i1}(t_1) = y^{i2}(t_2)}$ mesmo quando $\mathbf{t_1 \neq t_2}$, ou seja, o sistema é autônomo.

A **P1** estabelece que se $\mathbf{y^{i1}(t_1) = y^{i2}(t_2)}$ então, $\mathbf{y^{i1}(t)}$ e $\mathbf{y^{i2}(t)}$ estão na mesma órbita do plano de fase, e portanto, são soluções apenas defasadas no tempo de $(\mathbf{t_2 - t_1})$. Observe que se $\mathbf{\dot{y}^{i1}(t_1) = \dot{y}^{i2}(t_2)}$ então, $\mathbf{f[y^{i1}(t_1)] = f[y^{i}{}_2(t_2)]}$. Neste último caso, $\mathbf{y^{i1}(t_1) = y^{i2}(t_2)}$ somente se existir a inversa $\mathbf{f^{-1}(.)}$ de $\mathbf{f(.)}$ e ela for única. A questão que resta agora é saber se a derivada média $\mathbf{\tan_{\Delta t}{}^{k}\alpha^{i}}$ de $\mathbf{{}^{k}y^{i}}$ e $\mathbf{{}^{k+1}y^{i}}$ também é autônoma, ou seja, invariante no tempo? Para responder a esta questão a propriedade apresentada a seguir será de bastante utilidade.

Propriedade 8 (P8) - *Se* $\mathbf{y^{i1}(t)}$ *e* $\mathbf{y^{i2}(t)}$ *são soluções de* $\mathbf{\dot{y} = f(y)}$ *que partem, respectivamente, de* $\mathbf{y^{i1}(t_o = 0)}$ *e* $\mathbf{y^{i2}(t_o = 0)}$ *e se* $\mathbf{y^{i1}(t_o = 0) = y^{i2}(T)}$ *para* $T>0$ *então,* $\mathbf{y^{i1}(\Delta t) = y^{i2}(T + \Delta t)}$ *para qualquer* $\Delta \mathbf{t}$.

Demonstração: Se $\mathbf{y^{i2}(t)}$ é uma solução de $\mathbf{\dot{y} = f(y)}$ então, $\mathbf{y^{i2}(t + T)}$ também será solução pela **P1**. Observe que se $\mathbf{y^{i1}(0) = y^{i2}(T)}$ então, $\mathbf{y^{i1}(t) = y^{i2}(t + T)}$ para $\mathbf{t}$=0. Como $\mathbf{y^{i1}(0)}$ e $\mathbf{y^{i2}(T)}$ são soluções, respectivamente, para $\mathbf{y^{i1}(t)}$ e $\mathbf{y^{i2}(t + T)}$ em $\mathbf{t}$=0, então o teorema **T1** garante que elas serão únicas para todo $\mathbf{t}$. Logo, $\mathbf{y^{i1}(t) = y^{i2}(t + T)}$, e, em particular, se $\mathbf{t = \Delta t}$, tem-se que $\mathbf{y^{i1}(\Delta t) = y^{i2}(\Delta t + T)}$. A **P1** garante que se $\mathbf{y^{i1}(0) = y^{i2}(T)}$ então, $\mathbf{y^{i1}(t) = y^{i2}(t + T)}$ estão na mesma órbita do plano de fase para qualquer $\mathbf{t}$. Na verdade a equação $\mathbf{y^{i1}(t) = y^{i2}(t + T)}$ determina que as soluções $\mathbf{y^{i1}(t)}$ e $\mathbf{y^{i2}(t + T)}$ estão apenas defasadas no tempo de $\mathbf{T}>0$.□

Propriedade 9 (P9) - *Se* $\mathbf{y^{i1}(t_1) = y^{i2}(t_2)}$ *para* $\mathbf{i_1 \neq i_2}$ *e* $\mathbf{t_1 \neq t_2}$ *então,* $\mathbf{\tan_{\Delta t}\alpha^{i1}(t_1) = \tan_{\Delta t}\alpha^{i2}(t_2)}$ *para* $\mathbf{\Delta t > 0}$*, ou seja,* $\mathbf{\tan_{\Delta t}{}^{k}\alpha^{i1}}$ *é invariante no tempo*.

Demonstração: Por definição, $\mathbf{\tan_{\Delta t}\alpha^{i1}(t_1) = \dfrac{y^{i1}(t_1 + \Delta t) - y^{i1}(t_1)}{\Delta t}}$ e $\mathbf{\tan_{\Delta t}\alpha^{i2}(t_2) = \dfrac{y^{i2}(t_2 + \Delta t) - y^{i2}(t_2)}{\Delta t}}$. Como, por hipótese $\mathbf{y_1^{i1}(t_1) = y_2^{i2}(t_2)}$, então $\mathbf{y_1^{i1}(t_1 + \Delta t) = y_2^{i2}(t_2 + \Delta t)}$ e portanto, $\mathbf{\tan_{\Delta t}\alpha^{i1}(t_1) = \tan_{\Delta t}\alpha^{i2}(t_2)}$.□

A **P9** estabelece que $\mathbf{\tan_{\Delta t}{}^{k}\alpha^{i}}$ também é autônoma. Este resultado é bastante útil pois ele determina que *basta conhecer* os valores das $\mathbf{\tan_{\Delta t}{}^{k}\alpha^{i}}$ para $\mathbf{i = 1, 2, ..., \infty}$ *no instante inicial* $\mathbf{t_o}$ para uma região particular de interesse $\mathbf{[y_j^{min}, y_j^{max}]^n}$ para **j=1, 2, ..., n**, pois nos demais instantes $\mathbf{t > t_o}$ elas *se repetirão se a solução do sistema dinâmico não ultrapassar as fronteiras do intervalo n-dimensional das variáveis de estado* $\mathbf{[y_j^{min}, y_j^{max}]^n}$ para **j=1, 2, ..., n** especificadas em $\mathbf{t_o}$. A Figura 5.6 ilustra o raciocínio apresentado neste parágrafo.

Ainda deve ser observado que o sistema dinâmico quando propagado sempre para frente terá o ângulo $\mathbf{{}^{k}\alpha(i)}$ variando somente no intervalo $\mathbf{-\dfrac{\pi}{2} < {}^{k}\alpha(i) < \dfrac{\pi}{2}}$, e portanto, também será único. Com base na mesma observação, quando o sistema dinâmico for propagado somente para trás, então $\mathbf{\dfrac{\pi}{2} < {}^{k}\alpha(i) < \dfrac{3 \cdot \pi}{4}}$, e portanto, o ângulo $\mathbf{{}^{k}\alpha(i)}$ também será único.

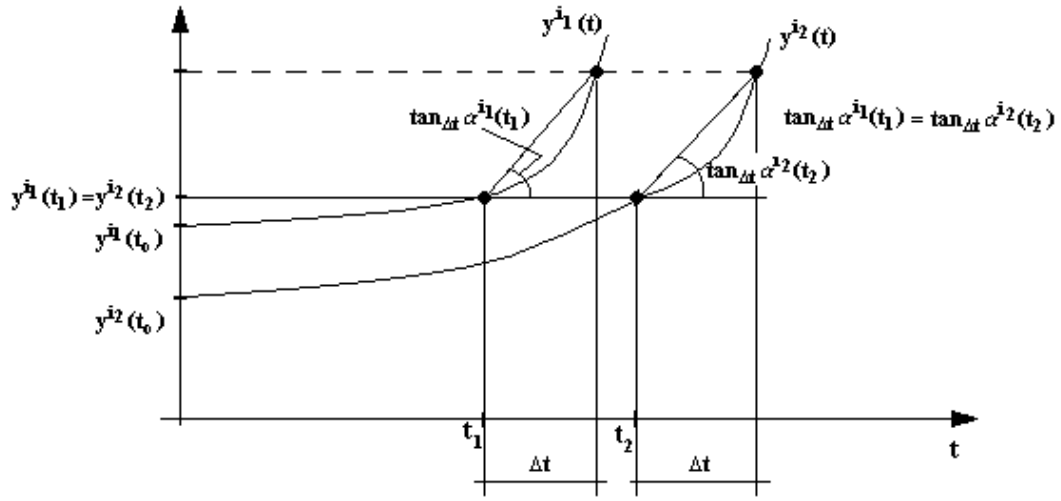


FIGURA 5.6 – A função discreta $\tan_{\Delta t}{}^{k}\alpha^{i}$ é autônoma, ou seja, invariante no tempo.

Teorema 5 (T5) *O resultado do **T4** continua válido quando valores discretizados de controle* $^{k}\mathbf{u}$ *em cada* $[\mathbf{t_k}, \mathbf{t_{k+1}}]$ *são utilizados para se para resolver o sistema dinâmico*:

$$\dot{\mathbf{y}}^{\mathbf{i}} = \mathbf{f}(\mathbf{y}^{\mathbf{i}}, \mathbf{u}) \tag{19}$$

Demonstração: De fato, basta notar que neste caso a função contínua $\mathbf{f}(\mathbf{y}^{\mathbf{i}}, \mathbf{u})$, com $^{k}\mathbf{u}$ aproximado como constante em cada $[\mathbf{t_k}, \mathbf{t_{k+1}}]$, pode ser vista como parametrizada em relação a variável de controle e, portanto, pode-se garantir em qualquer que seja o intervalo de discretização, a existência da derivada média $\dot{\mathbf{y}}^{\mathbf{i}}(\mathbf{t}_{\mathbf{k}}^{*}) = \dfrac{^{k+1}\mathbf{y}^{\mathbf{i}} - {}^{k}\mathbf{y}^{\mathbf{i}}}{\Delta \mathbf{t}} = \tan_{\Delta t}{}^{k}\alpha^{i}$ de modo que o resultado em (13) continua válido.□

5.5 Algoritmo do Método das Derivadas Médias Para Representar Sistemas Dinâmicos Através de uma Rede Neural *Feedforward.*

Os resultados teóricos da seção anterior serão utilizados aqui para fundamentar a representação de uma estrutura de integração do tipo Euler, com uma rede neural *feedforward* no papel das funções de derivadas médias em sua estrutura interna, para representar um sistema dinâmico não-linear de primeira ordem com uma precisão de resposta tão boa quanto qualquer integrador de mais alta ordem consegue obter. A representação de sistemas dinâmicos, como será descrita aqui, apresenta duas vantagens: a primeira é que ela consegue representar a estrutura interna do integrador de uma maneira mais simples facilitando e acelerando a fase de simulação, posterior ao treinamento, do sistema dinâmico não-linear representado pela rede; e a segunda vantagem é que este esquema simplifica, consideravelmente, o cálculo analítico necessário para deduzir as expressões algébricas das derivadas parciais necessárias para implementação computacional de uma estrutura de controle sobre a dinâmica não-linear treinada por este método.

Para facilitar a compreensão da metodologia, a ser apresentada aqui, a Figura 5.7 ilustra esquematicamente a estrutura de integração neural proposta. Como pode ser observado, a rede neural é treinada para aprender a função de derivadas médias, ou seja, $\mathbf{tan}_{\Delta t}{}^{k}\alpha^{i} \equiv \mathbf{tan}_{\Delta t}\alpha^{i}({}^{k}\mathbf{y}, {}^{k}\mathbf{u})$ a partir dos valores de estado ${}^{k}\mathbf{y}^{i}$, de controle ${}^{k}\mathbf{u}$ e de um intervalo de discretização pré fixado, $\Delta \mathbf{t}$. Neste diagrama ${}^{k+1}\mathbf{y}^{i}$ é o valor padrão de treinamento obtido *off line* pelo integrador numérico de elevada precisão, utilizado para resolver o sistema $\dot{\mathbf{y}}^{i} = \mathbf{f}(\mathbf{y}^{i}, \mathbf{u})$ e ${}^{k+1}\hat{\mathbf{y}}^{i}$ é o valor que a rede tenta estimar para ${}^{k+1}\mathbf{y}^{i}$ através de um treinamento supervisionado. Ainda com relação ao diagrama da Figura 5.7, a relação de recorrência entre ${}^{k}\mathbf{y}^{i}$ e ${}^{k+1}\mathbf{y}^{i}$ é expressa por ${}^{k+1}\mathbf{y}^{i} = \mathbf{tan}_{\Delta t}{}^{k}\alpha^{i} \cdot \Delta \mathbf{t} + {}^{k}\mathbf{y}^{i}$, onde $\mathbf{tan}_{\Delta t}{}^{k}\boldsymbol{\alpha}^{i} \cong \hat{\mathbf{f}}({}^{k}\mathbf{y}^{i}, {}^{k}\mathbf{u}, \hat{\mathbf{w}})$ é a derivada média a ser aproximada pela rede. É importante perceber que se ${}^{k+1}\mathbf{y}^{i}$ é obtido por uma integração do tipo Euler, então $\mathbf{tan}_{\Delta t}{}^{k}\alpha^{i}$ realmente convergirá para a função de derivadas

$\dot{y}^i = f(y^i, u)$, mas se ${}^{k+1}y^i$ for obtido por um integrador de mais alta ordem ou experimentalmente, então $\tan_{\Delta t}{}^k\alpha^i$ convergirá para a função de derivadas médias.

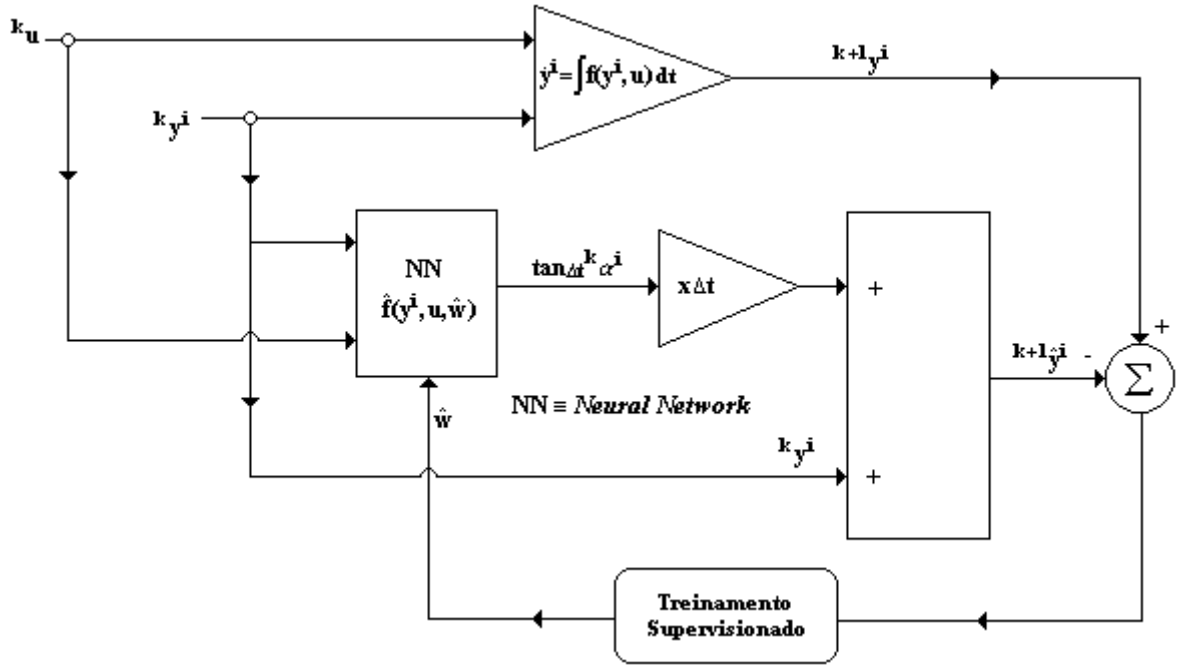


FIGURA 5.7 – Treinamento supervisionado de uma estrutura de integração do tipo Euler com uma rede *feedforward* em sua estrutura interna que aprende a função de derivadas médias do sistema dinâmico não-linear $\dot{y}^i = f(y^i, u)$.

A Figura 5.8 ilustra o que foi explicado no parágrafo anterior. Neste gráfico ${}^{k+1}y^i$, ${}^{k+1}y_a^i$ e ${}^{k+1}y_e^i$ representam, respectivamente, o valor exato da solução de $\dot{y}^i = f(y^i, u)$ em t_{k+1}, o valor aproximado de ${}^{k+1}y^i$ obtido por um integrador de alta ordem e o valor de ${}^{k+1}y^i$ obtido pelo integrador Euler. Como está esquematizado na Figura 5.8 tem-se que: se for feito ${}^{k+1}y^i = {}^{k+1}y_e^i$, no esquema da Figura 5.7, então, $\tan_{\Delta t}{}^k\alpha^i = \hat{f}({}^k y^i, {}^k u, \hat{w}) \cong f({}^k y^i, {}^k u)$, mas se se fizer ${}^{k+1}y^i = {}^{k+1}y_a^i$, no esquema da Figura 5.7, e ${}^{k+1}y_e^i$ estiver muito distante de ${}^{k+1}y^i$ então, $\tan_{\Delta t}{}^k\alpha^i$ tenderá para a

função de derivadas médias e não para a função de derivadas $\mathbf{f(y^i,u)}$ durante a fase de treinamento da rede.

Da seção anterior demonstrou-se matematicamente que a grandeza $\mathbf{tan}_{\Delta t}^{\mathbf{k}}\alpha^{\mathbf{i}}$, que será interpolada pela rede neural esquematizada na Figura 5.7, apresenta as seguintes características:

1) $\mathbf{tan}_{\Delta t}^{\mathbf{k}}\alpha^{\mathbf{i}} = \dfrac{{}^{\mathbf{k+1}}\mathbf{y}^{\mathbf{i}} - {}^{\mathbf{k}}\mathbf{y}^{\mathbf{i}}}{\Delta \mathbf{t}}$ é, por definição, a derivada média do intervalo $\mathbf{[t_k, t_{k+1}]}$ da curva $\mathbf{y^i(t)}$ que é solução do sistema dinâmico $\dot{\mathbf{y}}^{\mathbf{i}} = \mathbf{f(y^i,u)}$;

2) de **T1**, **P1**, **P2**, **P3** e **P4** têm-se que os valores das $\mathbf{tan}_{\Delta t}^{\mathbf{k+l}}\alpha^{\mathbf{i}}$ para **l=0, 1, ..., (L-1)** existem e são únicos, portanto $\mathbf{tan}_{\Delta t}^{\mathbf{k+1}}\alpha^{\mathbf{i}}$ é uma função estática com as mesmas propriedades qualitativas da função de derivadas ${}^{\mathbf{k+l}}\dot{\mathbf{y}}^{\mathbf{i}} = \mathbf{f}({}^{\mathbf{k+l}}\mathbf{y}^{\mathbf{i}}, {}^{\mathbf{k+l}}\mathbf{u})$. Na verdade, pode-se demonstrar que $\lim_{\Delta t \to 0} \mathbf{tan}_{\Delta t}^{\mathbf{k+l}}\mathbf{y}^{\mathbf{i}} = \mathbf{f}({}^{\mathbf{k+l}}\mathbf{y}^{\mathbf{i}}, {}^{\mathbf{k+l}}\mathbf{u})$. É importante perceber também que a função de derivadas $\dot{\mathbf{y}}^{\mathbf{i}} = \mathbf{f(y^i,u)}$ não depende de $\Delta \mathbf{t}$, mas as $\mathbf{tan}_{\Delta t}^{\mathbf{k+1}}\alpha^{\mathbf{i}}$ dependem. Esta última propriedade implica que o método das derivadas médias possui passo de integração fixo enquanto o método das derivadas instantâneas, proposto primeiramente por Wang e Lin, (1998b), pode possuir passo de integração variável;

3) do teorema **T2** $\mathbf{tan}_{\Delta t}^{\mathbf{k}}\alpha^{\mathbf{i}}$ é realmente a derivada exata de pelo menos um ponto interior ao intervalo $\mathbf{[t_k, t_{k+1}]}$ que é também, outra forma de garantir a existência de $\mathbf{tan}_{\Delta t}^{\mathbf{k}}\alpha^{\mathbf{i}}$;

4) dos teoremas **T3** e **T4**, propriedade **P6** segue que a relação de recorrência relacionando ${}^{\mathbf{k+1}}\mathbf{y}^{\mathbf{i}}$ com ${}^{\mathbf{k}}\mathbf{y}^{\mathbf{i}}$ e ${}^{\mathbf{k}}\mathbf{u}$ para a obtenção de uma solução discretizada para o sistema dinâmico $\dot{\mathbf{y}}^{\mathbf{i}} = \mathbf{f(y^i,u)}$ é realmente dada por ${}^{\mathbf{k+1}}\mathbf{y}^{\mathbf{i}} = \mathbf{tan}_{\Delta t}^{\mathbf{k}}\alpha^{\mathbf{i}} \cdot \Delta \mathbf{t} + {}^{\mathbf{k}}\mathbf{y}^{\mathbf{i}}$ que é uma estrutura de integração simples do tipo Euler;

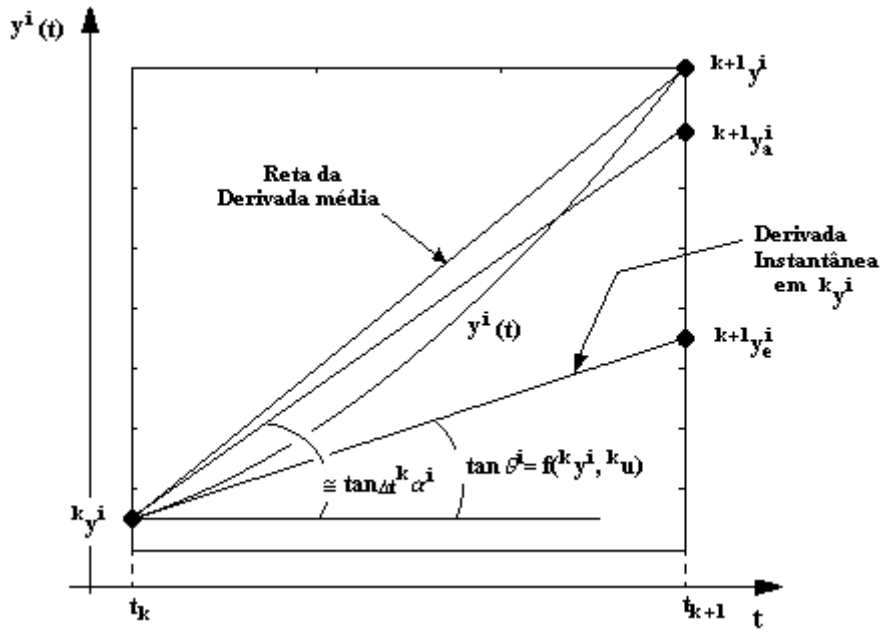


FIGURA 5.8 – Representação gráfica das derivadas médias $\tan_{\Delta t}{}^k\alpha^i$ e da função de derivadas $\tan^k\theta^i = f(^k y^i, ^k u)$ para o sistema dinâmico $\dot{y}^i = f(y^i, u)$.

5) do teorema **T5,** tanto a função de derivadas $f(^k y^i, ^k u)$ como a função de derivadas médias $\tan_{\Delta t}{}^k\alpha^i$ é invariante no tempo, porém parametrizada em relação a $^k u$.

Das cinco características, resumidas anteriormente, pode-se garantir a possibilidade de se usar uma rede neural *feedforward* para representação de sistemas dinâmicos através de uma estrutura de integração do tipo Euler, para um dado passo Δt. Basta considerar a capacidade de representação de funções destas redes (e.g., Zurada, 1992) para concluir sobre a possibilidade de controlar o erro local no processo de integração neural de Euler. O algoritmo para obter o integrador neural baseado em derivadas médias então, seria:

1) dados os domínios finitos de interesse $[y_j^{min}(t_o), y_j^{max}(t_o)]^{n1}$ em t_o para **j=1, 2, ..., n_1** das variáveis de estado e os domínios também finitos de interesse $[u_j^{min}, u_j^{max}]^{n2}$ para **j=1, 2, ..., n_2** das variáveis de controle geram-se **m** vetores

aleatórios, segundo uma distribuição uniforme, dentro destes intervalos na forma

$$\mathbf{p_i} = [\mathbf{y_1^i(t_o)}\ \mathbf{y_2^i(t_o)}\ \cdots \mathbf{y_{n_1}^i(t_o)};\ \mathbf{u_1^i(t_o)}\,\mathbf{u_2^i(t_o)}\cdots \mathbf{u_{n_2}^i(t_o)}]^{\mathbf{T}} \tag{20.a}$$

e

$$\mathbf{P} = [\,\mathbf{p_1 : p_2 : \cdots : p_m}\,]_{(\mathbf{n_1+n_2})\ \mathbf{x}\ \mathbf{m}} \tag{20.b}$$

que serão os vetores de entrada ou padrões de treinamento da entrada da rede *feedforward* no instante $\mathbf{t_o}$;

2) utilizando-se um integrador de alta ordem propagam-se todas as condições iniciais $\mathbf{p_i}$ para **i=1, 2, ..., m** com um passo de integração *fixo* de $\Delta\mathbf{t}$ para todas as condições iniciais $\mathbf{p_i}$. Desta forma, geram-se os seguintes vetores de estado no instante $\mathbf{t_o} + \Delta\mathbf{t}$:

$$\mathbf{p^{\Delta t}_i} = [\mathbf{y_1^i(t_o} + \Delta\mathbf{t)}\ \ \mathbf{y_2^i(t_o} + \Delta\mathbf{t)}\ \cdots \mathbf{y_{n_1}^{\ i}(t_o} + \Delta\mathbf{t)}\,]^{\mathbf{T}} \tag{21.a}$$

e

$$\mathbf{P^{\Delta t}} = [\,\mathbf{p^{\Delta t}_1 : p^{\Delta t}_2 : \cdots : p^{\Delta t}_m}\,]_{\mathbf{n_1\ x\ m}} \tag{21.b}$$

3) determinam-se os vetores de saída $\mathbf{T_i}$ ou padrões de treinamento de saída da rede neural na forma:

$$\mathbf{T_i} = \frac{\mathbf{1}}{\Delta\mathbf{t}} \cdot [\mathbf{y_1^i(t_o} + \Delta\mathbf{t)} \mathbf{-y_1^i(t_o)}\ \ \mathbf{y_2^i(t_o} + \Delta\mathbf{t)} \mathbf{-y_2^i(t_o)}\ \ \cdots\ \ \mathbf{y_{n_1}^i(t_o} + \Delta\mathbf{t)} \mathbf{-y_{n_1}^{\ i}(t_o)}]^{\mathbf{T}}$$

$$= [\mathbf{tan}_{\Delta\mathbf{t}}^{\mathbf{k}}\,\alpha_1^{\mathbf{i}}\ \ \mathbf{tan}_{\Delta\mathbf{t}}^{\mathbf{k}}\,\alpha_2^{\mathbf{i}}\ \cdots\ \mathbf{tan}_{\Delta\mathbf{t}}^{\mathbf{k}}\,\alpha_{\mathbf{n_1}}^{\mathbf{i}}\,]^{\mathbf{T}} = \left(\mathbf{tan}_{\Delta\mathbf{t}}^{\mathbf{k}}\,\alpha^{\mathbf{i}}\right)^{\mathbf{T}} \tag{22.a}$$

e

$$\mathbf{T} = [\mathbf{T_1 : T_2 : \cdots : T_m}\,]_{\mathbf{n_1\ x\ m}} \tag{22.b}$$

Como a função $\mathbf{tan}_{\Delta t}^{k}\alpha^{i}$ também é autônoma, então basta apenas uma propagação de Δt sobre todas as condições iniciais dos vetores de entrada $\mathbf{p_i}$ para **i=1, 2, ..., m** para montar os padrões de treinamento **P** e **T** da rede neural;

4) tendo-se os vetores de entrada **P** e os vetores de saída **T** da rede *feedforward* ela poderá, então, ser treinada através de um aprendizado supervisionado utilizando-se os algoritmos convencionais de retro-propagação;

5) após o treinamento da rede ter sido concluído com um erro quadrático médio **meq** desejado, então poder-se-á simular a dinâmica a partir da seguinte relação:

$$^{k+1}\mathbf{y}^{i} = \mathbf{tan}_{\Delta t}^{k}\alpha^{i}\cdot\Delta t + {}^{k}\mathbf{y}^{i} \tag{23}$$

Note-se que ao adotar os padrões de treinamento $\mathbf{T_i}$ na saída da rede e projeta-la na forma $\mathbf{T_i} = \mathbf{tan}_{\Delta t}^{k}\alpha^{i}$ evita-se de calcular a retro-propagação sobre a expressão $^{k+1}\mathbf{y}^{i} = \mathbf{tan}_{\Delta t}^{k}\alpha^{i}\cdot\Delta t + {}^{k}\mathbf{y}^{i}$. Observe que aplicar a retro-propagação sobre esta última expressão conduz a uma expressão ligeiramente diferente da encontrada na literatura, pois se $^{S+1}\mathbf{y}^{a} = \mathbf{tan}_{\Delta t}^{S}\alpha^{a}\cdot\Delta t + {}^{S}\mathbf{y}^{a} = \hat{\mathbf{f}}({}^{S}\mathbf{y}^{a}, {}^{S}\mathbf{u}, \hat{\mathbf{w}})\Delta t + {}^{S}\mathbf{y}^{a}$ e se deseja minimizar o funcional $\mathbf{J(t)} = \frac{1}{2}\cdot\mathbf{r}^{T}(\mathbf{t})\cdot\mathbf{r}(\mathbf{t},)$ para $\mathbf{r(t)} = {}^{S+1}\mathbf{y}^{a} - {}^{S+1}\bar{\mathbf{y}}^{a}$; **t=1, 2, ..., m,** onde **m** é o número total de padrões de treinamento, então tem-se que:

$$\frac{\partial \mathbf{J(t)}}{\partial\, \mathbf{w}_{jk}^{l}} = \mathbf{r(t,i)}\cdot\frac{\partial\, {}^{s+1}\mathbf{y}^{a}}{\partial \mathbf{w}_{jk}^{l}} = \underbrace{\left[\mathbf{r(t)}\cdot\frac{\partial \mathbf{f}({}^{s}\mathbf{y}^{a}, {}^{s}\mathbf{u}, \hat{\mathbf{w}})}{\partial \mathbf{w}_{jk}^{l}}\right]}_{\substack{\textbf{Retro-Propagação}\\ \textbf{Convencional}}}\cdot\Delta t \tag{24}$$

A equação (24) é a retro-propagação convencional acrescida pelo produto de Δt. Como se fez $\mathbf{T_i} = \mathbf{tan}_{\Delta t}^{k}\alpha^{i}$, que como demonstrado na seção anterior é uma função estática, então evitou-se de acrescentar o produto de Δt na retro-propagação. Esta observação é importante visto que já existem muitos *software* prontos de treinamento das redes *feedforward* que poderão ser aproveitados sem realizar nenhuma modificação do

algoritmo de treinamento original, mesmo porque, em muitos deles essa pequena modificação seria impossível por já estarem compilados.

A segunda observação é com relação ao fato do sistema de equações diferenciais ordinárias serem de ordem maior que um. Se o sistema for conhecido, então poder-se-á reduzi-lo ao de primeira ordem, e neste caso, recai-se no que já foi desenvolvido anteriormente. Entretanto, caso se conheça apenas um conjunto de vetores soluções de um sistema dinâmico cuja as equações diferenciais associadas aos vetores soluções sejam desconhecidas, então, neste caso, torna-se impossível conhecer a ordem do sistema dinâmico em questão, e portanto, introduzir entradas atrasadas na interpolação da função $\mathbf{tan}_{\Delta \mathbf{t}}{}^{\mathbf{k}}\alpha^{\mathbf{i}}$ poderá tornar-se necessário, caso a interpolação com apenas uma entrada atrasada fracasse na interpolação do sistema dinâmico original. Os corolários **C1** e **C2**, apresentados na seção anterior resolvem este problema, pois em **C2** tem-se que $\mathbf{tan}_{\mathbf{m}\cdot\Delta\mathbf{t}}{}^{\mathbf{k}}\alpha_{\mathbf{j}}^{\mathbf{i}} = \frac{\mathbf{1}}{\mathbf{m}} \cdot \sum_{\mathbf{l=0}}^{\mathbf{m-1}} \mathbf{tan}_{\Delta\mathbf{t}}{}^{\mathbf{k+l}}\alpha_{\mathbf{j}}^{\mathbf{i}}$ e quando substituída no corolário **C1** resulta que $^{\mathbf{k+m}}\mathbf{y}_{\mathbf{j}}^{\mathbf{i}} = \mathbf{tan}_{\mathbf{m}\cdot\Delta\mathbf{t}}{}^{\mathbf{k}}\alpha_{\mathbf{j}}^{\mathbf{i}} \cdot \mathbf{m} \cdot \Delta\mathbf{t} + {}^{\mathbf{k}}\mathbf{y}_{\mathbf{j}}^{\mathbf{i}}$. Estas duas equações podem, respectivamente, ser utilizadas no treinamento da rede neural com **m** entradas atrasadas e na simulação desta rede depois de concluído seu treinamento. As Figuras 5.9 e 5.10 ilustram gráfica e esquematicamente esta idéia.

Para finalizar esta seção resta saber qual a capacidade real de treinamento e aprendizagem da estrutura de integração neural apresentada e a influência do passo $\Delta\mathbf{t}$ neste aprendizado. Para responder a estas duas questões serão utilizados o teorema de Kolmogorov (e.g., Nascimento e Yoneyama, 2000) e a expressão analítica do erro local no processo de integração neural presente na estrutura de Euler da Figura 5.7.

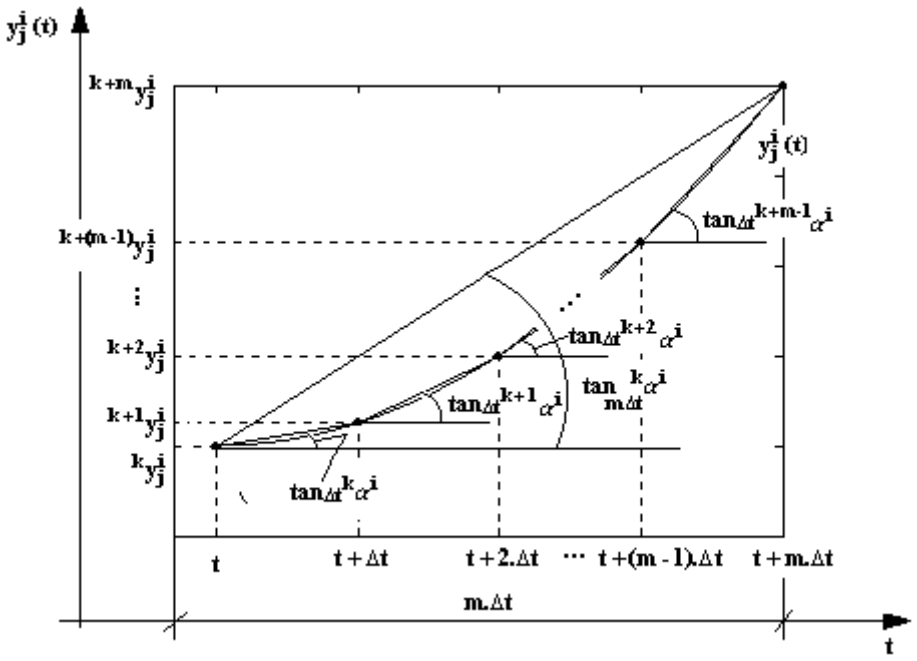


FIGURA 5.9 – Representação gráfica do conceito de entradas atrasadas aplicado à função $\tan_{m\Delta t}{}^{k}\alpha_j^i$.

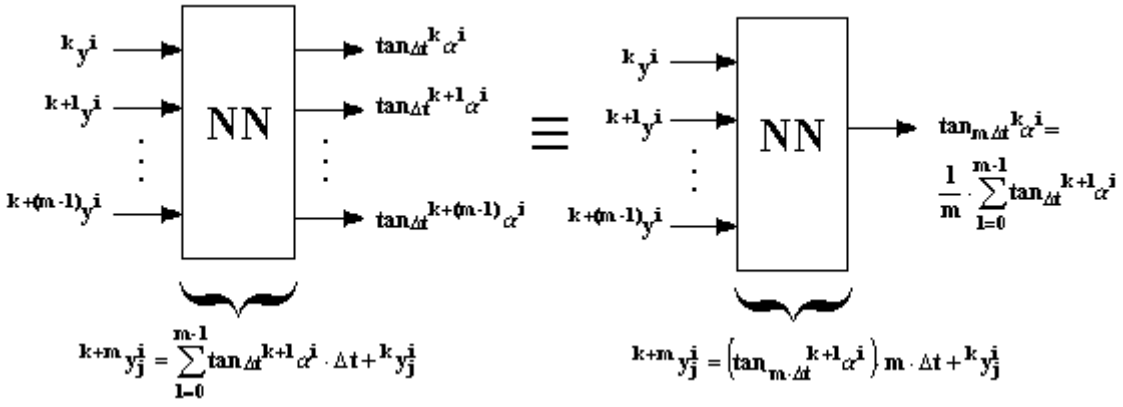


FIGURA 5.10 – Representação esquemática de uma rede neural com arquitetura *feedforward* projetada com m entradas atrasadas para representar à função $\tan_{m\Delta t}{}^{k}\alpha_j^i$.

Para analisar o erro de integração local da estrutura de integração neural do tipo Euler seja o valor exato $^{k+1}\mathbf{y}^{i}$ e o valor estimado $^{k+1}\hat{\mathbf{y}}^{i}$ pelo integrador dados, respectivamente, pelas equações (25.a) e (25.b).

$$^{k+1}\mathbf{y}^{i} = \mathbf{tan}_{\Delta t}{}^{k}\alpha^{i} \cdot \Delta t + {}^{k}\mathbf{y}^{i} \tag{25.a}$$

$$^{k+1}\hat{\mathbf{y}}^{i} \cong (\mathbf{tan}_{\Delta t}{}^{k}\alpha^{i} + \mathbf{e_m}) \cdot \Delta t + {}^{k}\mathbf{y}^{i} \tag{25.b}$$

onde,

$\mathbf{e_m}$ ... erro absoluto médio de saída da rede neural já treinada com a função de derivadas médias $\mathbf{tan}_{\Delta t}{}^{k}\alpha_{j}^{i}$ dentro de um domínio de interesse.

Subtraindo-se a equação (25.a) na equação (25.b) e elevando o resultado final ao quadrado, tem-se:

$$\left({}^{k+1}\mathbf{y}^{i} - {}^{k+1}\mathbf{y}^{i} \right)^{2} = \Delta t^{2} \cdot \mathbf{e_m}^{2} \tag{26}$$

No sentido do teorema de Kolmogorov (Loesch e Sari, 1996) entende-se por capacidade de aprendizagem de uma rede a existência de valores para os pesos de conexão que permitem aproximar o comportamento funcional da rede dos valores desejados, de modo que seu erro quadrático seja inferior a qualquer número positivo dado. Este teorema, provado em 1957, garante a representação universal de funções no espaço n-dimensional através da combinação linear de funções unidimensionais não-lineares. Seu enunciado, adaptado na década de 80 (e.g., Nascimento e Yoneyama, 2000) para o caso particular das redes neurais, garante que a equação (27) aproxima-se de qualquer precisão desejada uma vez que $\mathbf{e_m^2}$ poderá ser tão pequeno quanto se queira, para um passo $\Delta t > 0$ de integração fixo. Assim $^{k+1}\hat{\mathbf{y}}^{i}$ na equação (25.b) aproxima-se de qualquer precisão desejada uma vez que, para um passo $\Delta t > 0$ de integração fixo, $\mathbf{e_m}$ poderá ser tão pequeno quanto se queira, pois a rede neural *feedforward* está

aproximando, dentro de um domínio de interesse, a função de derivadas médias $\mathbf{tan}_{\Delta \mathbf{t}}^{\mathbf{k}} \alpha_{\mathbf{j}}^{\mathbf{i}}$ que é invariante no tempo.

Por fim, a equação (23) referente ao integrador neural das derivadas médias poderá então, ser utilizada como uma estrutura de modelo interno expressa, mais genericamente, pela equação (4) e que poderá ser aplicada numa estrutura de controle preditivo não-linear dada pela equação (5). Nestas condições, a única variável ainda a ser determinada é o Jacobiano das derivadas parciais dos estados futuros do sistema dinâmico em relação aos controles atrasados, exigido pela equação (5). Uma demonstração matemática rigorosa para as expressões analíticas dessas derivadas parciais no caso genérico de **n** horizontes a frente pode ser encontrada no Apêndice E. Estas expressões são também uma contribuição teórica original deste trabalho.

5.6 Comentários Finais.

Neste capítulo foi proposta uma nova abordagem para obter os modelos discretos para frente de sistemas dinâmicos onde um modelo interno é necessário. A possibilidade de se utilizar integradores numéricos de sistemas de equações diferenciais como modelos internos foi explorada. Foi mostrado que a estrutura destes integradores numéricos pode ser explorada para obter modelos neurais discretos para frente onde a rede neural tem que somente aprender e aproximar as funções de derivadas que são simplesmente de natureza algébrica estática. A seguir algumas conclusões (Rios Neto, 2001), podem ser tiradas quanto às expectativas de se utilizar tal abordagem:

1) é uma tarefa mais simples treinar uma rede neural com arquitetura *feedforward* para aprender uma função algébrica e estática do que treiná-la para aprender o modelo dinâmico discreto como explicado na seção 4.2 através da equação (2);
2) a arquitetura da rede neural resultante certamente será mais simples no que concerne ao número de camadas e número de neurônios uma vez que ela não terá que aprender a lei da dinâmica, mas somente a função de derivadas;

3) a utilização do integrador numérico para sistemas de equações diferenciais ordinárias, como um modelo aproximado de tempo discreto, não destrói a característica de processamento paralelo, uma vez que o algoritmo de integração numérica somente envolverá cálculos e avaliações de combinações lineares da rede neural treinada;
4) a flexibilidade de se poder variar o tamanho do passo e a ordem do integrador numérico pode ser utilizada para controlar a precisão desejada para a saída de resposta do sistema, desde que se treine a rede neural com uma tolerância compatível com aquela que se deseja no integrador;
5) quando os tempos de resposta da dinâmica não são muito pequenos e o modelo matemático de equações diferenciais ordinárias avaliado é razoavelmente bom, então a estrutura do integrador numérico pode ser diretamente utilizada como um modelo interno discreto;
6) mesmo em situações onde um modelo matemático teórico não pode ser avaliado e há somente pares de informação da dinâmica de entrada/saída do sistema obtidos experimentalmente, ainda assim, a estrutura do integrador numérico com uma rede *feedforward* no lugar da função de derivadas poderá ser treinada para obter um modelo interno discreto em esquemas de controle;
7) a utilização de uma rede neural em um modelo discreto do sistema dinâmico naturalmente permitirá a implementação de esquemas de controle adaptativos, devido a capacidade de aprendizado das redes neurais.

Existem, pois, três maneiras distintas de se resolver o problema de programação não-linear exigido pela estrutura de controle preditivo:

1) utilizar somente a rede neural para aproximar a dinâmica do sistema utilizando o critério de entradas atrasadas;
2) utilizar somente o integrador numérico como modelo discretizado da dinâmica do sistema;
3) utilizar em conjunto o integrador numérico e uma rede neural que fará o papel de representar as funções de derivadas para representar o sistema dinâmico.

Utilizar somente a rede neural levará a uma rede mais complexa e, portanto mais difícil de ser treinada, além de não possibilitar a variação do passo de integração. Utilizar somente a estrutura do integrador numérico com a função de derivadas não permitirá adaptar o modelo teórico do sistema dinâmico com o real da planta. Utilizar em conjunto a estrutura do integrador numérico com a função de derivadas instantâneas representada pela rede neural permitirá construir uma rede menos complexa e, portanto, de mais fácil treinamento, além de se poder projetar, neste caso, um modelo dinâmico com *passo de integração variável* e que seja possível também adapta-lo em tempo real com relação ao sistema original da planta.

A maior desvantagem do último método é que o cálculo das derivadas parciais necessárias na solução do problema de controle, torna-se mais complexo e complicado, sendo que para cada tipo de integrador a ser utilizado ter-se-á uma expressão diferente para as derivadas parciais e, em geral, estas expressões serão tão mais complexas e difíceis de serem obtidas quanto maior a ordem do integrador. O método das derivadas médias, aplicado a estrutura de integração do tipo Euler, têm por objetivo principal simplificar o cálculo destas derivadas parciais, pois neste caso, evita-se a utilização de um integrador de alta ordem, sem, no entanto, prejudicar a precisão de resposta na estimação dos estados do sistema dinâmico. Porém, nesta última estrutura, perde-se de um lado, pois ela é inevitavelmente uma estrutura de integração de *passo fixo*.

CAPÍTULO 6

RESULTADOS E TESTES

6.1 Introdução.

Neste capítulo serão apresentados os resultados e testes numéricos obtidos pela estrutura de controle preditivo neural aplicada a dois problemas não-lineares distintos: o da dinâmica da transferência de órbita Terra/Marte de um foguete e o do controle da atitude de um satélite artificial, apresentados, respectivamente, nas seções 6.2 e 6.3.

Com relação à representação da dinâmica destes problemas pela rede neural, três métodos distintos são considerados: o método NARMAX ou das entradas atrasadas, o método das derivadas instantâneas e o método das derivadas médias ou de Euler. No problema de transferência de órbitas Terra/Marte os três métodos são expostos, simultaneamente, para uma análise da eficiência apresentada por cada um deles. No problema do controle da atitude de satélites artificiais somente a dinâmica neural representada pelo método das derivadas médias é apresentada.

6.2 O Problema da Transferência de Órbitas Terra/Marte.

6.2.1 Dinâmica de Transferência de Órbita.

Os resultados práticos para o problema de transferência de órbita Terra/Marte, como ilustra a Figura 6.1, serão apresentados nesta seção. As variáveis de estado deste problema são: a massa do foguete **m**, o raio da órbita **r**, a velocidade radial **w** e a velocidade transversal **v**. A única variável de controle é o ângulo guiado do empuxo θ. As equações de *estado* (Sage, 1968) da dinâmica da transferência de órbita Terra/Marte do foguete de massa **m** são:

$$\dot{m} = -0.0749 \tag{1.a}$$

$$\dot{r} = w \tag{1.b}$$

$$\dot{w} = \frac{v^2}{r} - \frac{\mu}{r^2} + \frac{T \cdot \sin\theta}{m} \tag{1.c}$$

$$\dot{v} = \frac{-w \cdot v}{r} + \frac{T \cdot \cos\theta}{m} \tag{1.d}$$

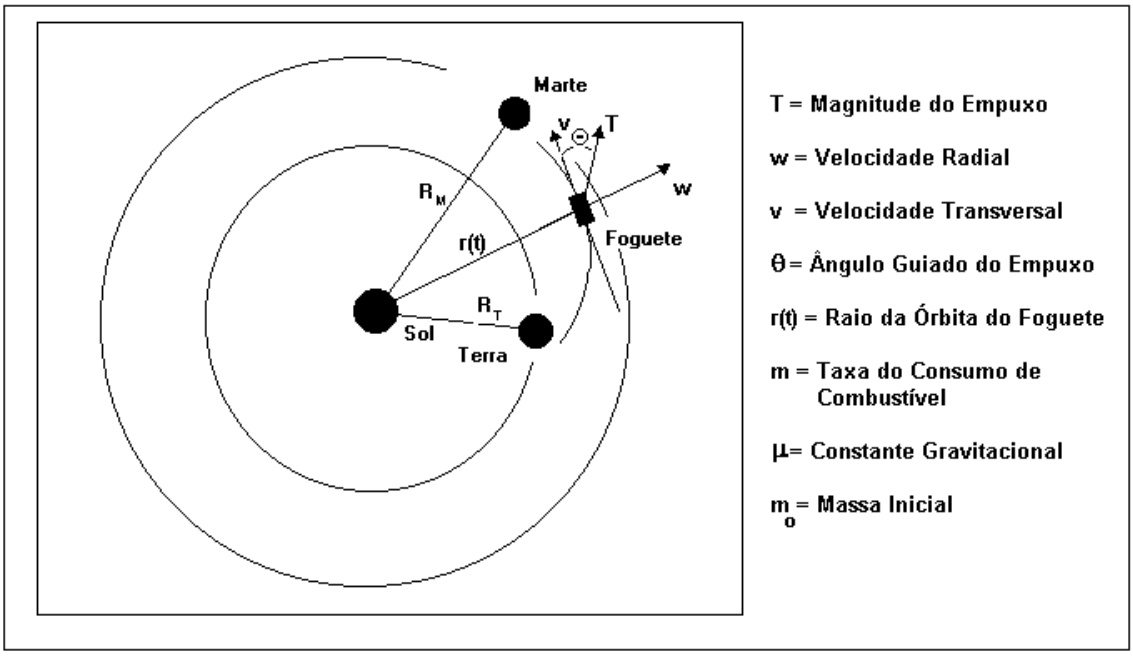


FIGURA 6.1– Esquema ilustrativo da transferência de órbita Terra/Marte.

As constantes normalizadas deste problema são: uma unidade de tempo é igual a 58.2 dias, uma unidade de comprimento é igual a distância média da Terra ao Sol (unidade astronômica), μ=1.0 (constante gravitacional), **T**=0.1405 (empuxo do foguete).

6.2.2 Resultados dos Testes.

Uma solução numérica da *dinâmica* obtida pelo integrador neural que utilizou o método das derivadas instantâneas é apresentada na Figura 6.2, com o emprego de uma política de controle aleatória entre $-\pi$ e $+\pi$ para o controle θ. A curva contínua representa a solução numérica obtida pelo integrador Runge-Kutta de quarta ordem que utilizou a função de derivadas original, os círculos representam a solução numérica do Runge-Kutta de quarta ordem que utilizou a função de derivadas neural e os triângulos são as soluções obtidas pelo integrador de Adams-Bashforth de quarta ordem que também utilizou a função de derivas neural. Neste gráfico é importante observar que há uma atualização das condições iniciais, em relação ao modelo teórico original, a cada 100 passos de integração e este passo foi adotado e igual a 0.01. A atualização das condições iniciais a cada 100 passos de integração tem a função de simular o desempenho do sistema dinâmico, em questão, em malha fechada, pois todo sistema de controle aplicado em tempo real tem suas condições iniciais atualizadas pelo sistema de navegação através da atuação de seus sensores. Desta maneira, não é necessário simular o desempenho do sistema dinâmico neural em malha aberta, caso este, bem mais difícil que o anterior.

Na Figura 6.3 é apresentada à mesma simulação da Figura 6.2, mas com a diferença fundamental que as condições iniciais dos problemas não são atualizadas em relação aos valores originais.

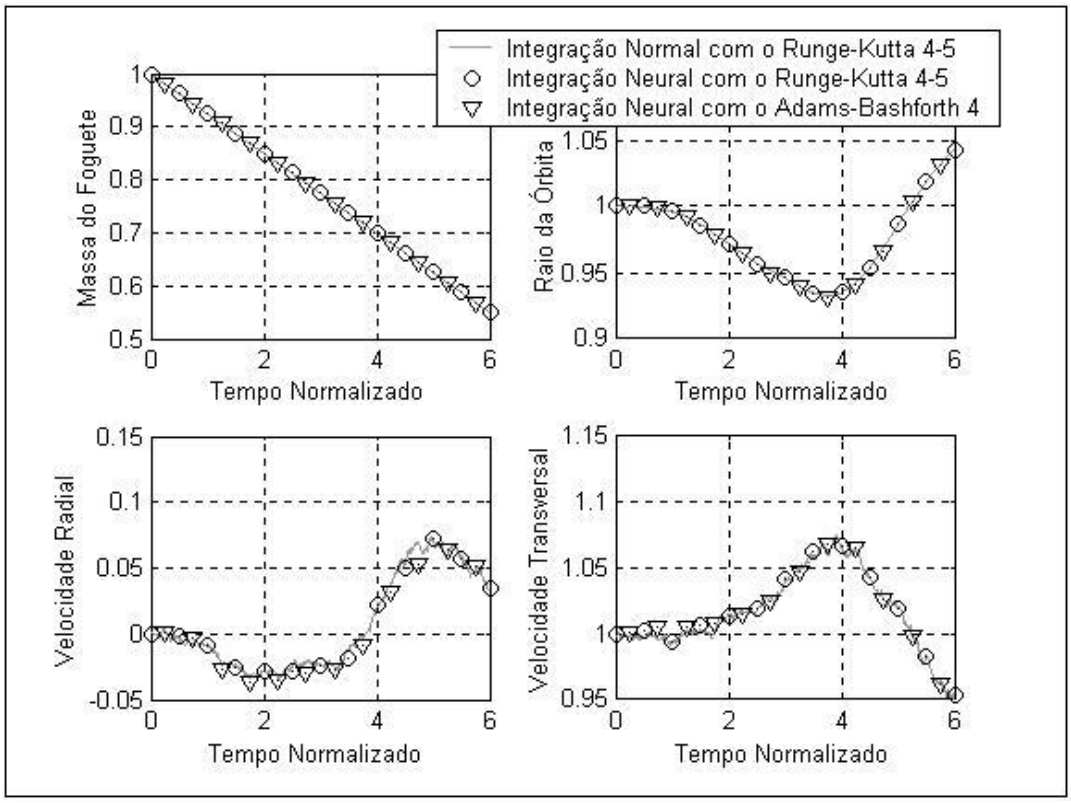


FIGURA 6.2 – Testes numéricos dos integradores neurais no problema da dinâmica de transferência de órbitas Terra/Marte com atualizações das condições iniciais após 100 propagações consecutivas (Δt=0.01).

Alguns dados importantes quanto a arquitetura e o desempenho da rede neural utilizada para obter os resultados numéricos das Figuras 6.2 e 6.3 são: rede com 5 entradas (4 variáveis de estados e 1 de controle), há somente uma camada interna com 41 neurônios *tansig* ($\lambda = 2$), rede com 4 saídas (funções de derivadas instantâneas), as camadas de entrada e interna possuem cada uma um neurônio extra de viés com entrada sempre constante e igual a –1, no treinamento neural foram utilizados 9000 vetores de entrada/saída durante o treinamento e 1000 vetores de entrada/saída durante os testes, a média dos erros quadráticos de treinamento e teste alcançados durante o treinamento foram respectivamente $3{,}3580.10^{-5}$ e $3{,}3565.10^{-5}$, a média das normas das diferenças entre os valores originais e neurais da Figura 6.2 é de $1{,}1219.10^{-2}$ e da Figura 6.3 é de $4{,}8765.10^{-2}$. Entende-se por média das normas das diferenças entre os valores originais e neurais o seguinte conjunto de expressões algébricas:

$$\mathbf{nd(t)} = \sqrt{\left[\mathbf{m(t)} - \hat{\mathbf{m}}\mathbf{(t)}\right]^2 + \left[\mathbf{r(t)} - \hat{\mathbf{r}}\mathbf{(t)}\right]^2 + \left[\mathbf{w(t)} - \hat{\mathbf{w}}\mathbf{(t)}\right]^2 + \left[\mathbf{v(t)} - \hat{\mathbf{v}}\mathbf{(t)}\right]^2} \qquad (2.a)$$

para $\mathbf{t = t_o + \Delta t, t_o + 2 \cdot \Delta t, ..., t_f}$ e,

$$\mathbf{mnd} = \frac{\mathbf{1}}{\mathbf{n}} \cdot \sum_{\mathbf{j=1}}^{\mathbf{n}} \mathbf{nd(t_o + j \cdot \Delta t)} \text{ para } \mathbf{n} = \frac{\mathbf{t_f - t_o}}{\mathbf{\Delta t}} + \mathbf{1} \qquad (2.b)$$

onde de maneira genérica,

x(t) ... valor real no instante t, neste caso, obtido pelo Runge-Kutta de 4ª ordem;

$\hat{\mathbf{x}}\mathbf{(t)}$ ... valor estimado pela rede neural no instante t.

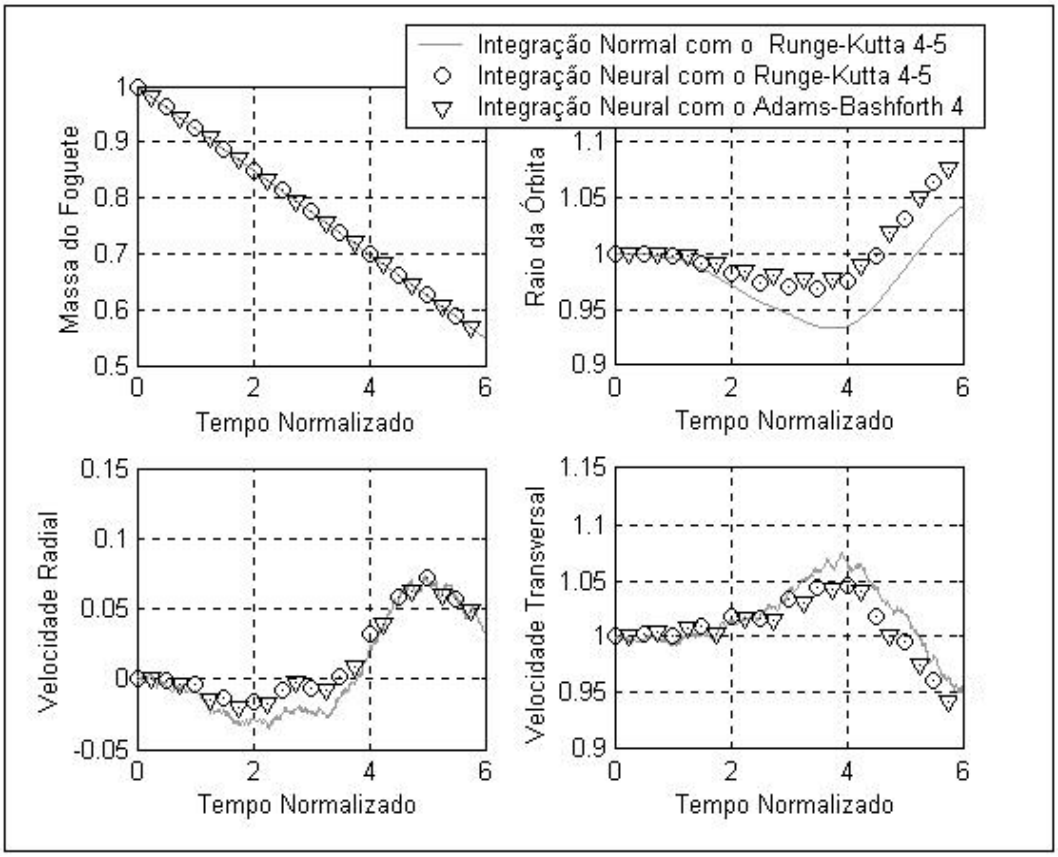


FIGURA 6.3 – Testes numéricos dos integradores neurais no problema da dinâmica de transferência de órbitas Terra/Marte sem atualizações das condições iniciais (Δt=0.01).

Para o caso bi-dimensional fica fácil entender o conceito físico de **nd(t)**. Por exemplo, seja um triângulo retângulo de catetos $\mathbf{e_1(t)}$ e $\mathbf{e_2(t)}$ e hipotenusa **nd(t)**. Neste caso, ter-se-ia que $\mathbf{nd(t) = \sqrt{e_1(t)^2 + e_2(t)^2}}$, ou seja, **nd(t)** é a estimativa do erro máximo admissível e válido para qualquer uma das variáveis $\mathbf{e_1(t)}$ e $\mathbf{e_2(t)}$, já que a hipotenusa de um triângulo retângulo é o maior dos lados. O valor **mnd** seria a média das normas obtidas para todos os instantes de propagação considerados.

Os vetores de normalização dos padrões de entrada e de saída da rede neural são respectivamente $\mathbf{vn_e}$=[1,0000 2,5998 1,0000 1,4999 7,8528] e $\mathbf{vn_s}$=[0,0749 1,0000 2,1103 1,8561]. Os domínios das variáveis de estado e de controle utilizados para o treinamento neural foram aqueles como apresentados pela Tabela 6.1.

Por outro lado, a Figura 6.4 apresenta a mesma simulação obtida pela Figura 6.3, mas utilizando o método das derivadas médias com o integrador de Euler ao invés do método das derivadas instantâneas. Nesta figura, não há atualização das condições iniciais e, portanto, apenas pela análise visual, percebe-se uma melhora significativa nos resultados obtidos.

TABELA 6.1 – Domínios das variáveis de estado e de controle utilizados no treinamento neural da função de derivadas instantâneas.

Função de Derivadas Instantâneas		
Variáveis	**Valor Mínimo**	**Valor Máximo**
M	**0,2**	**1,0**
R	**0,8**	**2,6**
W	**-1,0**	**+1,0**
V	**0,0**	**1,5**
θ	**-2,5. π**	**+2,5. π**

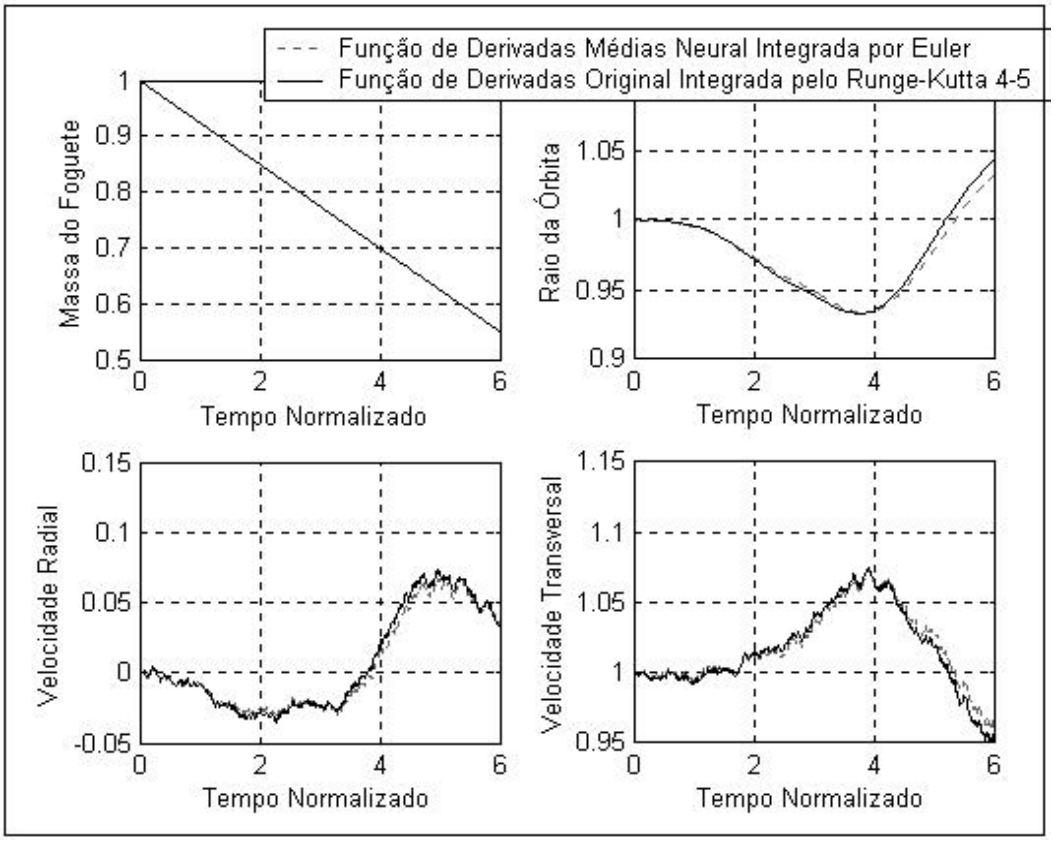


FIGURA 6.4 – Testes numéricos do integrador neural de passo fixo de Euler no problema da dinâmica de transferência de órbitas Terra/Marte (ΔT=0.01).

Alguns dados importantes quanto a arquitetura e o desempenho da rede neural utilizada para obter os resultados numéricos da Figura 6.4 são: rede com 5 entradas (4 variáveis de estados e 1 de controle), há somente uma camada interna com 41 neurônios *tansig* ($\lambda = 2$), rede com 4 saídas (funções de derivadas médias), as camadas de entrada e interna possuem cada uma um neurônio extra de viés com entrada sempre constante e igual a –1, no treinamento neural foram utilizados 3599 vetores de entrada/saída durante o treinamento e 1401 vetores de entrada/saída durante os testes, a média dos erros quadráticos de treinamento e teste alcançados durante o treinamento foram respectivamente $2{,}4789.10^{-6}$ e $2{,}7344.10^{-6}$, a média das normas das diferenças entre os valores originais e neurais é de $1{,}4916.10^{-2}$ e os vetores de normalização dos padrões de entrada e de saída da rede neural são respectivamente $\mathbf{vn_e}$=[1,000 2,000 1,4998 1,4998

3,7698] e $\mathbf{vn_s}$=[0,0749 1,4996 1,8416 2,6526]. A média dos erros quadráticos alcançada pelo método das derivadas instantâneas é de uma grandeza maior que aquele obtido pelo método das derivadas médias, pois, no caso das derivadas instantâneas, houve saturação do aprendizado, ou seja, a diminuição de uma casa decimal no erro de treinamento levará um tempo computacional de processamento extremamente elevado, e portanto, inviável na prática. Os domínios das variáveis de estado e de controle utilizados para o treinamento neural foram aqueles como apresentados pela Tabela 6.2.

TABELA 6.2 – Domínios das variáveis de estado e de controle utilizados no treinamento neural da função de derivadas médias.

Função de Derivadas Médias		
Variáveis	**Valor Mínimo**	**Valor Máximo**
M	**0,2**	**1,0**
R	**0,8**	**2,0**
W	**-1,5**	**+1,5**
V	**0,0**	**1,5**
θ	**$-1{,}2.\pi$**	**$+1{,}2.\pi$**

As Figuras 6.5 e 6.6 também apresentam as mesmas simulações obtidas, respectivamente, pelas Figuras 6.2 e 6.3, mas agora utilizando o método NARMAX ao invés do método das derivadas instantâneas. Na Figura 6.6 não há atualização das condições iniciais e, portanto, apenas pela análise visual, percebe-se que este último método é o menos eficiente de todos. Alguns dados importantes quanto a arquitetura e o desempenho da rede neural utilizada para obter os resultados numéricos das Figuras 6.5 e 6.6 são: rede com 5 entradas (4 variáveis de estados e 1 de controle no instante **t**), uma camada interna com 41 neurônios *tansig* ($\lambda = 2$), rede com 4 saídas (4 variáveis de estados no instante $\mathbf{t}+\Delta\mathbf{t}$ para $\Delta\mathbf{t}=\mathbf{0.01}$), as camadas de entrada e interna possuem cada uma um neurônio extra de viés com entrada constante e igual a –1, no treinamento neural foram utilizados 3599 vetores de entrada/saída durante o treinamento e 1401 vetores de entrada/saída durante os testes, a média dos erros quadráticos de treinamento

e teste, alcançados durante o treinamento, foram, respectivamente, $2{,}1115.10^{-7}$ e $2{,}1983.10^{-6}$, a média das normas das diferenças entre os valores originais e neurais da Figura 6.5 é de $7{,}4837.10^{-2}$ e da Figura 6.6 é de $1{,}8798.10^{-1}$ e os vetores de normalização dos padrões de entrada e de saída da rede neural são, respectivamente, $\mathbf{vn_e}$=[0.9996 1.9997 1.4999 1.4995 3.7692] e $\mathbf{vn_s}$=[0.9989 2.0132 1.5081 1.5148]. Os domínios das variáveis de estado e de controle utilizados para o treinamento neural foram àqueles como apresentados pela Tabela 6.3.

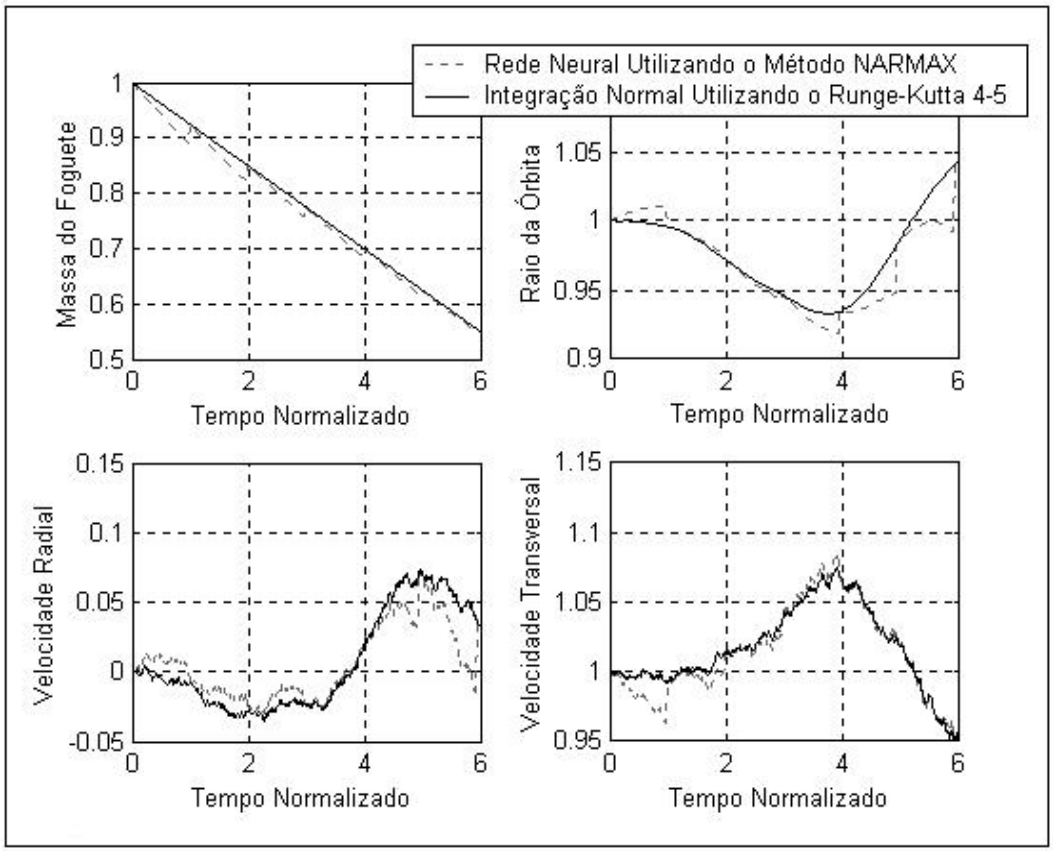


FIGURA 6.5 – Método NARMAX para a modelagem neural da dinâmica do problema de transferência de órbitas Terra/Marte com atualizações das condições iniciais após 100 propagações sucessivas (Δt=0.01).

Uma ressalva deve ser mencionada com respeito aos valores apresentados nas Tabelas 6.1, 6.2 e 6.3. Estes valores foram delimitados empiricamente. Por exemplo, em valores normalizados, se o raio da Terra é considerado igual a 1.0 então, o raio da órbita de Marte será de aproximadamente 1.6. Logo, numa transferência de órbitas entre esses

dois planetas, nada mais natural que delimitar o domínio de treinamento das rede neural no intervalo de segurança [0.8, 2.0]. As demais grandezas foram estipuladas através de exaustivas simulações numéricas no computador, que delimitou essa faixa nos padrões em consideração.

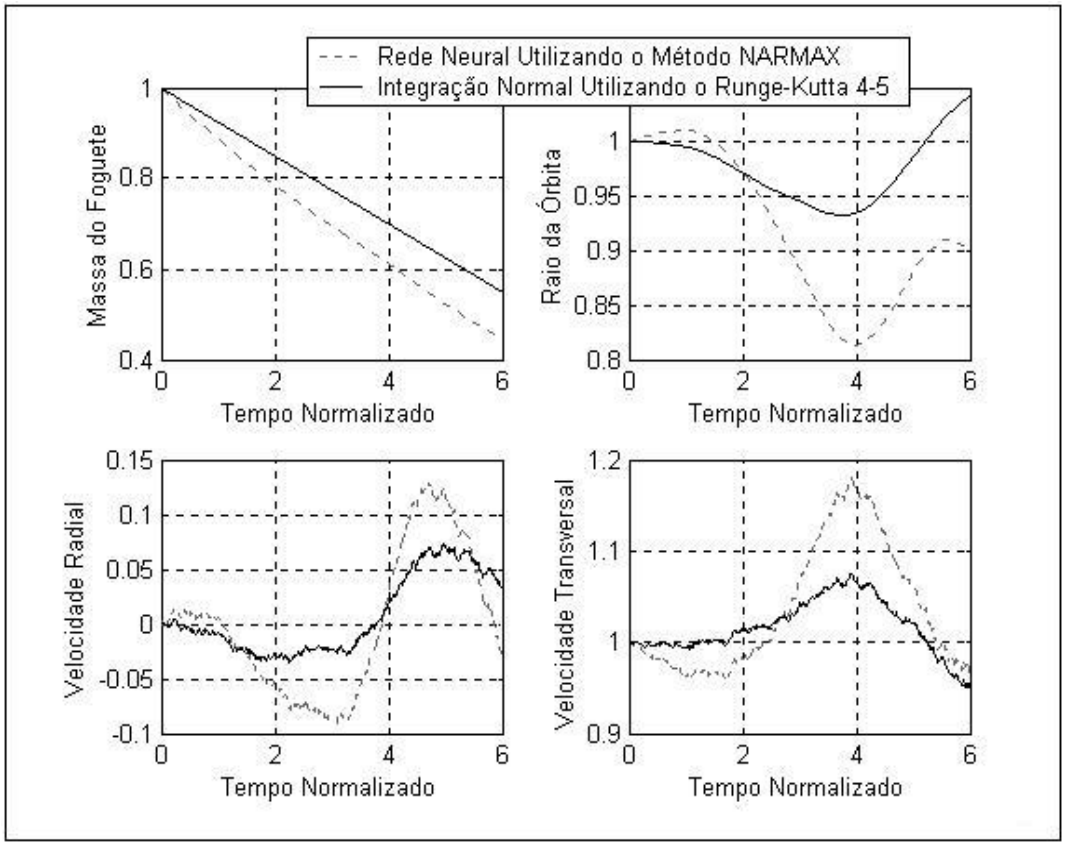


FIGURA 6.6 – Método NARMAX para a modelagem neural da dinâmica do problema da transferência de órbitas Terra/Marte sem atualizações das condições iniciais (Δt=0.01).

As demais figuras e gráficos apresentados nesta seção são referentes à aplicação das redes *feedforward* sobre as estruturas de controle preditivo neural. As Figuras 6.7 e 6.8 foram obtidas pelo método das derivadas instantâneas onde o integrador utilizado foi o de Adams-Bashforth de quarta ordem e o horizonte da previsão igual a 1. É importante perceber que no método das derivadas instantâneas é possível variar o passo de integração sem precisar treinar uma nova rede neural.

TABELA 6.3– Domínios das variáveis de estado e de controle utilizados no treinamento neural através do método NARMAX.

Função de Derivadas Médias		
Variáveis	**Valor máximo**	**Valor mínimo**
M	**0,2**	**1,0**
R	**0,8**	**2,0**
W	**-1,5**	**+1,5**
V	**0,0**	**1,5**
θ	**-1,2. π**	**+1,2. π**

Os resultados da Figura 6.7 mostram que para um passo de integração no valor de 0.01 unidades de tempo normalizado, consegue-se rastrear com precisão a trajetória de referência. A parte inferior da mesma figura demonstra que os controles estimados para este caso são relativamente suaves. Na Figura 6.8 tem-se o resultado de controle preditivo neural, onde o sistema parte com erro desprezível em relação ao início da trajetória de referência, mas por possuir um passo de integração e estimação iguais entre si e com o valor relativamente alto de 0.2 os resultados finais da simulação não ficaram muito precisos. A Figura 6.8 demonstra, portanto, que o método das derivadas instantâneas é sensível à escolha do passo de integração.

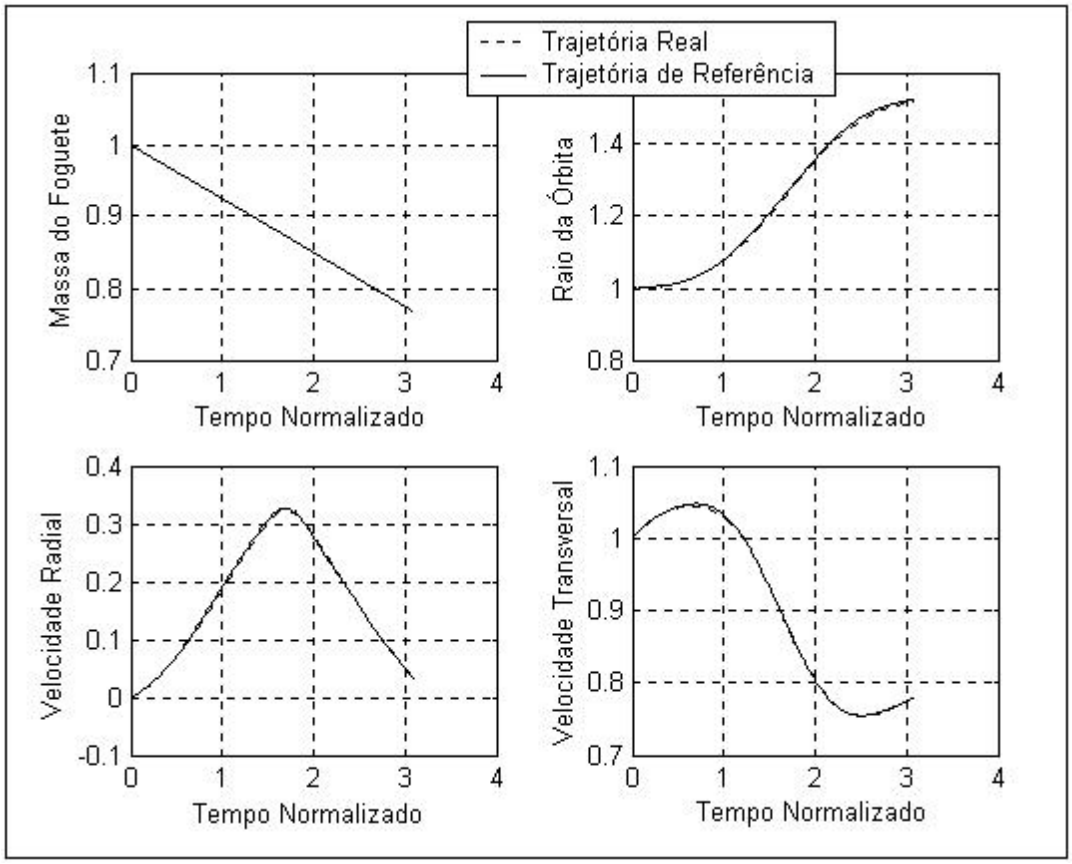
As Figuras de 6.9 à 6.14 são todas referentes à aplicação das redes *feedforward* sobre as estruturas de controle preditivo neural através do método das derivadas médias ou de integração média de Euler. As Figuras de 6.9 à 6.11 apresentam os resultados numéricos obtidos pela aplicação da rede neural treinada com passo de 0.01 para vários horizontes distintos, utilizados pela estrutura de controle preditivo, na estimação das trajetórias reais. Como pode ser visto todas essas simulações foram bem sucedidas.

As Figuras de 6.12 à 6.14 apresentam algo semelhante àquilo que foi apresentado nas Figuras de 6.9 à 6.11, mas aqui o passo de integração foi alterado para 0.1 ao invés de 0.01. Como o *software* desenvolvido neste trabalho só funciona corretamente quando o

passo de integração é igual ao passo de estimação, então para a realização destes últimos testes foi necessário treinar uma nova rede neural com passo de integração 0.1. Os resultados apresentados aqui são tambem semelhantes àqueles obtidos com passo de integração igual a 0.01.

TABELA 6.4 – Parâmetros e desempenho alcançado pelo estimador preditivo.

t_o=0.0	R_u=10^{-2}	Número de Iterações do Filtro de Kalman em cada instante = 30
t_f=3.1	Ry=10^{-12}	Erros Absolutos das Varáveis de Estado (EAVE) em t_f =
Δt = 0.01	$\overline{U}_o$= 0 e α = 0.2	[0,01.10^{-13} 4,97.10^{-3} 1,14.10^{-3} 4,75.10^{-4}]



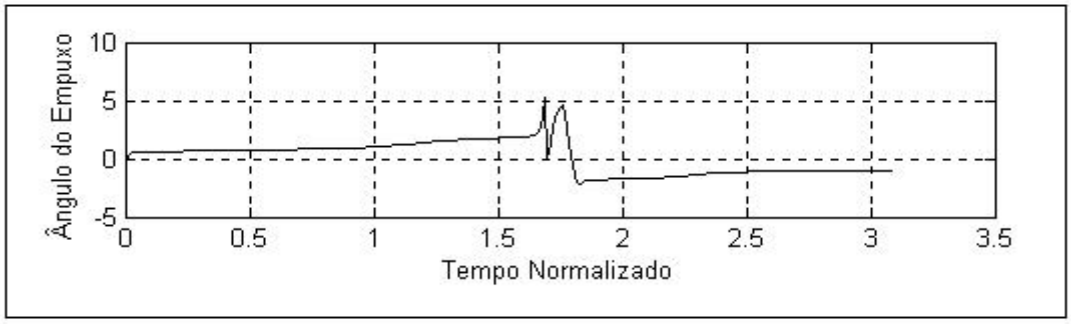


FIGURA 6.7 – Estrutura de controle preditivo neural aplicada à dinâmica de transferência de órbitas Terra/Marte utilizando o método de Adams-Bashforth de quarta ordem (Δt=0.01).

TABELA 6.5 – Parâmetros e desempenho alcançado pelo estimador preditivo.

T_o=0.0	R_u=10^{-2}	Número de Iterações do Filtro de Kalman em cada instante = 30
T_f=3.1	Ry=10^{-12}	Erros Absolutos das Varáveis de Estado (EAVE) em t_f =
Δt = 0.2	$\overline{U}_o$= 0 e α = 0.2	[7,5.10^{-3} 3,0.10^{-3} 3,04.10^{-2} 9,4.10^{-3}]

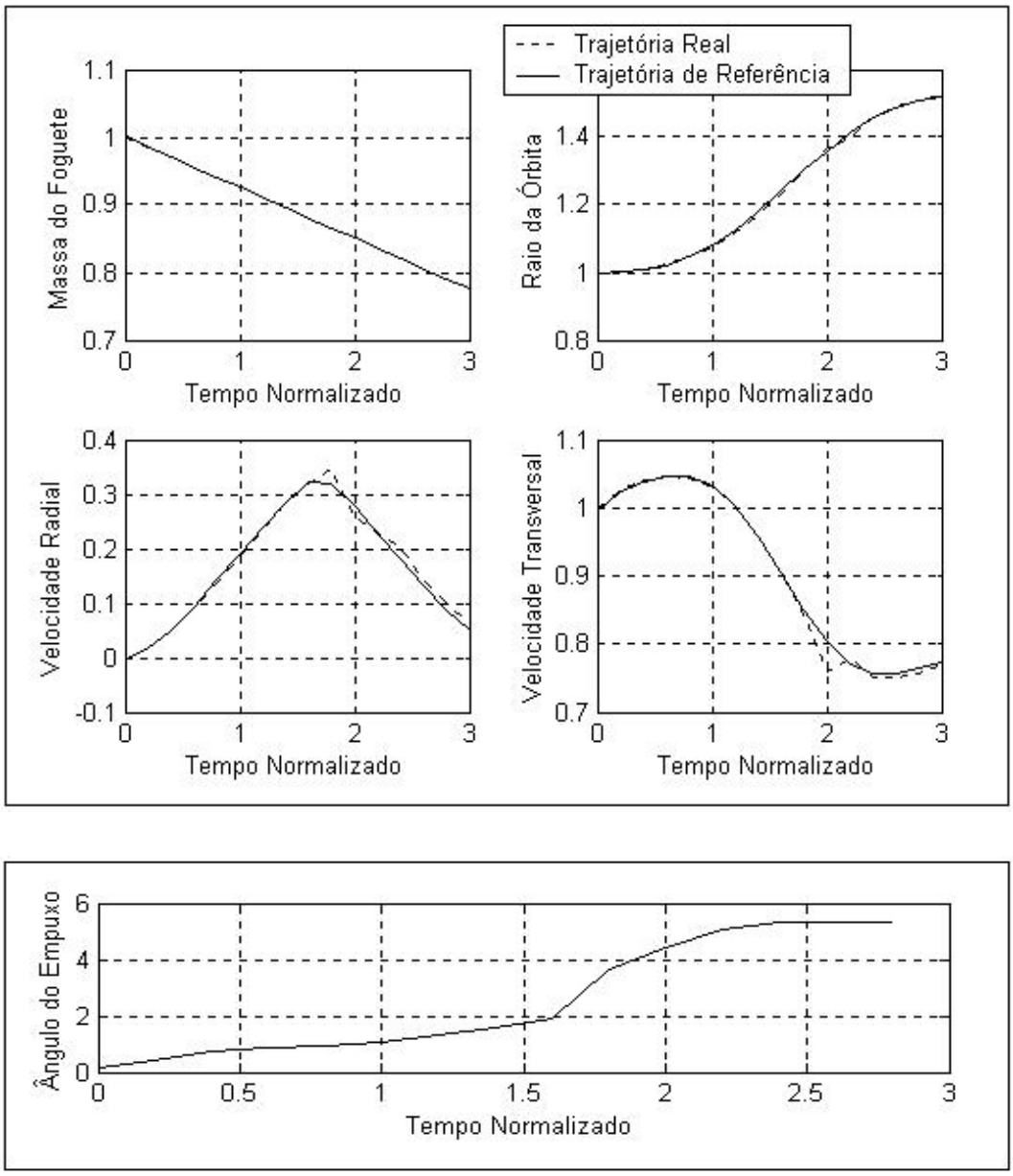


FIGURA 6.8 – Estrutura de controle preditivo neural aplicada à dinâmica de transferência de órbitas Terra/Marte utilizando o método de Adams-Bashforth de quarta ordem (Δt=0.2).

Tabela 6.6 – Parâmetros e desempenho alcançado pelo estimador preditivo.

t_o=0.0	R_u=10^{-2}	Número de Iterações do Filtro de Kalman em cada instante = 20
t_f=3.1	Ry=10^{-12}	Erros Absolutos das Varáveis de Estado (EAVE) em t_f =
Δt = 0.01	$\overline{U}_o$= 0 e α = 0.2	[0,4.10^{-13} 2,29.10^{-2} 5,37.10^{-3} 2,72.10^{-3}]

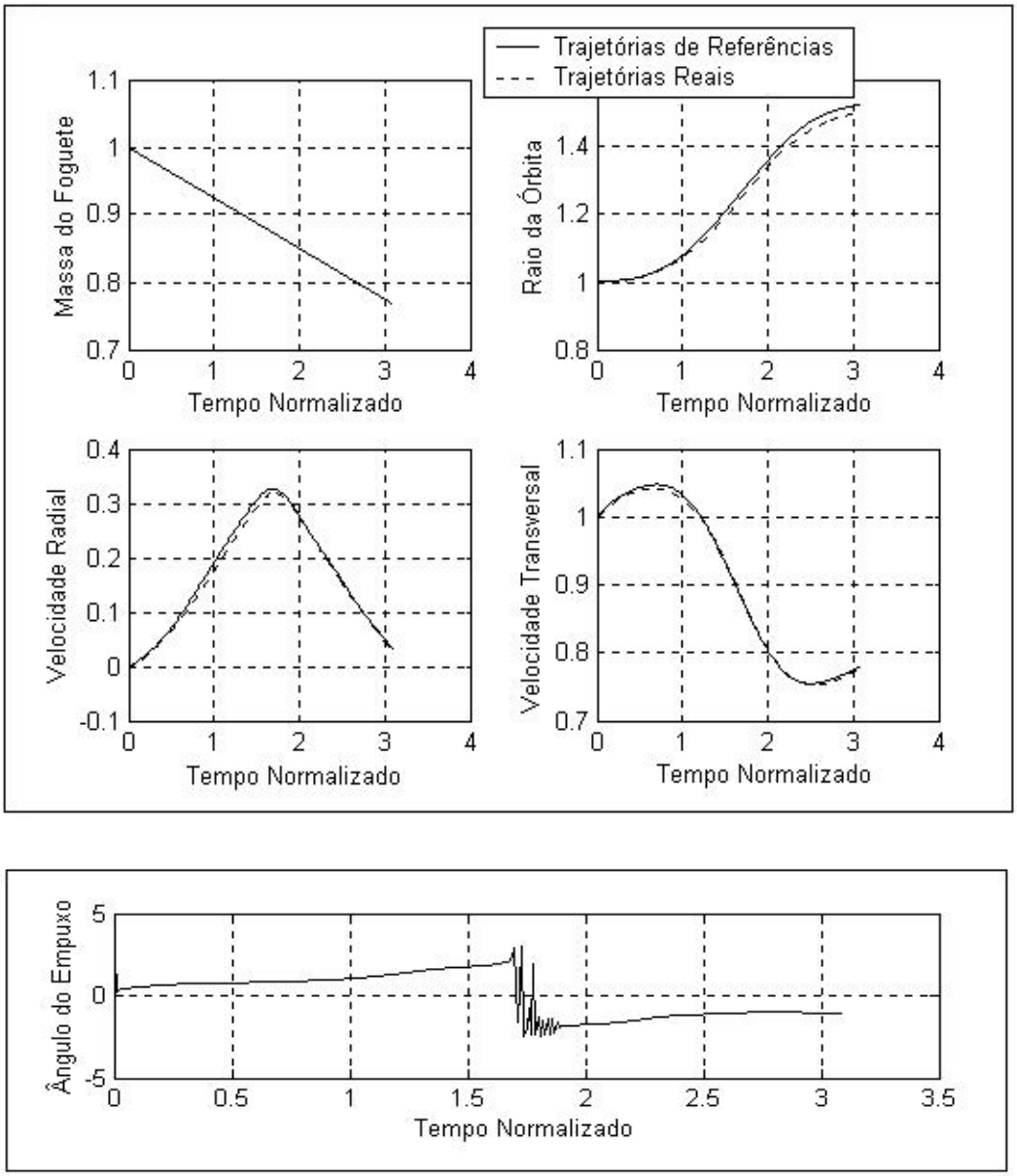
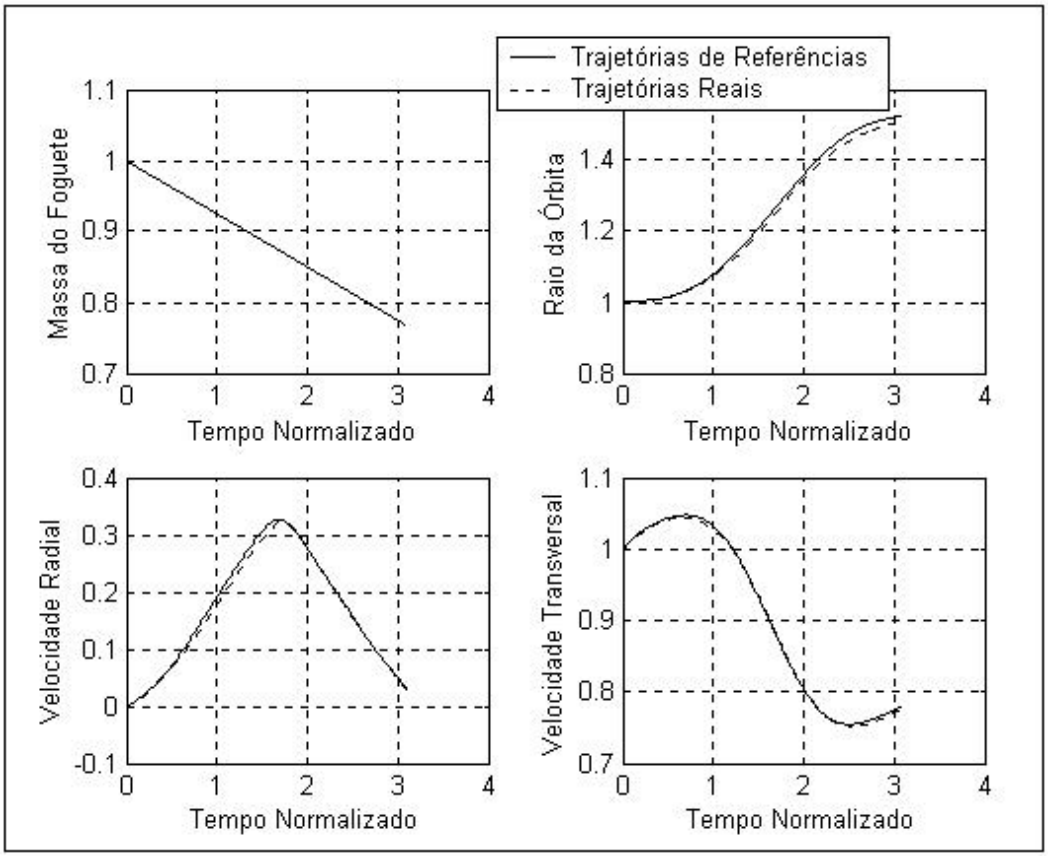


Figure 6.9 – Estrutura de controle preditivo neural aplicada à dinâmica de transferência de órbitas Terra/Marte utilizando o método de passo fixo de Euler ou das derivadas médias (Δt=0.01 e m=1).

TABELA 6.7 – Parâmetros e desempenho alcançado pelo estimador preditivo.

t_o=0.0	R_u=1	Número de Iterações do Filtro de Kalman em cada instante = 20
t_f=3.1	Ry=10^{-12}	Erros Absolutos das Varáveis de Estado (EAVE) em t_f =
Δt = 0.01	$\overline{U}_o$= 0 e α = 0.1	[0,4.10^{-13} 1,83.10^{-2} 2,97.10^{-3} 4,17.10^{-3}]



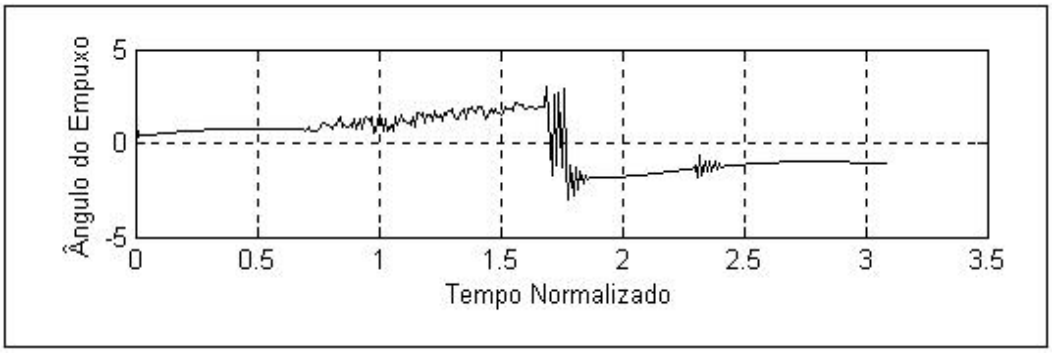
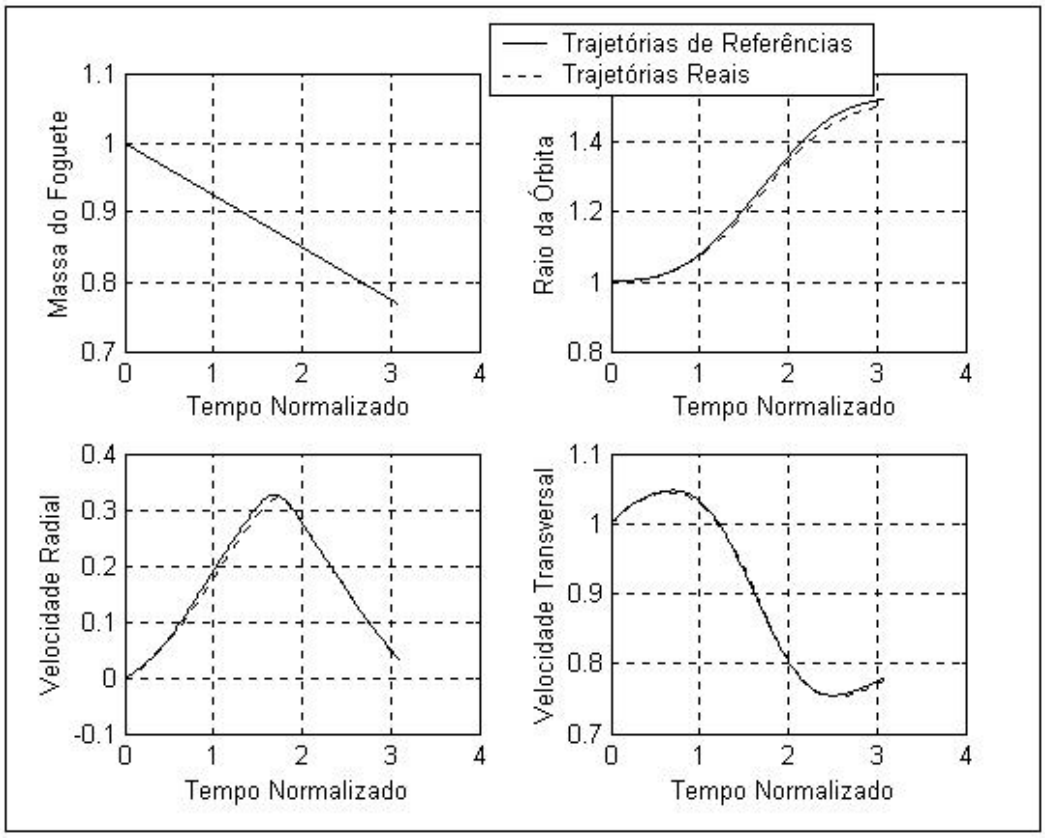


FIGURA 6.10 - Estrutura de controle preditivo neural aplicada à dinâmica de transferência de órbitas Terra/Marte utilizando o método de passo fixo de Euler ou das derivadas médias (Δt=0.01 e m=5).

TABELA 6.8-Parâmetros e desempenho alcançado pelo estimador preditivo.

t_o=0.0	R_u=1	Número de Iterações do Filtro de Kalman em cada instante = 20
t_f=3.1	Ry=10^{-12}	Erros Absolutos das Varáveis de Estado (EAVE) em t_f =
Δt = 0.01	$\overline{U}_o$= 0 e α = 0.1	[0,4.10^{-13} 1,91.10^{-2} 3,12.10^{-3} 4,20.10^{-3}]



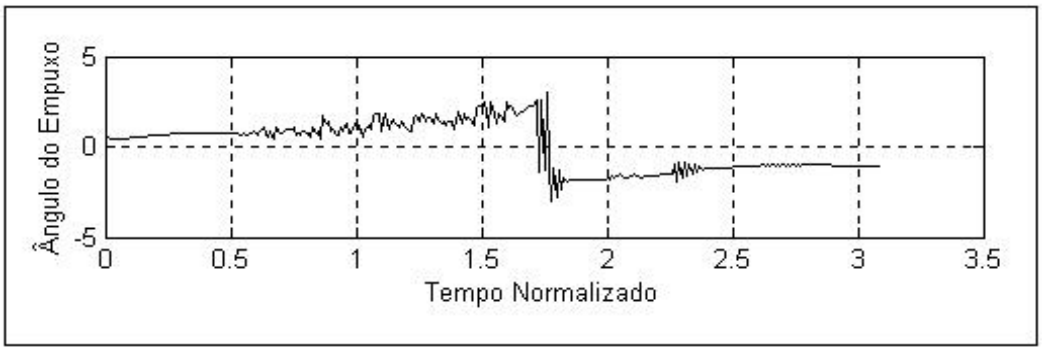


FIGURA 6.11 - Estrutura de controle preditivo neural aplicada à dinâmica de transferência de órbitas Terra/Marte utilizando o método de passo fixo de Euler ou das derivadas médias (Δt=0.01 e m=10).

TABELA 6.9-Parâmetros e desempenho alcançado pelo estimador preditivo.

t_o=0.0	R_u=1	Número de Iterações do Filtro de Kalman em cada instante = 10
t_f=3.1	Ry=10^{-12}	Erros Absolutos das Varáveis de Estado (EAVE) em t_f =
Δt = 0.1	$\overline{U}_o$= 0 e α = 0.2	[5,64.10^{-7} 5,58.10^{-3} 2,15.10^{-2} 1,62.10^{-3}]

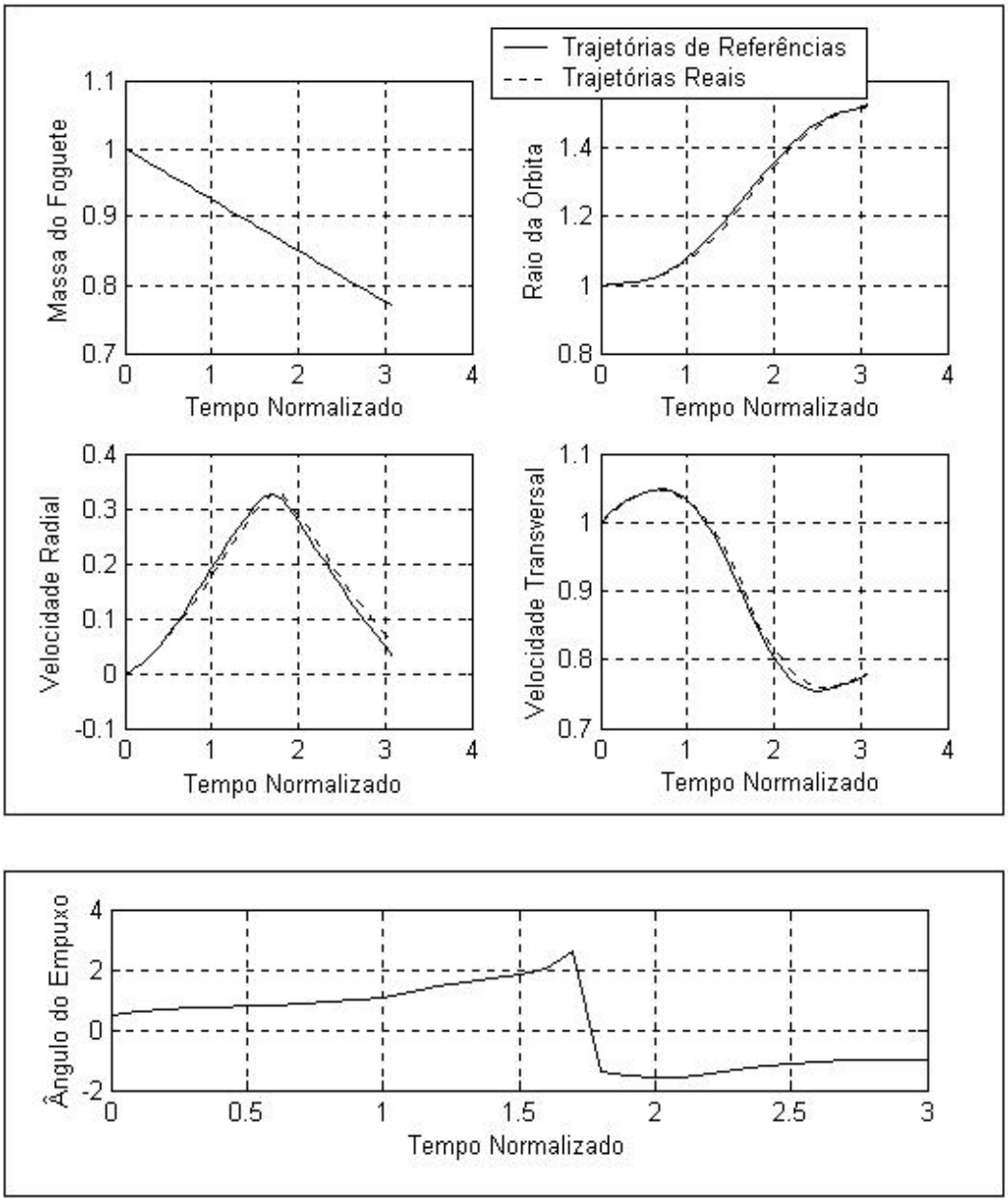
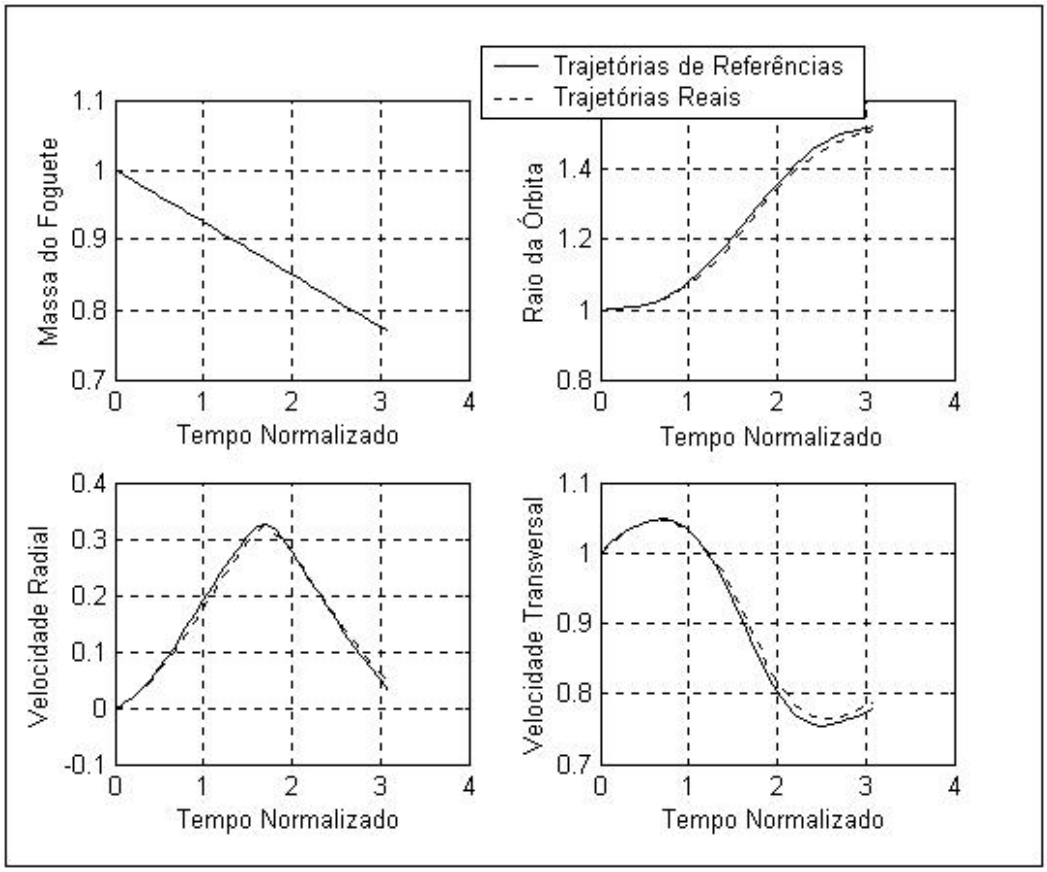


FIGURA 6.12 - Estrutura de controle preditivo neural aplicada à dinâmica de transferência de órbitas Terra/Marte utilizando o método de passo fixo de Euler ou das derivadas médias (Δt=0.1 e m=1).

TABELA 6.10-Parâmetros e desempenho alcançado pelo estimador preditivo.

t_o=0.0	$R_u=10^{-2}$	Número de Iterações do Filtro de Kalman em cada instante = 5
t_f=3.1	$Ry=10^{-12}$	Erros Absolutos das Varáveis de Estado (EAVE) em t_f =
$\Delta t = 0.1$	$\overline{U}_o = 0$ e $\alpha = 0.2$	$[\ 5{,}90.10^{-7} \quad 1{,}03.10^{-2} \quad 1{,}14.10^{-2} \quad 8{,}33.10^{-3}]$



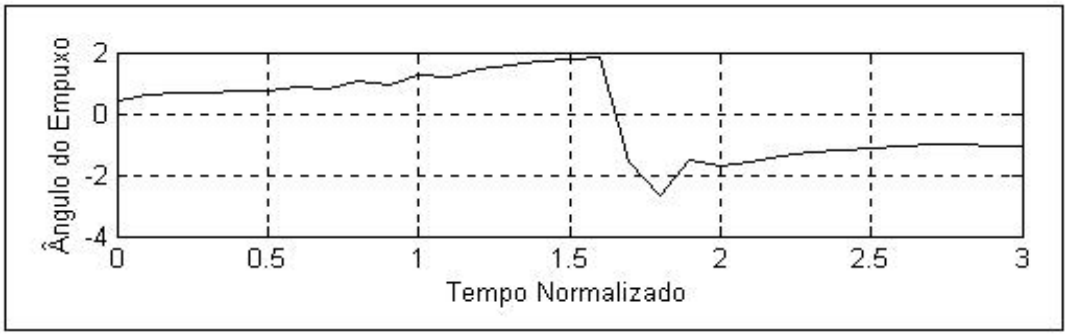


FIGURA 6.13 – Estrutura de controle preditivo neural aplicada à dinâmica de transferência de órbitas Terra/Marte utilizando o método de passo fixo de Euler ou das derivadas médias (Δt=0.1 e m=5).

TABELA 6.11-Parâmetros e desempenho alcançado pelo estimador preditivo.

t_o=0.0	$R_u=10^{-2}$	Número de Iterações do Filtro de Kalman em cada instante = 5
t_f=3.1	$Ry=10^{-12}$	Erros Absolutos das Varáveis de Estado (EAVE) em t_f =
$\Delta t = 0.1$	$\overline{U}_o = 0$ e $\alpha = 0.2$	$[\ 5{,}92.10^{-7} \quad 7{,}98.10^{-3} \quad 1{,}27.10^{-2} \quad 7{,}43.10^{-3}]$

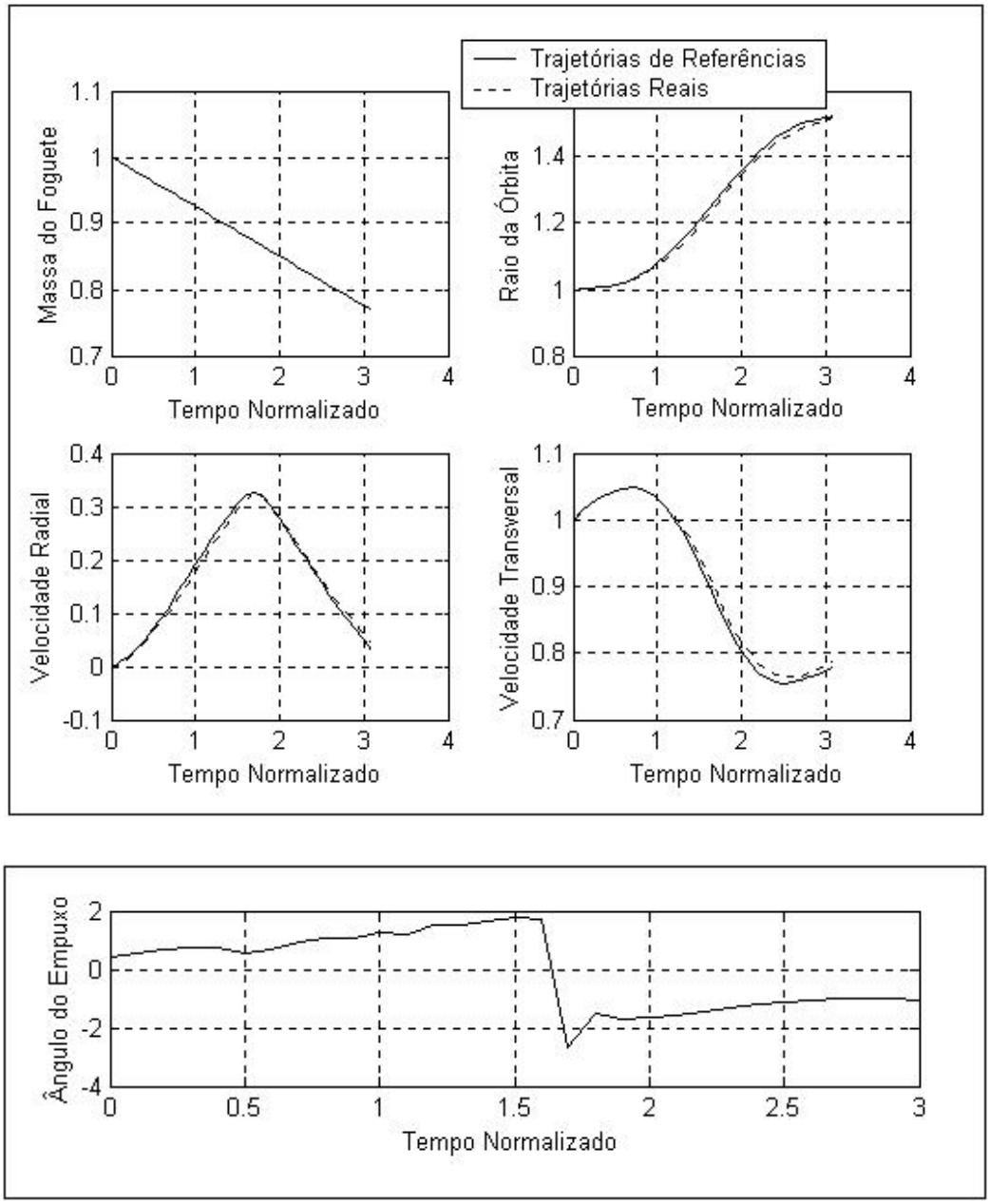


FIGURA 6.14 - Estrutura de controle preditivo neural aplicada à dinâmica de transferência de órbitas Terra/Marte utilizando o método de passo fixo de Euler ou das derivadas médias (Δt=0.1 e m=10).

6.2.3 Análise dos Resultados.

Uma nova abordagem para obter modelos de sistemas discretos para frente, para ser utilizada em esquemas de controle onde um modelo interno é necessário, foi aplicada para a dinâmica do problema de transferência de órbitas Terra/Marte. A estrutura de integradores numéricos *Ordinary Differential Equation* (ODE) foi utilizada para obter um modelo discreto para frente onde a rede neural teve somente que aprender e aproximar uma função de derivadas, que é de natureza estática, do sistema dinâmico considerado. Os testes reforçaram os seguintes aspectos característicos:

1) é uma tarefa mais simples treinar uma rede neural *feedforward* para aprender uma função algébrica estática de um sistema dinâmico ODE (onde as entradas são variáveis dos estados e dos controles), do que treiná-la para aprender um modelo discreto do tipo NARMAX (onde as entradas são compostas de respostas atrasadas de estados e controles); a rede neural inserida em um integrador ODE resultou ser mais simples, em termos da necessidade do números de neurônios, uma vez que ela não tem que aprender a lei da dinâmica, mas somente a função de derivadas;
2) para se controlar a precisão do erro do integração, no caso específico do método das derivadas instantâneas, pode-se variar o tamanho do passo do integrador e a ordem do integrador numérico;

O uso de integradores numéricos neurais como um modelo interno em uma estrutura de controle preditivo aplicados no exemplo prático do problema de transferência de órbitas implicou que:

1) interpretação estocástica dos erros dá mais realismo ao tratamento do problema, e facilita o ajuste das matrizes de pesos num funcional de controle preditivo;

2) a performance dos algoritmos de processamento paralelo garante a aplicabilidade em tempo real desta metodologia, pois o algoritmo convergiu adequadamente com somente um passo a frente para rastrear o horizonte de

controle, embora com vários horizontes a frente o modelo também tenha sido testado.

Para finalizar essa seção, a Tabela 6.12 trás um resumo dos parâmetros utilizados para gerar os gráficos das simulações apresentadas anteriormente pelas figuras de 6.7 à 6.14.

TABELA 6.12 – Resumo detalhado das simulações numéricas obtidas pela estrutura de controle preditivo neural sobre o problema de trasferência de órbitas Terra/Marte.

Estrutura de Controle Preditivo Neural	**Número de Iterações do filtro de Kalman em cada instante**	Δt	R_u	R_y	**Norma do Erro Absoluto**	**Número de Horizontes**
Derivadas Instantâneas (Adams-Bashforth de quarta ordem – Fig. 6.7)	30	0.01	10^{-2}	10^{-12}	$5{,}12.\ 10^{-3}$	1
Derivadas Instantâneas (Adams-Bashforth de quarta ordem – Fig. 6.8)	30	0.2	10^{-2}	10^{-12}	$3{,}28.\ 10^{-2}$	1
Derivadas Médias – Fig. 6.9	20	0.01	10^{-2}	10^{-12}	$2{,}37.\ 10^{-2}$	1
Derivadas Médias – Fig. 6.10	20	0.01	1	10^{-12}	$1{,}90.\ 10^{-2}$	5
Derivadas Médias – Fig. 6.11	20	0.01	1	10^{-12}	$1{,}98.\ 10^{-2}$	10
Derivadas Médias – Fig. 6.12	10	0.1	1	10^{-12}	$2{,}23.\ 10^{-2}$	1
Derivadas Médias – Fig. 6.13	5	0.1	10^{-2}	10^{-12}	$1{,}75.\ 10^{-2}$	5
Derivadas Médias – Fig. 6.14	5	0.1	10^{-2}	10^{-12}	$1{,}67.\ 10^{-2}$	10

6.3 O Problema de Controle da Atitude de Satélites Artificiais.

As equações cinemáticas (Wertz, 1978) e dinâmicas (Kaplan, 1976) para a representação do movimento da atitude de satélites artificiais atuando como corpos rígidos são dadas, respectivamente, por:

$$\begin{Bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\varphi} \end{Bmatrix} = \frac{1}{\cos\theta} \cdot \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \cos\varphi.\cos\theta & \sin\varphi.\cos\theta & 0 \\ -\cos\varphi.\sin\theta & -\sin\varphi.\sin\theta & \cos\theta \end{bmatrix} \cdot \begin{Bmatrix} w_x \\ w_y \\ w_z \end{Bmatrix} \tag{3.a}$$

$$\dot{w}_x = \left(\frac{I_y - I_z}{I_x} \right) \cdot w_y . w_z + \frac{T_x}{I_x}$$

$$\dot{w}_y = \left(\frac{I_z - I_x}{I_y} \right) . w_x . w_z + \frac{T_y}{I_y} \tag{3.b}$$

$$\dot{w}_z = \left(\frac{I_x - I_y}{I_z} \right) . w_x . w_y + \frac{T_z}{I_z}$$

onde, ϕ, θ e φ são os denominados ângulos de Euler e w_x, w_y e w_z são as velocidades angulares em relação aos eixos fixos ao corpo. As grandezas T_x, T_y e T_z são os torques externos que o satélite, que se comporta como corpo rígido, está sujeito no espaço, e são também as variáveis de controle. As trajetórias de referência são dadas somente para os ângulos de Euler. As expressões das referências para os ângulos de Euler são dadas por exponenciais decrescentes (Silva, 2001) e constantemente atualizadas pelos dados de navegação, ou seja:

$$\phi_{ref}(t_{k+1}) = \phi(t_k) \cdot \exp\left[-\beta \cdot (t_{k+1} - t_k) \right] \tag{4.a}$$

$$\theta_{ref}(t_{k+1}) = \theta(t_k) \cdot \exp\left[-\beta \cdot (t_{k+1} - t_k) \right] \tag{4.b}$$

$$\varphi_{ref}(t_{k+1}) = \varphi(t_k) \cdot \exp\left[-\beta \cdot (t_{k+1} - t_k) \right] \tag{4.c}$$

A matriz de inércia para o exemplo em questão foi adotada (Rimrott,1989) em:

$$\mathbf{I} = \begin{bmatrix} 30 & 0 & 0 \\ 0 & 40 & 0 \\ 0 & 0 & 20 \end{bmatrix} [\mathbf{kg.m^2}] \tag{5}$$

As equações (3.a) e (3.b) representam um sistema de equações diferenciais ordinárias não-linear com 6 variáveis de estado e 3 variáveis de controle.

Nesta seção serão apresentadas as simulações numéricas obtidas pela estrutura de controle preditivo neural que trabalhou com o modelo da dinâmica, representado pelo método das derivadas médias, na tentativa de representar as equações (3.a) e (3.b) por uma rede neural com arquitetura *feedforward* através de um treinamento supervisionado. Entretanto, antes disso é necessário dar um pequeno esclarecimento do funcionamento de uma estrutura de controle preditivo como mostrado na Figura 6.15.

A Figura 6.15 representa um esquema simplificado do funcionamento de uma estrutura de controle preditivo. O mais importante a saber a respeito deste fluxograma é da existência de dois modelos dinâmicos: o modelo de trabalho e o modelo de validação. Para ser correto, o modelo de trabalho deve ser representado pela rede neural treinada pelo método das derivadas médias e o modelo de validação por um integrador de alta-ordem integrando as funções de derivadas (3.a) e (3.b), onde Δt_1 é o passo de integração neural do método das derivadas médias, Δt_2 é o passo da estrutura de controle preditivo e Δt_3 é o passo do integrador de alta-ordem sobre o sistema dinâmico dado pelas funções de derivadas originais.

A Figura 6.15 faz-se necessária em vista da dificuldade, que algumas vezes é verificada, da rede neural aprender a dinâmica de um sistema. Como pode ser visto na Figura 6.20 e 6.21 a estrutura de controle preditivo neural apresenta ainda um pequeno desvio que oscila ligeiramente fora da referência do ângulo zero. Para explicar essa característica indesejável e atribuí-la ao fato de que é devida a um erro de treinamento neural ainda insuficiente se faz uso da Figura 6.15.

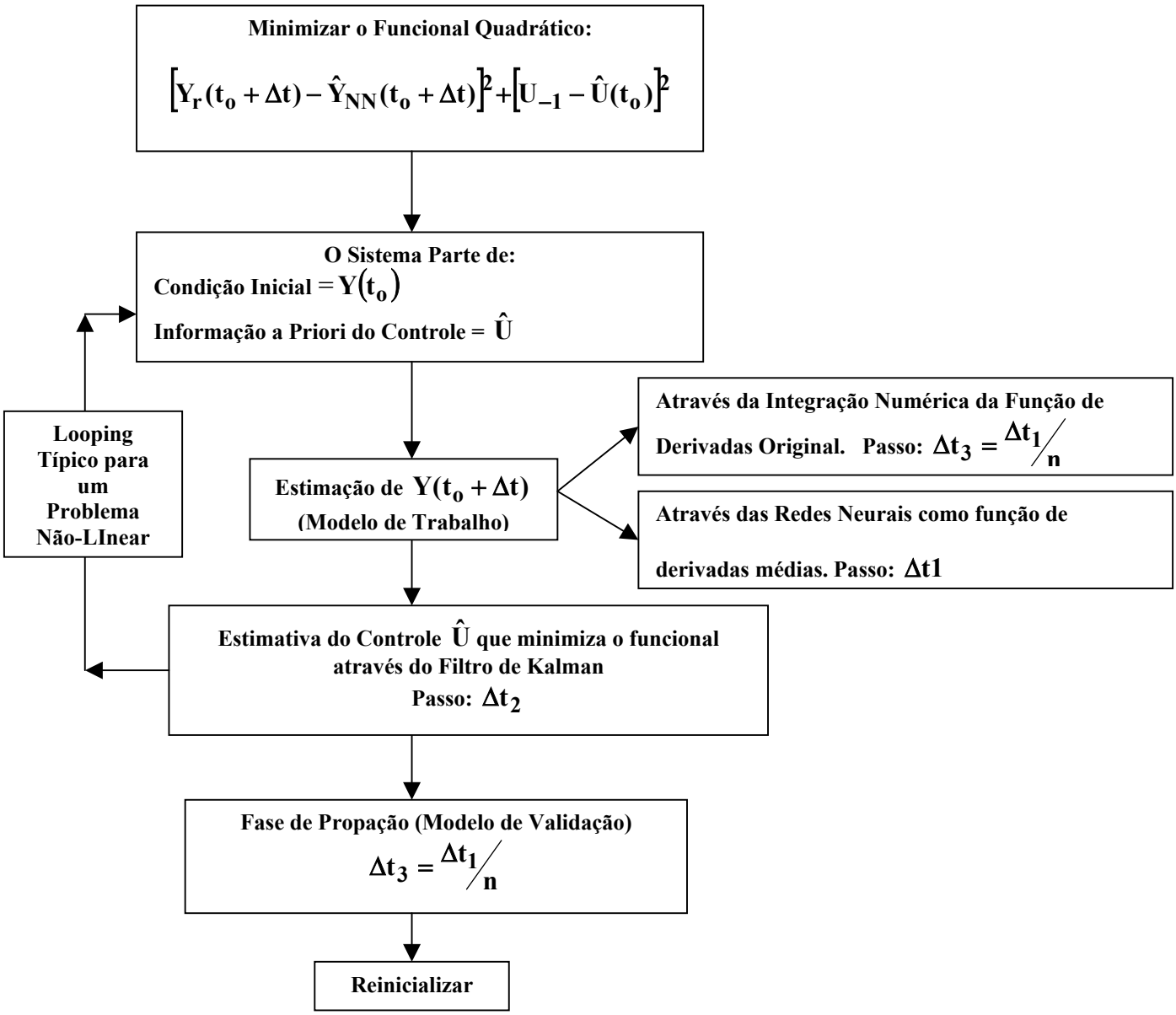


FIGURA 6.15 – Fluxograma geral de uma estrutura de controle preditivo.

Para comprovar a afirmação anterior primeiro é necessário analisar os casos ideais, ou seja, quando a estrutura de controle preditivo trabalha com os modelos de trabalho e de validação, ambos, sendo representados por um integrador de alta-ordem que faz uso das funções de derivadas originais dadas pelas expressões (3.a) e (3.b), sem a presença de ruídos no modelo de validação. Como podem demonstrar as Figuras 6.16 e 6.17, respectivamente, obtidas para os valores de β= **0.05** e β=**0.01** (ver equações 4.a à 4.b); nestes dois casos houve convergência e a influencia do parâmetro β é em apenas alargar

o intervalo de tempo para a estabilização do satélite na referencia zero, quando β decresce. Observe que aqui ainda não se fez uso da rede neural. Entretanto, o problema maior está presente nas Figuras 6.20 e 6.21, onde o modelo de trabalho é dado pela rede neural e o modelo de validação representado pelo integrador de alta-ordem. Neste caso, como já dito anteriormente, o algoritmo converge para um valor que oscila ligeiramente fora da referencia zero para dois dos ângulos. Para entender esse comportamento e saber que o responsável por essa imprecisão é um Erro quadrático Médio (EQM) de treinamento neural insuficiente, deve-se recorrer as Figuras 6.18 e 6.19.

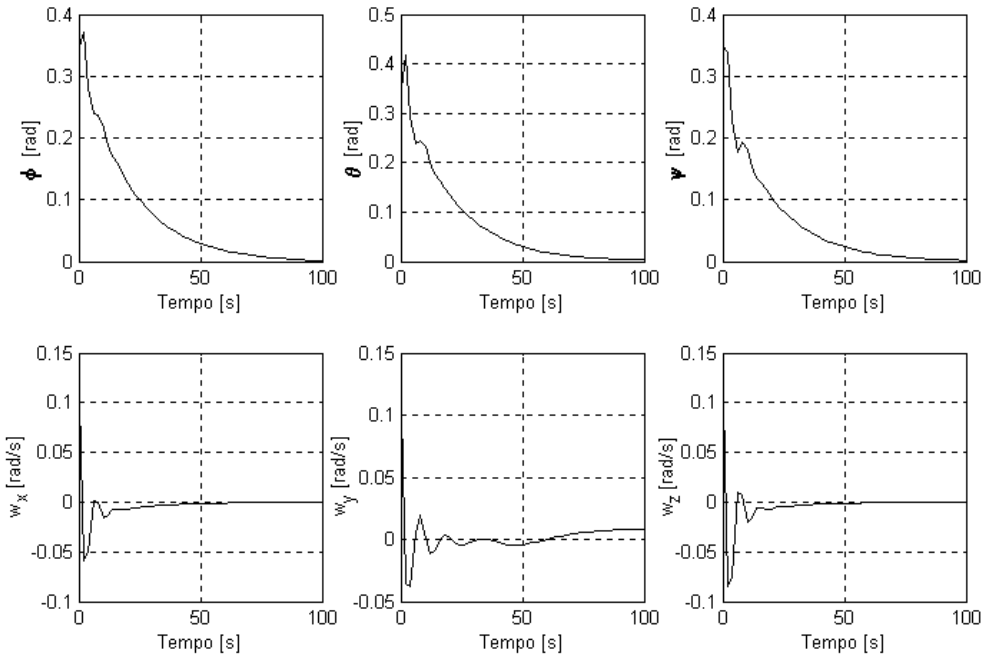
Na Figura 6.18 tanto o modelo de trabalho como o modelo de validação são dados sem a utilização da rede neural. Entretanto, nesta figura até o instante de t=30 [s] o modelo de validação está sem ruído, mas depois disso um ruído de aproximadamente de 120% do valor nominal é acrescentado ao modelo de validação, fazendo que a estrutura de controle preditivo convirja para valores fora dos nominais. A finalidade de se acrescentar o ruído é para simular o erro gerado pela rede neural, com a vantagem agora de poder especificá-lo na magnitude que se interessa, mas sem precisar passar por um processo penoso e demorado de treinamento neural para obter a grandeza desses mesmos erros.

A atenção principal está, então, na Figura 6.18 com ruído de 120% do valor nominal, pois esse comportamento é semelhante àquele obtido pela rede neural das Figuras 6.20 e 6.21, a menos de estar bem mais exagerado. O que se verifica experimentalmente é que se for diminuído o valor do ruído presente na Figura 6.18, então o esquema de controle preditivo convergirá para um valor nominal mais preciso, como ilustra a Figura 6.19, onde o ruído foi reduzido para 100%.

Entretanto, tentar diminuir os ruídos presentes nas Figuras 6.20 e 6.21 é uma tarefa, na prática, difícil uma vez que se observou a saturação do aprendizado da rede neural, ou seja, o tempo de processamento do treinamento neural em questão torna-se grande demais para ser considerado viável.

Condições Iniciais: $\phi = \theta = \varphi = 20^{o} = \frac{\pi}{180} \cdot 20$ [rad]	**Condições Iniciais:** $\omega_x = \omega_y = \omega_z = 1[\text{rpm}] = \frac{\pi}{30} \cdot 1$ [rad/s]
t_o= 0 [s] e t_f= 100 [s]	$\beta = 0.05$, $R_u=10^{-2}$ e $R_y=10^{-6}$
Δt_1 (Passo da Rede Neural)= Não foi utilizado	Δt_3 (Passo do Modelo de Validação) = $\frac{2}{50}$ [s] Runge-Kutta de Quarta Ordem
Δt_2 (Passo do Controle Preditivo)= 2 [s]	Iterações do Filtro de Kalman = 15 Iterações (em cada instante)

Variáveis de Estados:



Variáveis de Controles Estimados:

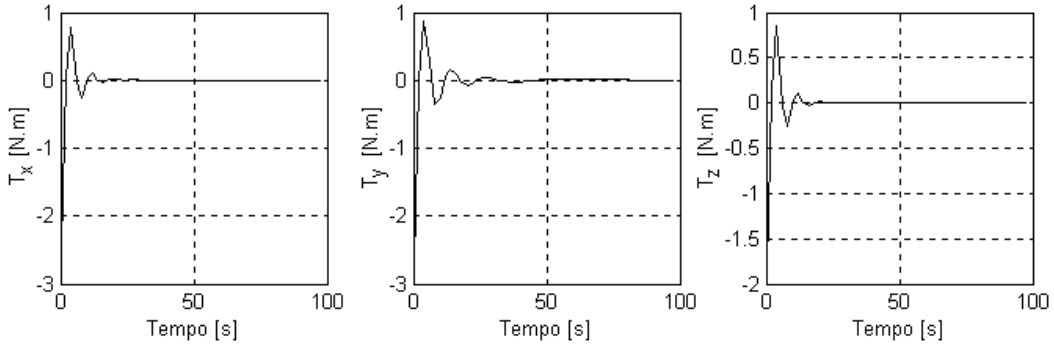
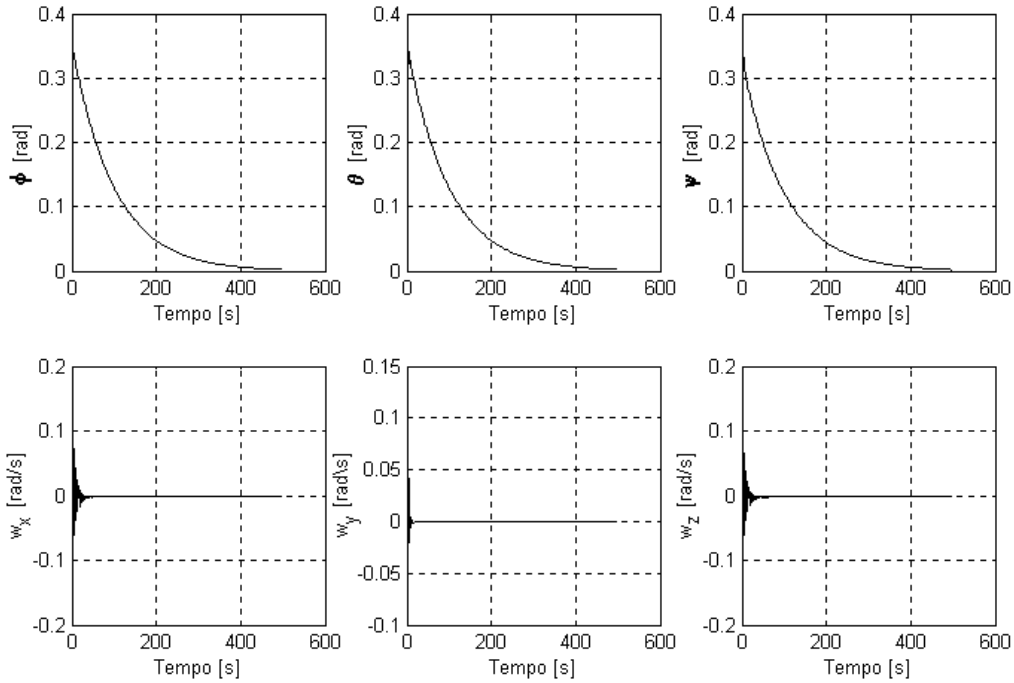


FIGURA 6.16 – Estrutura de controle preditivo com apenas o integrador, sem ruído e β=0.05.

Condições Iniciais: $\phi = \theta = \varphi = 20^{o} = \frac{\pi}{180} \cdot 20$ [rad]	**Condições Iniciais:** $\omega_x = \omega_y = \omega_z = 1[\text{rpm}] = \frac{\pi}{30} \cdot 1$ [rad/s]
t_o= 0 [s] e t_f= 500 [s]	$\beta = 0.01$, $R_u=10^{-2}$ e $R_y=10^{-6}$
Δt_1 (Passo da Rede Neural)= Não foi utilizada	Δt_3 (Passo do Modelo de Validação) = $\frac{2}{50}$ [s] Runge-Kutta de Quarta Ordem
Δt_2 (Passo do Controle Preditivo)= 2 [s]	Iterações do Filtro de Kalman = 40 Iterações (em cada instante) e $\alpha = 0.15$

Variáveis de Estados:



Variáveis de Controles Estimados:

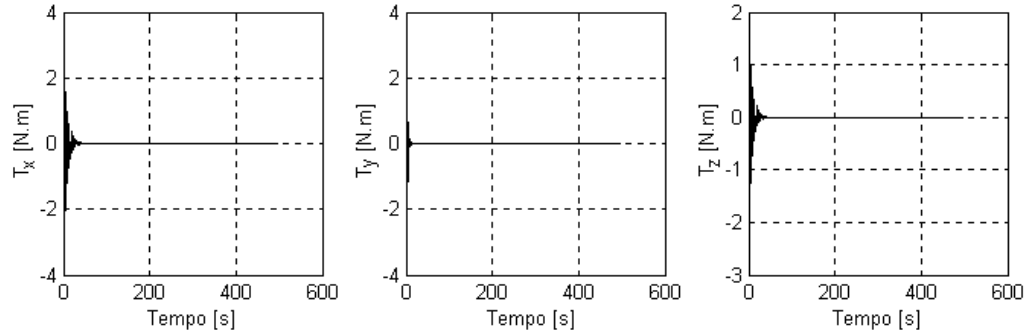
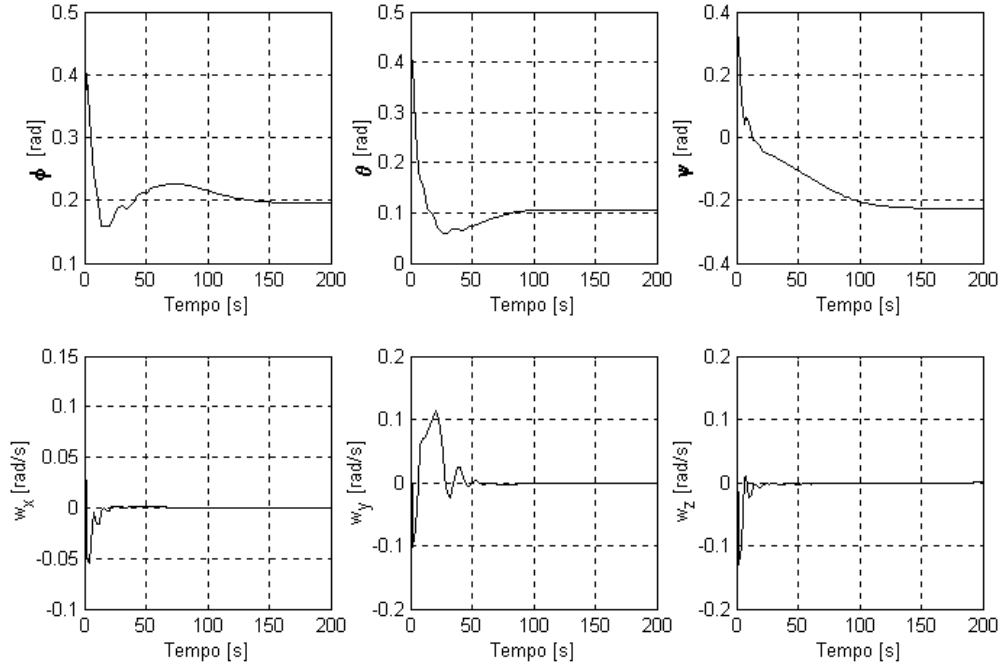


FIGURA 6.17 – Estrutura de controle preditivo com apenas o integrador, sem ruído e β=0.01.

Condições Iniciais: $\phi = \theta = \varphi = 20^{o} = \frac{\pi}{180} \cdot 20$ [rad]	**Condições Iniciais:** $\omega_x = \omega_y = \omega_z = 1[\text{rpm}] = \frac{\pi}{30} \cdot 1$ [rad/s]
$t_o = 0$ [s] e $t_f = 200$ [s]	$\beta = 0.05$, $R_u = 10^{-2}$ e $R_y = 10^{-6}$
Δt_1 (Passo da Rede Neural)= Não foi utilizada	Δt_3 (Passo do Modelo de Validação) = $\frac{2}{50}$ [s] Runge-Kutta de Quarta Ordem
Δt_2 (Passo do Controle Preditivo)= 2 [s]	Iterações do Filtro de Kalman = 15 Iterações (em cada instante) e $\alpha = 0.15$

Variáveis de Estados:



Variáveis de Controles Estimados:

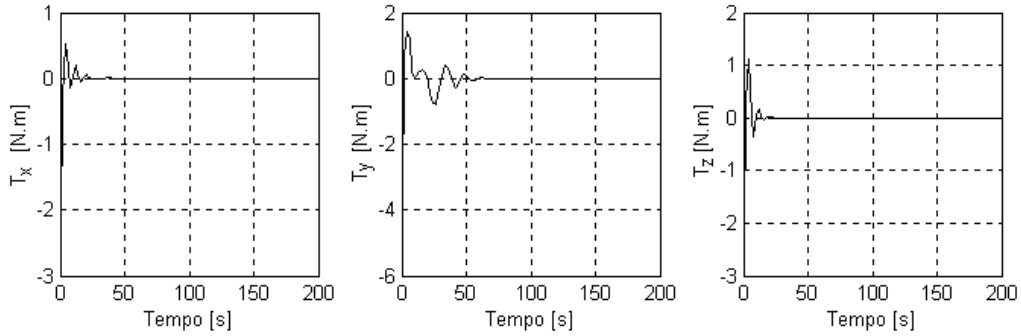
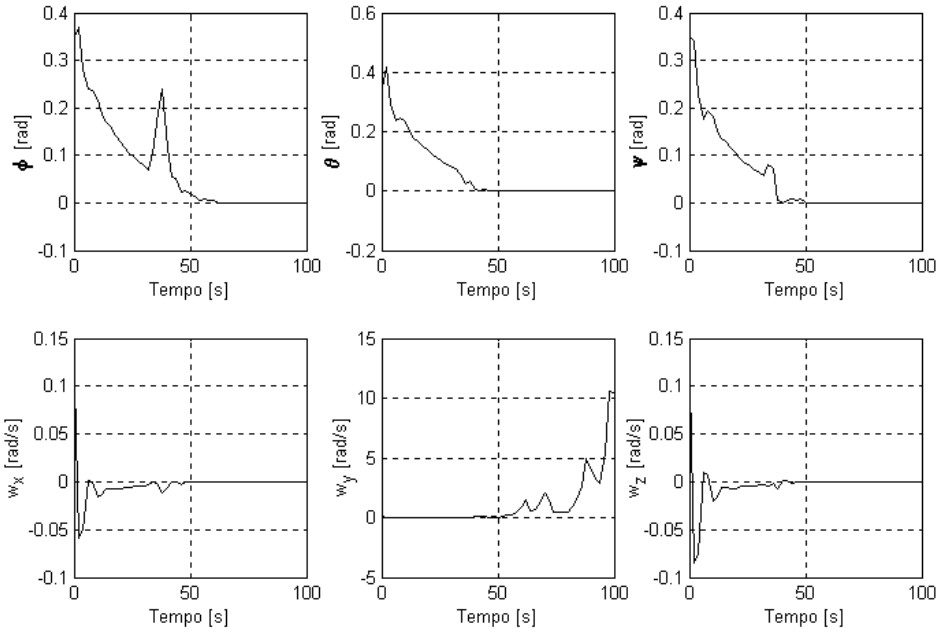


FIGURA 6.18 – Estrutura de controle preditivo com apenas o integrador e com ruído de 120% entre o modelo de trabalho e o modelo de validação a partir de t=30[s].

Condições Iniciais: $\phi = \theta = \varphi = 20^{o} = \frac{\pi}{180} \cdot 20$ [rad]	**Condições Iniciais:** $\omega_x = \omega_y = \omega_z = 1[\text{rpm}] = \frac{\pi}{30} \cdot 1$ [rad/s]
$t_o = 0$ [s] e $t_f = 100$ [s]	$\beta = 0.05$, $R_u = 10^{-2}$ e $R_y = 10^{-6}$
Δt_1 (Passo da Rede Neural)= Não foi utilizada	Δt_3 (Passo do Modelo de Validação) = $\frac{2}{50}$ [s] Runge-Kutta de Quarta Ordem
Δt_2 (Passo do Controle Preditivo)= 2 [s]	Iterações do Filtro de Kalman = 15 Iterações (em cada instante) e $\alpha = 0.15$

Variáveis de Estados:



Variáveis de Controles Estimados:

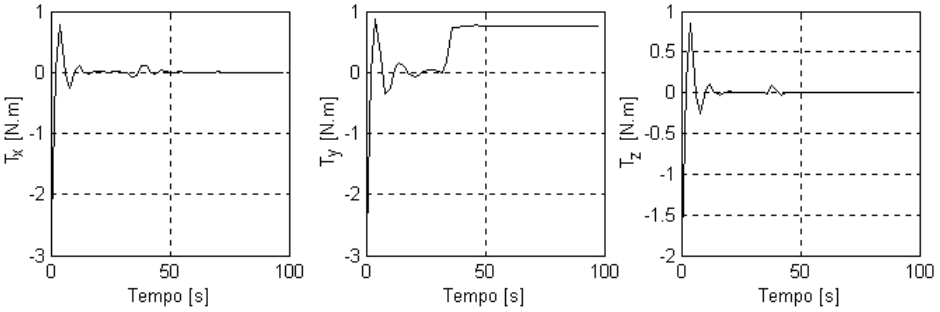
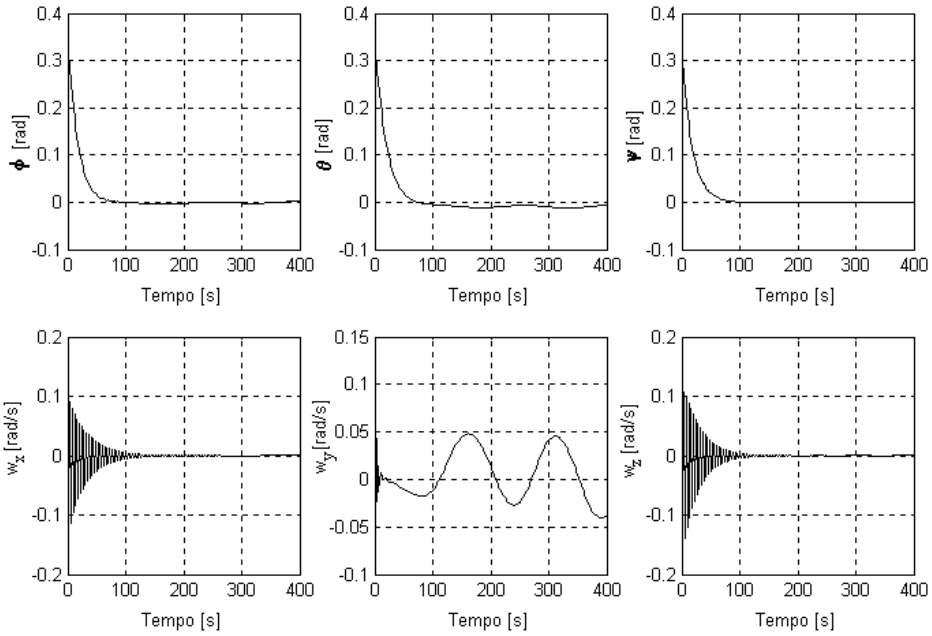


FIGURA 6.19 – Estrutura de controle preditivo com apenas o integrador e com ruído de 100% entre o modelo de trabalho e o modelo de validação a partir de t=30 [s].

Condições Iniciais: $\phi = \theta = \varphi = 20^{o} = \frac{\pi}{180} \cdot 20$ [rad]	**Condições Iniciais:** $\omega_x = \omega_y = \omega_z = 1[\text{rpm}] = \frac{\pi}{30} \cdot 1$ [rad/s]
$t_o = 0$ [s] e $t_f = 400$ [s]	$\beta = 0.05$, $R_u = 10^{-2}$ e $R_y = 10^{-6}$
Δt_1 (Passo da Rede Neural)= 0.4	Δt_3 (Passo do Modelo de Validação) = $\frac{2}{50}$ [s] Runge-Kutta de Quarta Ordem
Δt_2 (Passo do Controle Preditivo)= 2 [s]	Iterações do Filtro de Kalman = 15 Iterações (em cada instante) e $\alpha = 0.15$

Variáveis de Estado:



Variáveis de Controle:

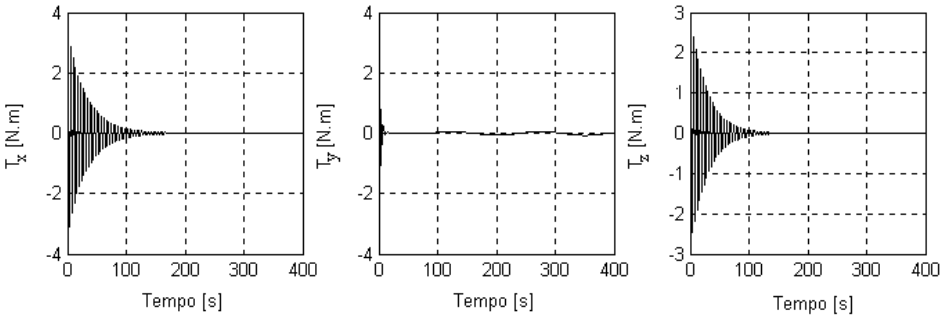
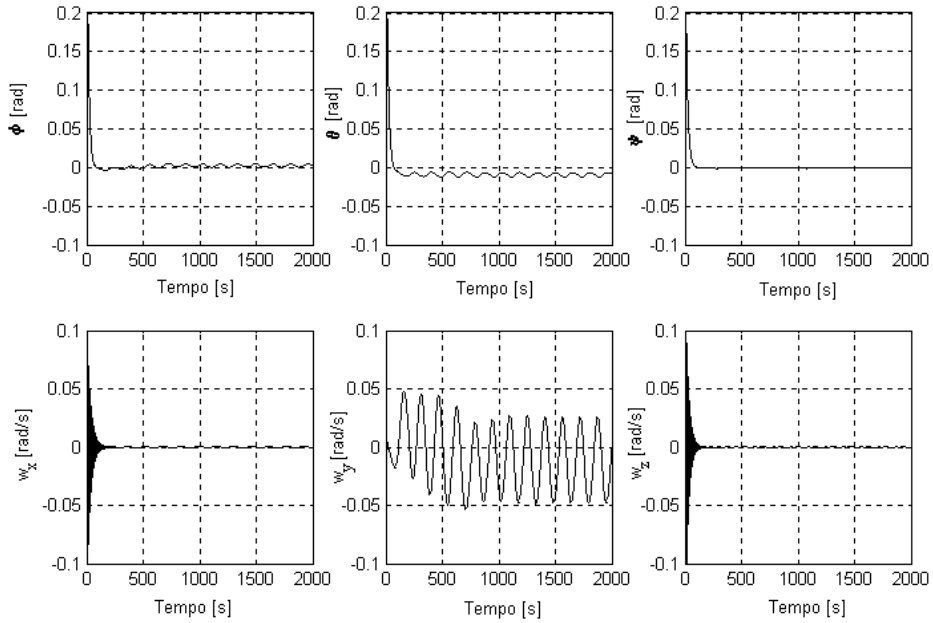


FIGURA 6.20 – Estrutura de controle preditivo com a rede neural trabalhando como modelo de trabalho e o integrador como modelo de validação.

Condições Iniciais: $\phi = \theta = \varphi = 20^{o} = \frac{\pi}{180} \cdot 20$ [rad]	**Condições Iniciais:** $\omega_x = \omega_y = \omega_z = 1[\text{rpm}] = \frac{\pi}{30} \cdot 1$ [rad/s]
$t_o = 0$ [s] e $t_f = 2000$ [s]	$\beta = 0.05$, $R_u = 10^{-2}$ e $R_y = 10^{-6}$
Δt_1 (Passo da Rede Neural)= 0.4	Δt_3(Passo do Modelo de Validação) = $\frac{2}{50}$ [s] Runge-Kutta de Quarta Ordem
Δt_2 (Passo do Controle Preditivo)= 2 [s]	Iterações do Filtro de Kalman = 15 Iterações (em cada instante) e $\alpha = 0.15$

Variáveis de Estado:



Variáveis de Controle:

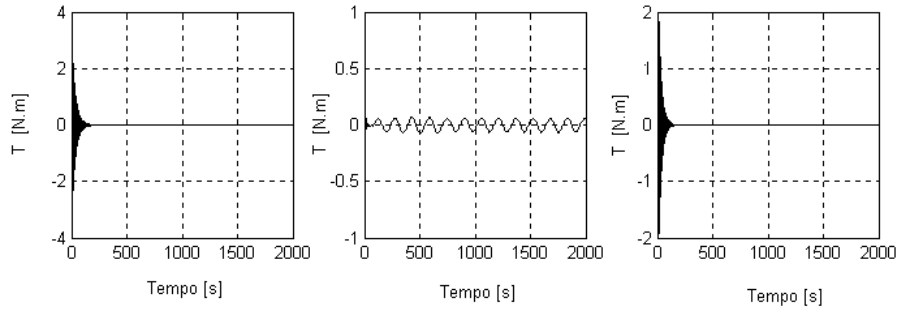


FIGURA 6.21 – Estrutura de controle preditivo com a rede neural trabalhando como modelo de trabalho e o integrador como modelo de validação.

6.3.1 Comentários e Conclusões.

As Figuras 6.20 e 6.21 demonstram que os resultados obtidos pela estrutura de controle preditivo neural apresentam uma oscilação ligeiramente afastada da condição nominal desejada. A Figura 6.21 destaca o fato que essa oscilação não pode ser amortecida pelo modelo de trabalho, representado pela rede neural, com o decorrer do tempo e como já observado anteriormente por Carrara (1997).

Há evidencias bastante fortes que isso se deve ao fato do EQM de aprendizado da rede neural saturar em torno de um valor ainda insuficiente para controlar sistemas dinâmicos que representam a atitude de satélites artificiais. O método de treinamento neural utilizado para fazer a rede *feedforward* aprender a dinâmica de atitude de um satélite foi o do filtro de Kalman com processamento paralelo e recursivo e com a heurística de se atualizar empiricamente as matrizes de covariâncias dos pesos da rede, como explicado no Capítulo 3. Mesmo após a melhora obtida no EQM com o emprego da atualização das matrizes de covariâncias dos pesos de treinamento, ainda assim, observou-se um aprendizado insuficiente conseguido pela rede. Treinar a rede neural com um erro ainda menor que o conseguido é uma tarefa do ponto de vista computacional inviável, pois o tempo requerido para isso seria muito grande. Solucionar esse problema de saturação da rede neural, torna-se então uma área de pesquisa bastante importante a ser explorada em trabalhos futuros.

CAPÍTULO 7

COMENTÁRIOS E CONCLUSÕES

Neste trabalho foi proposta, desenvolvida e testada a metodologia de se utilizar integradores numéricos trabalhando em conjunto com as redes neurais artificiais com arquitetura do tipo *feedforward* como modelo interno em controle preditivo. Embora, Wang e Lin (1998b) já tenham feito isso na tentativa bem sucedida de representar sistemas dinâmicos, neste trabalho estendeu-se a metodologia usando também derivadas médias e a ênfase foi na aplicação em teoria de controle e, em particular, nas estruturas de controle preditivo. Neste caso, a rede neural simplesmente faz o papel da função de derivadas do sistema dinâmico não-linear autônomo, mas com a principal vantagem de simplificar a fase computacional de treinamento neural. Todos os resultados numéricos e simulações obtidos neste trabalho, foram processados no MATLAB. Apesar de ter sido utilizados também os algoritmos de processamento paralelo do filtro de Kalman, essa implementação foi considerada apenas em nível de *software* e não em nível de *hardware*.

Entretanto, verificaram-se vários problemas de implementação computacional da metodologia com as derivadas instantâneas. Entre eles citam-se: dificuldades no desenvolvimento das expressões analíticas da retro-propagação dos estados futuros em relação aos controles atrasados e generalização do horizonte de controle na estrutura de controle preditivo; a avaliação de viabilidade de implementação *numérica* da retro-propagação em contraste com a analítica; e as determinações da ordem e do tipo do integrador a ser utilizado em conjunto com a rede neural. Todos esses problemas tiveram que ser explorados para a consolidação desta metodologia. Deste modo, esse trabalho que inicialmente era de caracter tecnológico transformou-se num trabalho de caracter teórico, computacional e tecnológico.

Desta maneira, para solucionar alguns dos problemas citados anteriormente é que foi desenvolvido o método das derivadas médias. Em primeiro lugar, este método tem a

vantagem de ser de primeira ordem, porém com uma precisão equivalente aos integradores de mais alta ordem. O fato do método das derivadas médias ser de primeira ordem e possibilitar uma precisão elevada tem-se uma série de vantagens, tais como: as expressões analíticas para os Jacobianos, exigidas durante o treinamento da rede neural e da aplicação da estrutura de controle preditivo, resultam ser bem mais simples e bem mais fáceis de serem obtidas e aplicadas do que aquelas exigidas pelos integradores de mais elevada ordem, além de exigir um tempo de processamento computacional menor. O único inconveniente deste método é que ele é de passo fixo, ou seja, querer alterar o passo de integração deste método exige o treinamento de uma nova rede neural. A Tabela 7.1 registra as conclusões quanto às características das alternativas derivadas instantâneas e derivadas médias. Observe que não há uma única regra mágica que permita realizar esta escolha, mas sim um conjunto de informações consistentes a ser consideradas para cada caso em particular.

TABELA 7.1 – Principais características dos tipos de integrador neural.

Método das Derivadas Instantâneas	**Método das Derivadas Médias**
Todos os tipos de integradores são aplicados a ele.	Basta o integrador do tipo Euler
Complexidades *analíticas* na retro-propagação ainda a serem superadas	Retro-propagação *analítica* já bem definida
Retro-propagação *numérica* aparentemente satisfatória	Retro-propagação *numérica* equivalente a analítica
Passo de integração *variável*	Passo de Integração *fixo*
Processamento mais *lento*, pois possui expressões analíticas mais complicadas	Processamento mais *rápido*, pois possui expressões analíticas mais simples
Controle estimado *aparentemente* mais suave	Controle estimado *aparentemente* menos suave
Assume-se que o sistema dinâmico é de primeira ordem	Pode-se trabalhar com entradas atrasadas

Os testes numéricos reforçaram os seguintes aspectos características da modelagem com uma estrutura neural de integração numérica:

1) é uma tarefa mais simples treinar uma rede neural *feedforward* para aprender uma função algébrica estática de um sistema dinâmico *Ordinary Differential Equation* (ODE), do que treiná-la para aprender um modelo discreto do tipo NARMAX; a rede neural inserida em um integrador ODE resultou ser mais simples, em termos da necessidade dos números de neurônios, uma vez que ela não tem que aprender a lei da dinâmica, mas somente a função de derivadas;

2) a utilização de uma rede neural num modelo de sistema dinâmico discreto naturalmente permitirá a implementação de esquemas de controle adaptativos, por explorar a capacidade de aprendizado da rede neural;

O uso de integradores numéricos neurais e de filtragem de Kalman em controle preditivo implicou que:

1) a interpretação estocástica dos erros dá mais realismo ao tratamento do problema, e facilita o ajuste das matrizes de pesos num funcional de controle preditivo;

2) a versão do processamento paralelo local do algoritmo do filtro de Kalman para estimar as ações de controle tem uma eficácia equivalente àquela correspondente ao algoritmo do filtro de Kalman aplicado ao treinamento neural, como esperado, uma vez que eles são algoritmos completamente similares utilizados para resolver numericamente problemas de estimação de parâmetros;

3) somente um passo a frente foi suficiente para rastrear o horizonte de controle, embora com mais horizontes a frente o modelo também tenha sido testado; esta característica juntamente com a performance dos algoritmos de processamento paralelo, garante a aplicabilidade em tempo real.

Finalmente, é importante notar que mesmo em situações onde um modelo matemático ODE não está disponível, a estrutura de integrador numérico com uma rede *feedforward* no lugar da função de derivadas pode ser treinada para obter um modelo interno discreto em esquemas de controle. Note também que poder-se-ia utilizar diretamente um

integrador numérico ODE como um modelo discreto do sistema dinâmico em estruturas de controle preditivo; entretanto neste caso não se teria a facilidade inerente de uma rede neural em estruturas de controle adaptativo para melhorar o desempenho do sistema de controle em tempo real.

Por outro lado, há ainda algumas tarefas que podem ser desenvolvidas futuramente na tentativa de aprimorar este trabalho. Entre elas, citam-se:

1) deduzir as expressões analíticas da retro-propagação numa estrutura de controle preditivo para integradores de alta ordem. Por exemplo, procurar expressões analíticas da retro-propagação para uma gama bem ampla de integradores neurais de simples e múltiplos passos e verificar uma possível ordem de formação para essas expressões analíticas. Há bastante trabalho algébrico nesta tarefa que deve ser analisado com cuidado, pois as expressões numéricas da retro-propagação são bem mais simples, genéricas e precisas. Entretanto, excelentes resultados analíticos já foram obtidos neste sentido, como, por exemplo, o jacobiano para a estrutura de integração neural *feedforward* do tipo Runge-Kutta de quarta ordem (Wang e Lin, 1998b) para o treinamento de sistemas dinâmicos, a inserção das estruturas de integração neural na teoria de controle (Rios Neto, 2001) e o jacobiano da estrutura de controle preditivo para o método das derivadas médias, este último, desenvolvido neste trabalho;

2) desenvolver estruturas de integração neural utilizando outros tipos de arquitetura neural, além das redes *feedforward*;

3) verificar se a função analítica que determina a arquitetura de uma rede *feedforward* é integrável, pois neste caso evitar-se-ía a utilização de integradores numéricos e consequentemente os erros globais de propagação das soluções obtidas numéricamente;

4) resolver muitos outros problemas práticos para o desenvolvimento e amadurecimento dos problemas tecnológicos de controle possíveis de serem resolvidos através da metodologia apresentada aqui.

REFERÊNCIAS BIBLIOGRÁFICAS

Ames, W. F. *Numerical methods for parcial differential equations*. 2. ed., New York: Academic Press, 1977.

Apolloni, B.; Battini, F.; Lucisano, C. A co-operating neural approach for spacecrafts attitude control. *Neurocomputing*, v. 16, n. 4, p. 279-307, Sep.1997.

Balakrishnan, S. N.; Weil, R. D. Neurocontrol: a literature survey. *Mathl. Comput. Modelling,* v. 23, n. 1-2, p. 101-107, Jan.1996.

Basak, J.; Pal, N. R.; Pal, S. K. A connectionist system for learning and recognition of structures: application to handwritten characters. *Neural Networks*, v. 8, n. 4, p. 643-652, 1995.

Bassoe, C.-F. Automated diagnoses from clinical narratives: a medical system based on computerized medical records, natural language processing, and neural network technology. *Neural Networks*, v. 8, n. 2, p. 313-319, 1995.

Benne, M.; Grondin-Perez, B.; Chabriat, J.-P.; Hervé, P. Artificial neural networks for modelling and predictive control of an industrial evaporation process. *Journal of Food Engineering*, v. 46, n. 4, p. 227-234, Dec. 2000.

Blanco, A.; Delgado, M.; Pegalajar, M. C. A genetic algorithm to obtain the optimal recurrent neural network. *International Journal of Aproximate Reasoning*, v. 23, n. 1, p. 67-83, Jan. 2000.

Botto, M. A.; da Costa, J. S. A comparison of nonlinear predictive control techniques using neural network models. *Journal of Systems Architecture*, v. 44, n. 8, p. 597-616, 1998.

Braga, A. P.; Ludermir, T. B.; Carvalho, A. C. P. L. F. *Redes neurais artificiais teoria e aplicações*. Rio de Janeiro: LTC (Livros Técnicos Científicos), 2000.

Braun, M. *Differential equations and their applications*: an introduction to applied mathematics. 3. ed., New York: Springer-Verlag, Applied Mathematical Sciences 15, 1983.

Carrara, V. *Redes neurais aplicadas ao controle de atitude de satélites com geometria variável*. 1997. 202 p. INPE-6384-TDI/603. Tese (Doutorado em Ciências Espacial) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 1997.

Carrara, V.; Rios Neto, A., Identificação de modelos dinâmicos de satélites com geometria variável através de redes neurais. In: Congresso Brasileiro de Engenharia Mecânica-COBEM, 14., 1997, Bauru, SP, código COB687. *Anais* ... Bauru: UNESP, 1997, p. 145-152 . 1 CD-ROM.

Carrara, V. ; Rios Neto, A. A neural network satellite attitude controller with error based reference trajectory. *Número Especial do J. of the Braz. Soc. Mechanical Sciences*, v. 21, ISSN 0100-7386, p. 425-431, 1998.

Carrara, V.; Varotto, S. E. C.; Rios Neto, A. Satellite attitude control using multilayer perceptron neural networks**,** *American Astronautical Society - AAS*, Advances in the Astronautical Sciences, v. 100, Part 1, ISBN 0-87703-453-2, AAS 98-345, p. 565 – 579, 1998.

Carrara, V.; Rios Neto, A., Redes neurais para controle de atitude de satélites**.** In: Simpósio Brasileiro de Automação Inteligente-SBAI, 4., 1999, Escola Politécnica da USP- EPUSP, São Paulo, SP, ISBN 85-87469-01-0. *Anais* ... São Paulo: USP, 1999, p. 77-82**.**

Chandran, P. S. Comments on "comparative analysis of backpropagation and the extended kalman filter for training multilayer perceptrons*". IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 16, n. 8, p. 862-863, Aug. 1994.

Chen, S.; Billings, S. A. Neural networks for nonlinear dynamic system modelling and identification. *Int. J. Control*, v. 56, n. 2, p. 319-346, 1992.

Chiang, J.-H. A hybrid neural network model in handwritten word recognition. *Neural Networks*, v. 11, n.2, p. 337-346, Mar 1998.

Ciocoiu, I. B., RBF networks training using a dual extended Kalman filter. *Neurocomputing*, v. 48, n. 1-4, p. 609-622, Oct. 2002.

Clarke, D.W.; Mohtadi, C.; Tuffs, P. S. Generalized predictive control-part I. The basic algorithm. *The Journal of IFAC the International Federation of Automatic Control*. *Automatica*, v. 23, n. 2, p. 137-148, Mar 1987a.

Clarke, D.W.; Mohtadi, C.; Tuffs, P. S. Generalized predictive control-part II. Extensions and interpretations. *The Journal of IFAC the International Federation of Automatic Control. Automatica*, v. 23, n. 2, p. 149-160, Mar 1987b.

Cybenko, G. Continuous valued networks with two hidden layers are sufficient. *Technical Report*, Department of Computer Science, Tufts University,1988.

Economou, C. G.; Morari, M.; Palsson, B. O. Internal Model Control. *Ind. Eng. Chem. Process Des. Dev.,* v. 25, p. 403-411, 1986.

Ender, L.; Filho, R. M. Design of multivariable controller based on neural networks. *Computer and Chemical Engineering*, v. 24, n. 1-7, p. 937-943, July 2000.

Ferreira, P. M.; Faria, E. A.; Ruano, A. E. Neural network models in greenhouse air temperature prediction. *Neurocomputing*, v. 43, n. 1-4 ,p. 51-75, Mar 2002.

Gelb, A.; Kasper, J. F.; Nash, R. A.; Price, C. F.; Sutherland, A. A. *Applied optimal estimation*. 14. ed., Massachusetts: The M.I.T. Press, The Analytic Sciences Corporation, 1996.

Hagan, M. T.; Menhaj, M. B. Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks*, v. 5, n. 6, p. 989-993, Nov. 1994.

He, S.; Reif, K.; Unbehauen, R. Multilayer neural networks for solving a class of partial differencial equations. *Neural Networks*, v. 13, n. 3, p. 385-396, Apr. 2000.

Henrici, P. *Elements of Numerical Analysis*. New York: John Wiley & Sons, 1964.

Huang, D.; Van Cauwenberghe, A. R. Neural-network-based multiple feedback long-range predictive control. *Neurocomputing*, v. 18, n. 1-3 , p. 127-139, Jan. 1998.

Hornik, K.; Stinchcombe, M.; White, H. Multilayer feedforward networks are universal approximators. *Neural Networks*, v. 2, n. 5, p. 359-366, 1989.

Hunt, K. J.; Sbarbaro, D. Neural networks for nonlinear internal model control. *IEE Proceedings – D*, v. 138, n. 5, p. 431-438, Sep. 1991.

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, Nov. 1992.

Jarmulak, J.; Spronck, P.; Kerckhoffs, E. J. H. Neural networks in process control: model-based and reinforcement trained controllers. *Computers and Electronics in Agriculture*, v. 18, n. 1-2, p. 149-166, Aug. 1997.

Jazwinski, A. H. *Stochastic processes and filtering theory*. New York and London: Academic Press, 1970.

Kalman, R. E. A new approach to linear filtering and prediction problems. *Transaction of the ASME – Journal of Basic Engineering*, p. 35-45, Mar 1960.

Kaplan, M. H. *Modern spacecraft dynamics & control*. New York: John Wiley & Sons, 1976.

Kaplan, W. *Cálculo avançado*, 9. ed., São Paulo: Edgard Blücher Ltda., v. 1 e v. 2, 1998.

Kasparian, V.; Batur, C. Model reference based neural network adaptive controller. *ISA Transactions*, v. 37, n. 1, p. 21-39, Mar 1998.

Kim, S. K.; Kim, J. W.; Kim, H. J. On-line recognition of cursive Korean characters using neural networks. *Neurocompuring*, v. 10, n.3 , p. 291-305, Apr.1996.

Kovács, L.; Solt, Z. *Redes neurais artificiais. fundamentos e aplicações*. 2. ed., São Paulo: Edição acadêmica, Collegium Cognitivo, 1996.

Lambert, J. D. *Computational methods in ordinary differential equations*. New York: John Wiley & Sons, 1973.

Lapidus, L.; Seinfeld, J. H. *Numerical solution of ordinary differential equations*. New York and London: Academic Press, 1971.

Liu, G. P.; Kadirkamanathan, V.; Billings, S. A. Stable sequential identification of continuous nonlinear dynamical systems by growing radial basis function networks. *Int. J. Control*, v. 65, n.1, p. 53-69, 1996.

Liu, G. P.; Kadirkamanathan, V.; Billings, S. A. Predictive control for non-linear systems using neural networks. *Int. J. Control*, v. 71, n. 6, p. 1119-1132, 1998.

Loesch, C.; Sari, S. T. *Redes neurais artificiais fundamentos e modelos*. Blumenau: Editora da Furb, Brasil, 1996.

Mai-Duy, N.; Tran-Cong, T. Numerical solution of differential equations using multiquadric radial basis function networks. *Neural Networks*, v.14, n. 2, p. 185-199, Mar 2001.

Maybeck, P. S. *Stochastic models, estimation, and control*. New York: Academic Press, v. 1, 1979.

Mills, P. M.; Zomaya, A. Y.; Tadé, M. O. Adaptative model-based control using neural networks. *Int. J. Control*, v. 60, n. 6, p. 1163-1192, 1994.

Minsky, M. L.; Papert, S. A. *Perceptron*. Cambridge, MA: The MIT Press, 1969.

Munem, M. A.; Foulis, D. J. *Cálculo*. Rio de Janeiro: Editora JC, v. 1 e v. 2, 1978.

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

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

Nascimento Jr., C. L.; Yoneyama, T. *Inteligência artificial em controle e automação*. 1. ed., São Paulo: Edgard Blucher Ltda., Fapesp, 2000.

Nikravesh, M.; Farell, A. E.; Stanford, T. G. Dynamic neural network control for non-linear systems: optimal neural network structure and stability analysis. *Chemical Engineering Journal*, v. 68, n. 1, p. 41-50, July 1997.

Nilsson, N. J. *Learning Machines*. New York: McGraw-Hill, 1965.

Norgaard, M.; Ravn, O.; Poulsen, N. K.; Hansen, L. K. *Neural networks for modelling and control of dynamic systems*. London: Spring, 2000.

Papoulis, A. *Probabilit, random variables, and stochastic processes*. New York: McGraw-Hill, 1965.

Pennington, R. H. *Introductory computer methods and numerical analysis*. London: The Macmillan Company, 1970.

Rao, K. R. A *Review on numerical methods for inicial value problems*. São José dos Campos: INPE, Feb. 1984, 173 p. (INPE-3011-RPI/088).

Rimrott, F. P. J. *Introductory Attitude Dynamics*. New York : Springer-Verlag, 1989, Mechanical Engineering Series.

Rios Neto, A.; Kuga, H. K. Kalman filtering state noise adaptive estimation. In: Iasted Int. Conference in Telecom and Control, 1985, Rio de Janeiro, Brasil. *Anais* ... Rio de Janeiro: TELECON'85, 1985, p. 210-213.

Rios Neto, A.; Rama Rao, K. A stochastic approach to global error estimation in ODE multistep numerical integration. *Journal of Computational and Applied Mathematics*, v. 30, n. 3, p. 257-281, 1990.

Rios Neto, A. Stochastic optimal linear parameter estimation and neural nets training in systems modeling. *RBCM – J. of the Braz. Soc. Mechanical Sciences*, v. 19, n. 2, p. 138-146, 1997.

Rios Neto, A. Design of a Kalman filtering based neural predictive control method. In: Congresso Brasileiro de Automática – CBA, 13., 2000, Florianópolis, código 543/00. *Anais* ... Florianópolis: UFSC (Universidade Federal de Santa Catarina), 2000, p. 2130-2134. 1 CD-ROM.

Rios Neto, A. Dynamic systems numerical integrators in neural control schemes. In: Congresso Brasileiro de Redes Neurais, 5., 2001, Rio de Janeiro, RJ, Brasil. *Anais* ... Rio de Janeiro: Conselho Nacional de Redes Neurais, 2001, p. 85-88. 1 CD-ROM.

Rivals, I.; Personnaz, L. A recursive algorithm based on the extended Kalman filter for the training of feedforward Neural Models. *Neurocomputing*, v. 20, n. 1-3, p. 279-294, Aug. 1998.

Ruck, D. W.; Rogers, S. K.; Kabrisky, M.; Maybeck, P.S.; Oxley, M. E. Comparative analysis of backpropagation and the extended Kalman filter for training multilayer perceptrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 14, n. 6, p. 686-691, Jun. 1992.

Rumelhart, D. E.; Hilton, G. E.; Williams, R. J. Learning internal representations by error propagation. In: Rumelhart, D. E., McClelland, J. L.(ed). *Parallel data processing*. Cambridge, MA: M.I.T. Press, 1986. v. 1, cap. 8.

Sage A. P. *Optimum systems control*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1968.

Salvadori, M. G.; Baron, M. L. *Numerical Methods in Engineering*. Englewood Cliffs, NJ: Prentice-Hall, 1961.

Shah, S.; Palmieri, F.; Datum, M. Optimal filtering algorithms for fast learning in feedforward neural networks. *Neural Networks*, v. 5, n. 5, p. 779-787, 1992.

Shustorovich, A.; Thrasher, C. W. Neural network positioning and classification of handwritten characters. *Neural Networks*, v. 9, n. 4, p. 685-693, 1996.

Silva, J. A.; Rios Neto, A. Preliminary testing and analysis of an adaptative neural network training Kalman filtering algorithm. In: Brasilian Conference on Neural Networks, 4., 1999, São José dos Campos, SP, Brasil. *Anais* ... São José dos Campos: ITA, 1999, p. 988-999.

Silva, J. A.; Rios Neto, A. Neural predictive satellite attitude control based on Kalman filtering algorithms. In: International Symposium in Space Dynamics, CNES, 2000, Biarritz, França. *Anais* ... Biarritz: CNES (Centre National Detudes Spatiales), 2000, pp. 565-574.

Silva, J. A. *Controle preditivo utilizando redes neurais artificiais aplicado a veículos aeroespaciais*. 2001. 239 p. (INPE-8480-TDI/778). Tese (Doutorado em Ciências Espacial) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2001.

Silva, J. A.; Rios Neto, A. Neural predictive flight trajectory control based on Kalman filtering algorithms. In: Congresso Brasileiro de Engenharia Mecânica - COBEM 2001, 16., 2001, Uberlândia, MG, ISBN: 85-85769-06-8. *Anais* ... Uberlândia: UFB (Universidade Federal de Uberlândia), 2001. 1 CD-ROM.

Simon, D.; El-Sherief, H. Navegation satellite selection using neural networks. *Neurocompuring*, v. 7, n. 3, p. 247-258, Apr. 1995.

Simon, D. Training radial basis neural networks with the extended Kalman filter. *Neurocomputing*, v. 48, n. 1-4 , p. 455-475, Oct. 2002.

Sokolnikoff, I. S.; Redheffer, R. M. *Mathematics of physics and modern engineering*. 2. ed., Tokyo: McGraw-Hill Kogakusha, LTD., 1966.

Soloway, D.; Haley, P.J. *Neural generalized predictive control: A Newton-Raphson implementation*. Hampton, Virginia: NASA, 1997, 17 p. (NASA - Technical Memorandum 110244).

Soloway, D.; Haley, P.; Gold, B. *Real-time adaptative control using neural generalized predictive control*. Virginia: NASA Langley Research Center, 1998. (NASA – Technical Memorandum).

Sorense, P. H.; Norgaard, M.; Ravn, O.; Poulsen, N. K. Implementation of neural network based non-linear predictive control. *Neurocomputing*, v. 28, n. 1-3, p. 37-51, Oct. 1999.

Sotomayor Tello, J. M. *Lições de equações diferenciais ordinárias*. Rio de Janeiro: Projeto Euclides, 1979, Impa (Instituto de Matemática Pura e Aplicada) e CNPq.

Stengel, R. F. *Stochastic optimal control*. New York: John Waley, 1986.

Tan, Y.; Van Cauvenberghe, A. R. Otimization techniques for the design of a neural predictive controller. *Neurocomputing*, v.10, n. 1, p. 83-96, Jan. 1996.

Tan, Y.; Saif, M. Neural-networks-based nonlinear dynamic modeling for automotive engines. *Neurocomputing*, v. 30, n. 1-4, p. 129-142, Jan. 2000.

Varotto, S. E. C. Controle de atitude de satélites artificiais terrestres utilizando redes neurais artificiais. 1997. 199 p. (INPE-6745-TDI/635). Tese (Doutorado em Ciências Espacial) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 1997.

Vidyasagar, M. *Nonlinear systems analysis*. New Jersey: Prentice-Hall, Networks Series, Electrical Engineering Series , 1978.

Wang, Y.; Lin, C. A second-order learning algorithm for multilayer networks based on block hessian matrix. *Neural Networks*, v. 11, n. 9, p. 1607-1622, Dec. 1998a.

Wang, Y.-J.; Lin, C.-T. Runge-Kutta neural network for identification of dynamical systems in high accuracy. I*EEE Transactions On Neural Networks*, v. 9, n. 2, p. 294-307, Mar 1998b.

Werbos, P. J. *Beyond regression: new tools for prediction and analysis in the behavior sciences*. Ph.D. Thesis, Harvard University, Commitee on Applied Mathematics, 1974.

Wertz, J. R. *Spacecraft attitude determination and control*. London: D. Reidel, Astrophysics and Space Science Library, v. 73,1978.

Wilson, E. B. *Advanced Calculus*. New York: Dover Publications, 1958.

Zurada, J. M. *Introduction to Artificial Neural System*. St. Paul, MN, USA: West Pub. Co., 1992.

APÊNDICE A

EQUIVALÊNCIA MATEMÁTICA ENTRE OS MÉTODOS DE TREINAMENTO NEURAL

Neste apêndice pretende-se desenvolver matematicamente a relação existente entre os métodos do *gradiente* e do *filtro de Kalman estendido* utilizados no treinamento de uma rede neural. Como será visto a seguir, provar-se-á que o método do gradiente é um caso particular do filtro de Kalman estendido (Chandran, 1994). Inicialmente dar-se-á o desenvolvimento matemático do método do gradiente e, em seguida, o desenvolvimento analítico do filtro de Kalman estendido provando-se no final que o primeiro método é um caso particular do segundo.

No treinamento de uma rede neural o algoritmo do gradiente do erro (erroneamente conhecido como retro-propagação ou *backpropagation*) é sem dúvida alguma o mais difundido e utilizado. As razões para isso são muitas: tempo de processamento de cada iteração bastante reduzido, é de fácil implementação computacional e exige baixa memória computacional durante seu processamento. A desvantagem deste algoritmo é sua baixa velocidade de convergência; exigindo, muitas vezes, centenas ou até milhares de iterações para que um treinamento seja bem sucedido. O algoritmo do gradiente utiliza-se do critério de minimização do erro quadrático médio (Zurada, 1992) da rede para o ajuste dos pesos durante o processo de aprendizagem. No caso de um treinamento supervisionado seqüencial , a função que deve ser minimizada é o funcional da forma,

$$\mathbf{min\ J(t,i)} = \frac{\mathbf{1}}{\mathbf{2}} \cdot \mathbf{r^T(t,i) \cdot r(t,i)} \tag{1}$$

para,

$$\mathbf{r(t,i)} = \mathbf{y(t)} - \bar{\mathbf{y}}\mathbf{(t,i)} \qquad \text{para } \mathbf{t=1, 2, ..., p} \text{ e } \mathbf{i=1, 2, ..., I} \tag{2}$$

onde,

r(t, i) ... vetor de resíduos;

J(t, i) ... índice de desempenho quadrático do t-ésimo vetor de padrões de treinamento de saída desejado da rede na i-ésima iteração;

$\mathbf{y(t) = [y_1(t)\ y_2(t)\ ...\ y_{n_L}(t)]^T}$ ... t-ésimo vetor de padrões de treinamento de saída desejado da rede;

$\mathbf{\bar{y}(t,i) = [\bar{y}_1(t,i)\ \bar{y}_2(t,i)\ ...\ \bar{y}_{n_L}(t,i)]^T}$ ... saída da rede relacionada ao t-ésimo padrão de treinamento na i-ésima iteração;

i ... índice que caracteriza a natureza não-linear e iterativa do método de treinamento neural;

t ... índice que especifica o padrão de treinamento atual;

$\mathbf{n_L}$ ... número total de saídas da rede;

p ... número total de padrões de treinamento;

I ... número total de iterações realizadas pelo método.

O gradiente do funcional **J(t,i)** da camada **l** da rede é definido como:

$$\nabla \mathbf{J^l(t,i)} = \begin{bmatrix} \dfrac{\partial \mathbf{J(t,i)}}{\partial \mathbf{w^l_{11}}} & \cdots & \dfrac{\partial \mathbf{J(t,i)}}{\partial \mathbf{w^l_{1n_{l-1}+1}}} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial \mathbf{J(t,i)}}{\partial \mathbf{w^l_{n_k 1}}} & \cdots & \dfrac{\partial \mathbf{J(t,i)}}{\partial \mathbf{w^l_{n_l n_{l-1}+1}}} \end{bmatrix} \tag{3}$$

Para uma melhor compreensão dos índices presentes na matriz (3) ver a Figura 3.6 do Capítulo 3. O ajuste dos pesos é realizado na direção oposta ao do gradiente da função **J(t,i)** (e. g., Carrara, 1997), ou seja,

$$\mathbf{W^l(t+1,i) = W^l(t,i) - \alpha \cdot \nabla J^l(t,i)} \tag{4.a}$$

para,

$$\mathbf{W^l(1,1) = \overline{W}} \tag{4.b}$$

$$\mathbf{W^l(1,i+1) = W^l(p,i) - \alpha \cdot \nabla J^l(p,i)} \tag{4.c}$$

onde,

$\mathbf{0 < \alpha \leq 1}$ ... taxa de aprendizagem da rede.

A escolha de α geralmente é empírica. Valores muito alto de α podem causar divergência do método, por outro lado, valores pequenos se caracterizam por um aprendizado muito lento podendo acontecer que os pesos, oriundos da estimação, fiquem presos a um mínimo local. É importante observar que o gradiente $\nabla \mathbf{J^l(t,i)}$ é composto pelas derivadas parciais dos *erros quadráticos das saída da rede* em relação aos pesos. Entretanto, é conveniente expressar este gradiente em função das derivadas parciais em relação as *saídas da rede,* visto que, a retro-propagação, como desenvolvida no Capítulo 3, foi computada em relação somente a estas últimas derivadas. Isto pode ser realizado de maneira bastante simples, como segue:

$$\mathbf{J(t,i) = \frac{1}{2} \cdot r^T(t,i) \cdot r(t,i) = \frac{1}{2} \cdot \left\{ r_1(t,i) \ldots r_{n_L}(t,i) \right\} \cdot \begin{Bmatrix} r_1(t,i) \\ \vdots \\ r_{n_L}(t,i) \end{Bmatrix}}$$

$$= \frac{1}{2} \cdot \left[r_1^2(t,i) + \ldots + r_{n_L}^2(t,i) \right] \tag{5}$$

Empregando a regra da cadeia para diferenciar a equação (5) em relação ao peso $\mathbf{w^l_{jk}}$, tem-se:

$$\frac{\partial \mathbf{J^l(t,i)}}{\partial \mathbf{w^l_{jk}}} = \mathbf{r_1(t,i)} \cdot \frac{\partial \mathbf{r_1(t,i)}}{\partial \mathbf{w^k_{ij}}} + \ldots + \mathbf{r_{n_L}(t,i)} \cdot \frac{\partial \mathbf{r_{n_L}(t,i)}}{\partial \mathbf{w^k_{ij}}} \tag{6}$$

Substituindo a relação (2) na equação (6), vem:

$$\frac{\partial \mathbf{J^l(t,i)}}{\partial \mathbf{w^l_{jk}}} = \mathbf{r_1(t,i)} \cdot \frac{\partial \left[\mathbf{y_1(t) - \bar{y}_1(t,i)}\right]}{\partial \mathbf{w^k_{ij}}} + \ \mathbf{...} \ + \mathbf{r_{n_L}(t,i)} \cdot \frac{\partial \left|\mathbf{y_{nL}(t) - \bar{y}_{n_L}(t,i)}\right|}{\partial \mathbf{w^k_{ij}}} \tag{7}$$

Simplificando a equação (7), resulta:

$$\frac{\partial \mathbf{J^l(t,i)}}{\partial \mathbf{w^l_{jk}}} = -\mathbf{r_1(t,i)} \cdot \frac{\partial \, \mathbf{\bar{y}_1(t,i)}}{\partial \mathbf{w^k_{ij}}} - \ \mathbf{...} \ - \mathbf{r_{n_L}(t,i)} \cdot \frac{\partial \, \mathbf{\bar{y}_{n_L}(t,i)}}{\partial \mathbf{w^k_{ij}}} \tag{8.a}$$

Ou

$$\frac{\partial \mathbf{J^l(t,i)}}{\partial \mathbf{w^l_{jk}}} = \mathbf{-\, r(t,i)} \cdot \frac{\partial \, \mathbf{\bar{y}(t,i)}}{\partial \, \mathbf{w^l_{jk}}} \tag{8.b}$$

Onde o operador na equação (8.b) representa o *produto escalar* entre dois vetores. O vetor $\frac{\partial \, \mathbf{\bar{y}(t,i)}}{\partial \mathbf{w^l_{jk}}}$,presente na equação (8.b), pode ser calculado através da retro-propagação como desenvolvida no Capítulo 3. Substituindo a equação (8.b) na equação (4.a), resulta:

$$\Delta \mathbf{W^l(t+1,i)} = \mathbf{W^l(t+1,i) - W^l(t,i)} =$$

$$\alpha \cdot \begin{bmatrix} \mathbf{r(t,i)} \cdot \frac{\partial \mathbf{\bar{y}(t,i)}}{\partial \mathbf{w^1_{11}}} & \cdots & \mathbf{r(t,i)} \cdot \frac{\partial \mathbf{\bar{y}(t,i)}}{\partial \mathbf{w^1_{1n_{l-1}+1}}} \\ \vdots & \vdots & \vdots \\ \mathbf{r(t,i)} \cdot \frac{\partial \mathbf{\bar{y}(t,i)}}{\partial \mathbf{w^1_{n_l 1}}} & \cdots & \mathbf{r(t,i)} \cdot \frac{\partial \mathbf{\bar{y}(t,i)}}{\partial \mathbf{w^1_{n_l n_{l-1}+1}}} \end{bmatrix} \tag{9}$$

Lembrando que,

$$\mathbf{r(t,i)} \cdot \frac{\partial \, \mathbf{\bar{y}(t,i)}}{\partial \, \mathbf{w^l_{jk}}} = \sum_{\mathbf{r=1}}^{\mathbf{n_L}} \left[\mathbf{y_r(t) - \bar{y}_r(t,i)}\right] \cdot \frac{\partial \, \mathbf{\bar{y}_r(t,i)}}{\partial \, \mathbf{w^l_{jk}}} \tag{10}$$

pode-se então, expressar a equação (9) na forma escalar,

$$\Delta \mathbf{w}_{\mathbf{jk}}^{\mathbf{l}}(\mathbf{t}+\mathbf{1},\mathbf{i}) = \mathbf{w}_{\mathbf{jk}}^{\mathbf{l}}(\mathbf{t}+\mathbf{1},\mathbf{i}) - \mathbf{w}_{\mathbf{jk}}^{\mathbf{l}}(\mathbf{t},\mathbf{i})$$

$$= \alpha \cdot \sum_{\mathbf{r}=\mathbf{1}}^{\mathbf{n_L}} \left[\mathbf{y_r}(\mathbf{t}) - \overline{\mathbf{y}}_{\mathbf{r}}(\mathbf{t},\mathbf{i})\right] \cdot \frac{\partial \overline{\mathbf{y}}_{\mathbf{r}}(\mathbf{t},\mathbf{i})}{\partial \mathbf{w}_{\mathbf{jk}}^{\mathbf{l}}} \qquad (11)$$

A expressão (11) pode ser utilizada para implementar todas as atualizações de todos os pesos da rede neural. A seguir desenvolver-se-á uma expressão semelhante para o filtro de Kalman estendido podendo-se então, em seguida, verificar as semelhanças e diferenças existentes entre estes dois métodos.

As Figuras A.1 e A.2 representam o fluxograma do método do gradiente do erro para a atualização dos pesos de uma rede neural com uma camada interna. Como pode ser observado nestas figuras ele é bastante genérico pois, permite especificar o número de entradas e saídas da rede, além de permitir escolher o número de neurônios da camada interna. As funções de ativação podem ser tanto do tipo *logsig* como do tipo *tansig*.

Por outro lado, a abordagem do filtro de Kalman estendido para o treinamento de uma rede neural multi-camadas considera os pesos da rede como os estados de um sistema a ser estimado. Uma vez que os pesos não têm qualquer dinâmica este torna-se um problema de *estimação estático*. As equações dos estados e das medidas para este sistema pode ser escrita (Chandran, 1994) como:

$$\mathbf{w}^{\mathbf{l}}(t+\mathbf{1}) = \mathbf{w}^{\mathbf{l}}(t) \qquad (12.a)$$

$$\mathbf{y}(t) = \mathbf{f}[\mathbf{w}(t), \mathbf{x}(t)] + \mathbf{v}(t) \qquad (12.b)$$

onde,

w(t) ... vetor de pesos contendo todos os parâmetros de treinamento de todas as camadas da rede;

$\mathbf{w}^{\mathbf{l}}(\mathbf{i})$ ... vetor de pesos contendo todos os parâmetros da l-ésima camada da rede;

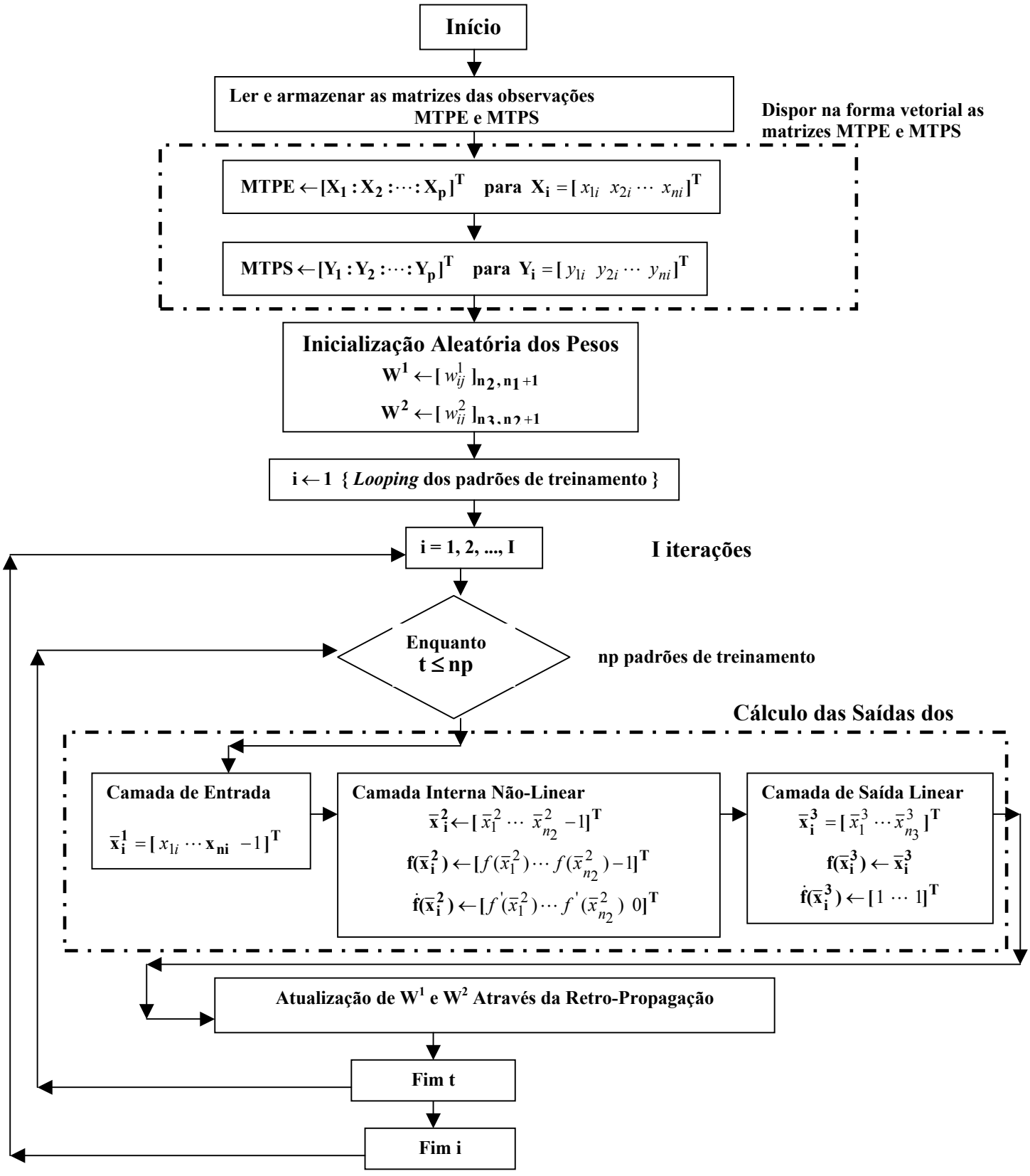


FIGURA A.1– Fluxograma para a atualização dos pesos de uma rede *feedforward* de uma camada interna e np padrões de treinamento em I iterações.

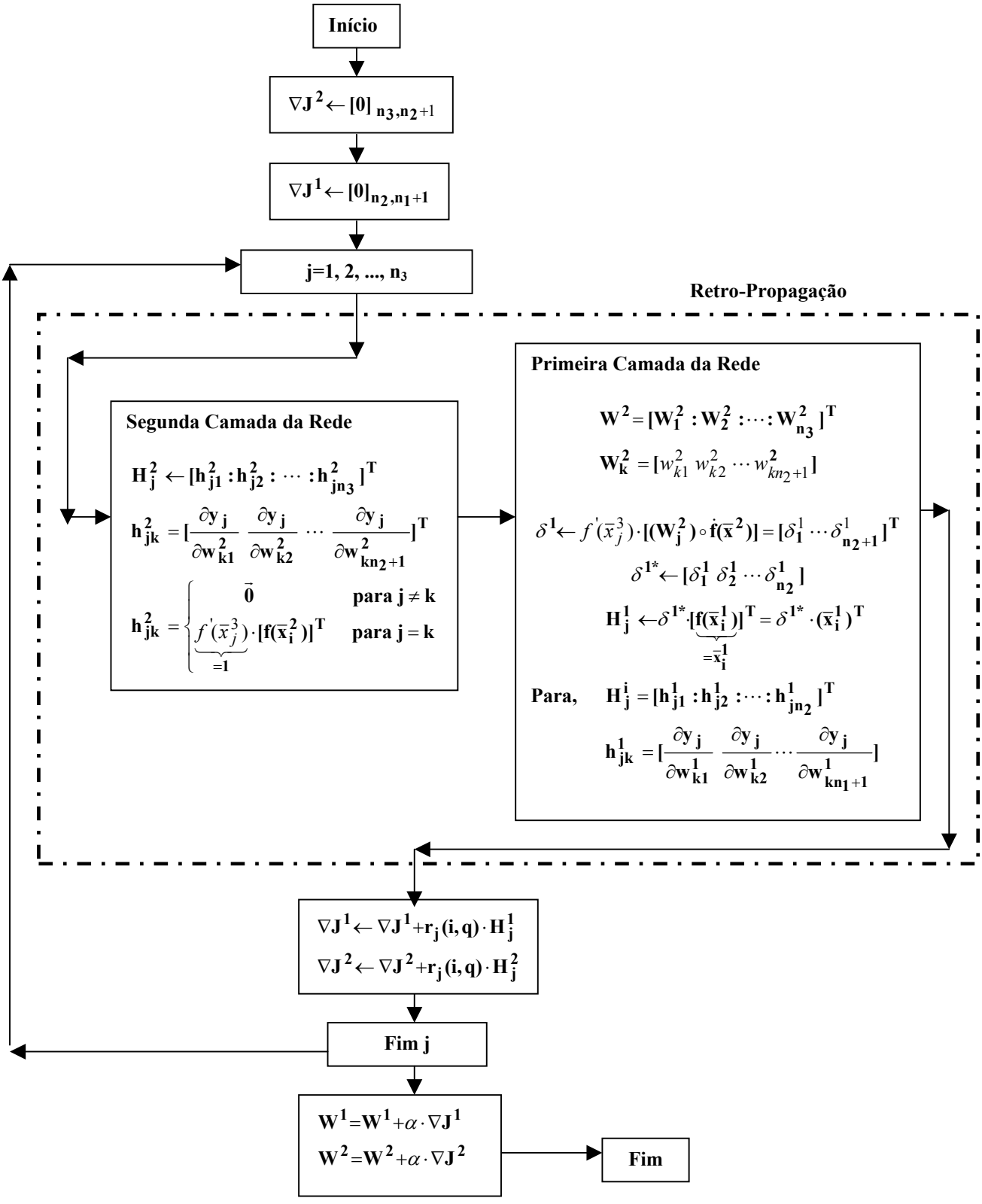


FIGURA A.2 – Detalhamento da retro-propagação e da atualização dos pesos nas camadas da rede utilizando o método do gradiente.

x ... vetor de padrões de entrada da rede;

y ... vetor de padrões desejados na saída da rede;

v ... seqüência de ruído Gaussiano branco com média nula e uma covariância de $\varepsilon \cdot \mathbf{I}$;

f[w(t),x(t)] ... função não-linear de mapeamento dos estados para gerar as saídas da rede, ou seja, ela descreve a rede;

t ...índice do tempo.

A forma *standard* para as equações do filtro de Kalman estendido na versão *full* em lotes que representa a solução para o sistema descrito em (12.a) e (12.b) são (e.g., Stengel, 1986):

$$\hat{\mathbf{w}}^{\mathbf{l}}(\mathbf{i}) = \overline{\mathbf{w}}^{\mathbf{l}}(\mathbf{i}) + \mathbf{k}(\mathbf{i}) \cdot [\mathbf{z}(\mathbf{i}) - \mathbf{H}(\mathbf{i}) \cdot \overline{\mathbf{w}}^{\mathbf{l}}(\mathbf{i})] \tag{13.a}$$

$$\mathbf{k}(\mathbf{i}) = \overline{\mathbf{P}}^{\mathbf{l}} \cdot \mathbf{H}^{\mathbf{T}}(\mathbf{i}) \cdot [\mathbf{H}(\mathbf{i}) \cdot \overline{\mathbf{P}}^{\mathbf{l}} \cdot \mathbf{H}^{\mathbf{T}}(\mathbf{i}) + \mathbf{R}]^{-\mathbf{1}} \tag{13.b}$$

$$\overline{\mathbf{w}}(\mathbf{i}+\mathbf{1}) = \hat{\mathbf{w}}(\mathbf{i}) \tag{13.c}$$

onde,

$$\alpha(\mathbf{i}) \leftarrow \alpha(\mathbf{i}+\mathbf{1}) \tag{13.d}$$

$$\mathbf{z}(\mathbf{i}) = \alpha(\mathbf{i}) \cdot [\mathbf{y} - \overline{\mathbf{y}}(\mathbf{i})] + \mathbf{H}(\mathbf{i}) \cdot \overline{\mathbf{w}}^{\mathbf{l}}(\mathbf{i}) \tag{13.e}$$

$$\mathbf{H}(\mathbf{i}) = \hat{\mathbf{f}}_{\mathbf{w}^{\mathbf{l}}}[\mathbf{x}, \overline{\mathbf{w}}(\mathbf{i})] \tag{13.f}$$

Após **I** iterações sucessivas a solução em lotes do estimador será,

$$\hat{\mathbf{w}}^{\mathbf{l}} = \mathbf{w}^{\mathbf{l}}(\mathbf{I}) \tag{14.a}$$

$$\mathbf{P}^{\mathbf{l}} = [\mathbf{I} - \mathbf{K}(\mathbf{I}) \cdot \mathbf{H}(\mathbf{I})] \cdot \overline{\mathbf{P}}^{\mathbf{l}} \tag{14.b}$$

Para demonstrar matematicamente a relação existente entre o filtro de Kalman e o método do gradiente é conveniente escrever a equação do ganho de Kalman na forma alternativa (e.g., Jacobs, 1974) como segue,

$$\mathbf{K(i)} = \mathbf{P^l} \cdot \mathbf{H^T(i)} \cdot (\varepsilon \cdot \mathbf{I})^{-1} \tag{15.a}$$

$$\mathbf{P^l} = \overline{\mathbf{P}}^{\mathbf{l}} - \overline{\mathbf{P}}^{\mathbf{l}} \cdot \mathbf{H^T(i)} \cdot [\mathbf{H(i)} \cdot \overline{\mathbf{P}}^{\mathbf{l}} \cdot \mathbf{H^T(i)} + \varepsilon \cdot \mathbf{I}]^{-1} \cdot \mathbf{H(i)} \cdot \overline{\mathbf{P}}^{\mathbf{l}} \tag{15.b}$$

Partindo-se da forma alternativa para o ganho de Kalman **k(i)** pode-se demonstrar que a única condição a ser satisfeita para reduzir o filtro de Kalman estendido ao método do gradiente é fazer:

$$\mathbf{P} = \mathbf{p} \cdot \mathbf{I} = \begin{cases} \mathbf{p_{ij}} = \mathbf{p} & \mathbf{se \;\; i = j} \\ \mathbf{p_{ij}} = \mathbf{0} & \mathbf{se \;\; i \neq j} \end{cases} \tag{16}$$

Para verificar isso inicialmente desenvolve-se a expressão para a obtenção de $\Delta \mathbf{w^l(i)}$. Partindo-se da equação (13.a), vem:

$$\Delta \mathbf{w^l(i)} = \hat{\mathbf{w}}^{\mathbf{l}}\mathbf{(i)} - \overline{\mathbf{w}}^{\mathbf{l}}\mathbf{(i)} = \mathbf{k(i)} \cdot \underbrace{[\mathbf{z(i)} - \mathbf{H(i)} \cdot \overline{\mathbf{w}}^{\mathbf{l}}\mathbf{(i)}]}_{\alpha\mathbf{(i)} \cdot [\mathbf{y} - \overline{\mathbf{y}}\mathbf{(i)}]}$$

$$= \mathbf{P} \cdot \mathbf{H^T(i)} \cdot [\mathbf{y} - \overline{\mathbf{y}}\mathbf{(i)}] \cdot \alpha\mathbf{(i)} \tag{17}$$

A equação anterior na forma expandida resulta em,

$$\begin{Bmatrix} \Delta \mathbf{w_1^l(i)} \\ \Delta \mathbf{w_2^l(i)} \\ \vdots \\ \Delta \mathbf{w_q^l(i)} \end{Bmatrix} = \begin{bmatrix} \mathbf{p_{11}} & \cdots & \mathbf{p_{1q}} \\ \vdots & \ddots & \vdots \\ \mathbf{p_{q1}} & \cdots & \mathbf{p_{qq}} \end{bmatrix} \cdot \begin{bmatrix} \dfrac{\partial \overline{\mathbf{y}}_{\mathbf{1}}\mathbf{(i)}}{\partial \overline{\mathbf{w}}_{\mathbf{1}}\mathbf{(i)}} & \cdots & \dfrac{\partial \overline{\mathbf{y}}_{\mathbf{np}}\mathbf{(i)}}{\partial \overline{\mathbf{w}}_{\mathbf{1}}\mathbf{(i)}} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial \overline{\mathbf{y}}_{\mathbf{1}}\mathbf{(i)}}{\partial \overline{\mathbf{w}}\mathbf{q(i)}} & \cdots & \dfrac{\partial \overline{\mathbf{y}}_{\mathbf{np}}\mathbf{(i)}}{\partial \overline{\mathbf{w}}_{\mathbf{q}}\mathbf{(i)}} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{y_1} - \overline{\mathbf{y}}_{\mathbf{1}}\mathbf{(i)} \\ \mathbf{y_2} - \overline{\mathbf{y}}_{\mathbf{2}}\mathbf{(i)} \\ \vdots \\ \mathbf{y_{np}} - \overline{\mathbf{y}}_{\mathbf{np}}\mathbf{(i)} \end{bmatrix} \tag{18}$$

Efetuando o primeiro produto matricial, tem-se:

$$\begin{Bmatrix} \Delta w_1^l(i) \\ \Delta w_2^l(i) \\ \vdots \\ \Delta w_q^l(i) \end{Bmatrix} = \begin{bmatrix} p_{11} \cdot \frac{\partial \bar{y}_1(i)}{\partial \bar{w}_1(i)} + \cdots + p_{1q} \cdot \frac{\partial \bar{y}_1(i)}{\partial \bar{w}_q(i)} & \cdots & p_{11} \cdot \frac{\partial \bar{y}_{np}(i)}{\partial \bar{w}_1(i)} + \cdots + p_{1q} \cdot \frac{\partial \bar{y}_{np}(i)}{\partial \bar{w}_q(i)} \\ \vdots & \ddots & \vdots \\ p_{q1} \cdot \frac{\partial \bar{y}_1(i)}{\partial \bar{w}_1(i)} + \cdots + p_{qq} \cdot \frac{\partial \bar{y}_1(i)}{\partial \bar{w}_q(i)} & \cdots & p_{q1} \cdot \frac{\partial \bar{y}_{np}(i)}{\partial \bar{w}_1(i)} + \cdots + p_{qq} \cdot \frac{\partial \bar{y}_{np}(i)}{\partial \bar{w}_q(i)} \end{bmatrix} \cdot \begin{bmatrix} y_1 - \bar{y}_1(i) \\ y_2 - \bar{y}_2(i) \\ \vdots \\ y_{np} - \bar{y}_{np}(i) \end{bmatrix} \tag{19}$$

Na forma escalar a equação (19) assume a forma mais compacta:

$$\Delta w_j^l(i) = \alpha(i) \cdot \sum_{k=1}^{q} p_{jk} \cdot \left\{ \sum_{r=1}^{np} [y_r - \bar{y}_r(i)] \cdot \frac{\partial \bar{y}_r(i)}{\partial \bar{w}_k(i)} \right\} \text{ para } j=1, 2, ..., q \tag{20}$$

Reagrupando os termos da equação anterior, vem:

$$\Delta w_j^l(i) = \alpha(i) \cdot p_{jj} \cdot \sum_{r=1}^{np} [y_r - \bar{y}_r(i)] \cdot \frac{\partial \bar{y}_r(i)}{\partial \bar{w}_j(i)} + \tag{21}$$

$$\alpha(i) \cdot \sum_{k=1, k \neq j}^{q} p_{jk} \cdot \left\{ \sum_{r=1}^{np} [y_r - \bar{y}_r(i)] \cdot \frac{\partial \bar{y}_r(i)}{\partial \bar{w}_k(i)} \right\}$$

Por hipótese, a equação (16) estabelece que $p_{jk} = 0$ para $j \neq k$. Assim, o segundo termo do lado direito da equação (21) se anula, chegando-se então, a seguinte equação:

$$\Delta w_j^l(i) = \alpha(i) \cdot p_{jj} \cdot \sum_{r=1}^{np} [y_r - \bar{y}_r(i)] \cdot \frac{\partial \bar{y}_r(i)}{\partial \bar{w}_j(i)} \tag{22}$$

A expressão (22) é idêntica aquela dada pela equação (9). Logo, está demonstrado que o método do gradiente é um caso particular do filtro de Kalman estendido quando se faz a matriz $P = p \cdot I$ .Para finalizar este apêndice é apresentado uma noção sobre as performances computacional e adaptativa dos dois métodos de treinamento neural apresentados anteriormente. Entende-se por performance computacional o tempo de processamento de cada iteração de um determinado algoritmo de treinamento neural. Performance adaptativa é a capacidade de aprendizado de um algoritmo em cada

iteração. Em geral a performance adaptativa decai exponencialmente com o avanço das iterações em qualquer algoritmo de treinamento.

A Figura A.3 apresenta um gráfico referente a performance computacional entre os métodos do gradiente e do filtro de Kalman. Este gráfico apresenta um caso particular do treinamento de uma rede *feedforward* com uma camada interna. Para todos os treinamento apresentados neste gráfico a rede foi sempre treinada com cinco entradas e cinco saídas.

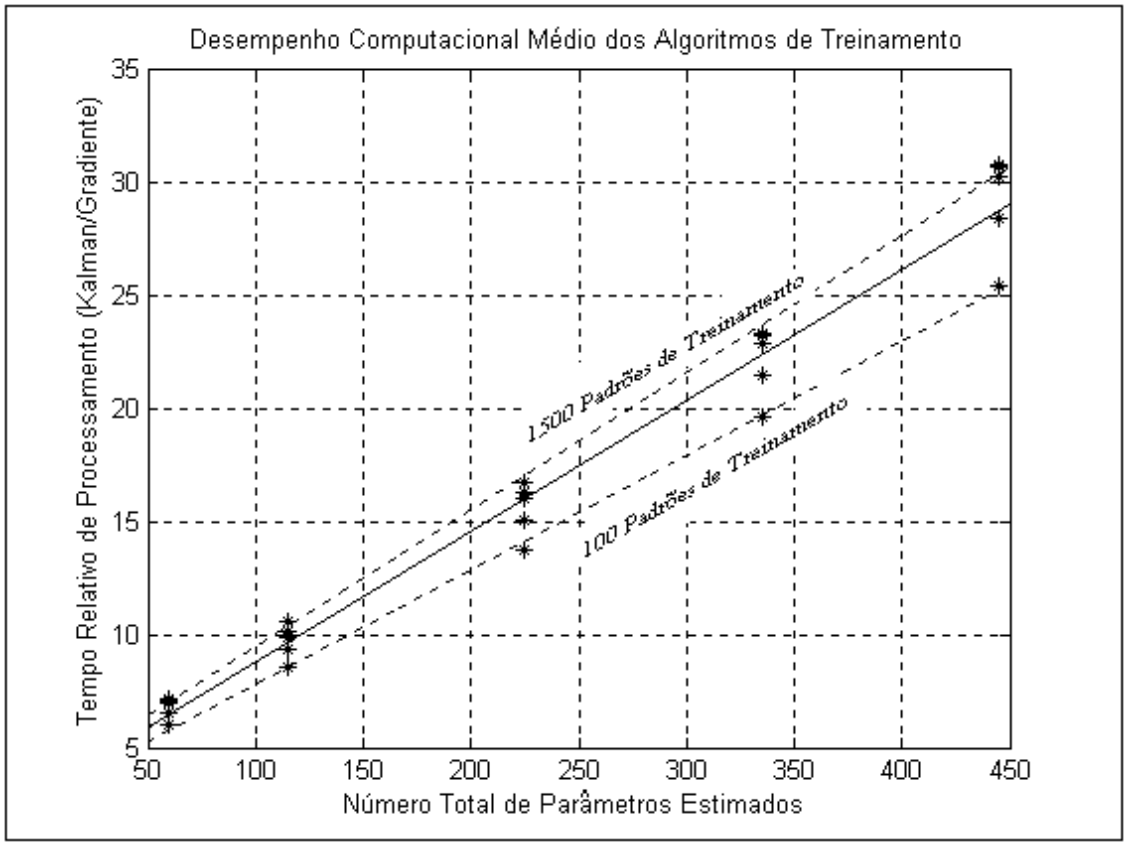


FIGURA A.3 – Tempo de processamento relativo de cada iteração entre os métodos do filtro de Kalman e do gradiente.

Este gráfico apresenta a relação média existente entre o tempo de processamento do algoritmo do filtro de Kalman em relação ao do gradiente. Por exemplo, uma rede com um número total de 450 parâmetros a serem estimados e com aproximadamente 1500 padrões de treinamento, o algoritmo do filtro de Kalman com processamento paralelo será cerca de 30 vezes mais lento que o algoritmo do gradiente. A performance

adaptativa não é exibida no gráfico acima, mas em geral ela é mais significativa para o filtro de Kalman, compensando assim, sua baixa performance computacional. O gráfico da Figura A.3 foi obtido utilizando-se um computador Pentium III de 500 MHz e com 376 MB de RAM.

O método do gradiente pode dar passos grandes no início do treinamento, mas no decorrer das iterações, quando o algoritmo estiver próximo de um mínimo, o método do gradiente pode ficar oscilando em torno da solução ótima. Por outro lado, o método do filtro de Kalman consegue resultados mais precisos, entretanto este método obtêm a solução ótima gradativamente no decorrer das iterações devido a hipótese de linearização. Para evitar a presença de um mínimo local, em geral e independentemente do método de treinamento utilizado, reinicia-se o aprendizado com uma quantidade maior de neurônios na camada interna.

APÊNDICE B

FLUXOGRAMA DA FILTRAGEM DE KALMAN COM PROCESSAMENTO PARALELO E RECURSIVO

O fluxograma de detalhamento do filtro de Kalman com processamento paralelo para o treinamento de uma rede *feedforward* é apresentado nesta seção. A Figura B.1 apresenta um esquema simplificado deste algoritmo. Ele é quase que uma extensão natural do fluxograma apresentado no Capítulo 2 para estimadores não-lineares. Entretanto, há duas diferenças básicas apresentadas aqui: primeiro que este algoritmo é específico para o caso de uma rede *feedforward* com uma camada interna e segundo que ele apresenta o processamento paralelo em sua estrutura.

Ainda com relação a Figura B.1 existem dois *loops* que contêm: o cálculo do fluxo de sinais dos neurônios da rede (item a), o cálculo da retro-propagação (item b) e a aplicação do filtro de Kalman (item c). Os itens a e b são detalhados na Figura B.2 e o item c é detalhado na Figura B.3. Na Figura B.2 repare que tanto o fluxo de sinais das saídas dos neurônios como o cálculo da retro-propagação são realizados para uma rede *feedforward* com uma camada interna não-linear e uma camada de saída linear. O número de entradas da rede, o número de neurônios da camada interna e o número de saídas da rede são dadas, respectivamente, por $\mathbf{n_1}$, $\mathbf{n_2}$ e $\mathbf{n_3}$. O cálculo da retro-propagação é baseado nas equações de (10.a) à (10.c) do Capítulo 3.

A Figura B.3 ilustra a aplicação propriamente dita da filtragem de Kalman com processamento paralelo e recursivo. A parte superior deste fluxograma apresenta a aplicação do filtro de Kalman na camada de saída da rede e a parte inferior deste fluxograma apresenta a aplicação do filtro de Kalman na camada interna da rede. Observe que aqui são utilizados os graus de paralelismo $\mathbf{g_1}$ e $\mathbf{g_2}$. Por exemplo, se $\mathbf{g_1}$=1 então, o filtro de Kalman é aplicado a cada um dos neurônios da camada interna, se $\mathbf{g_1}$=2 então, o filtro de Kalman será aplicado a cada dois neurônios da camada interna e assim por diante. Repare que se $\mathbf{g_1}$=$\mathbf{n_2}$ então, a versão paralela se confunde com a própria versão *full*. O mesmo raciocínio é válido para o grau de paralelismo $\mathbf{g_2}$.

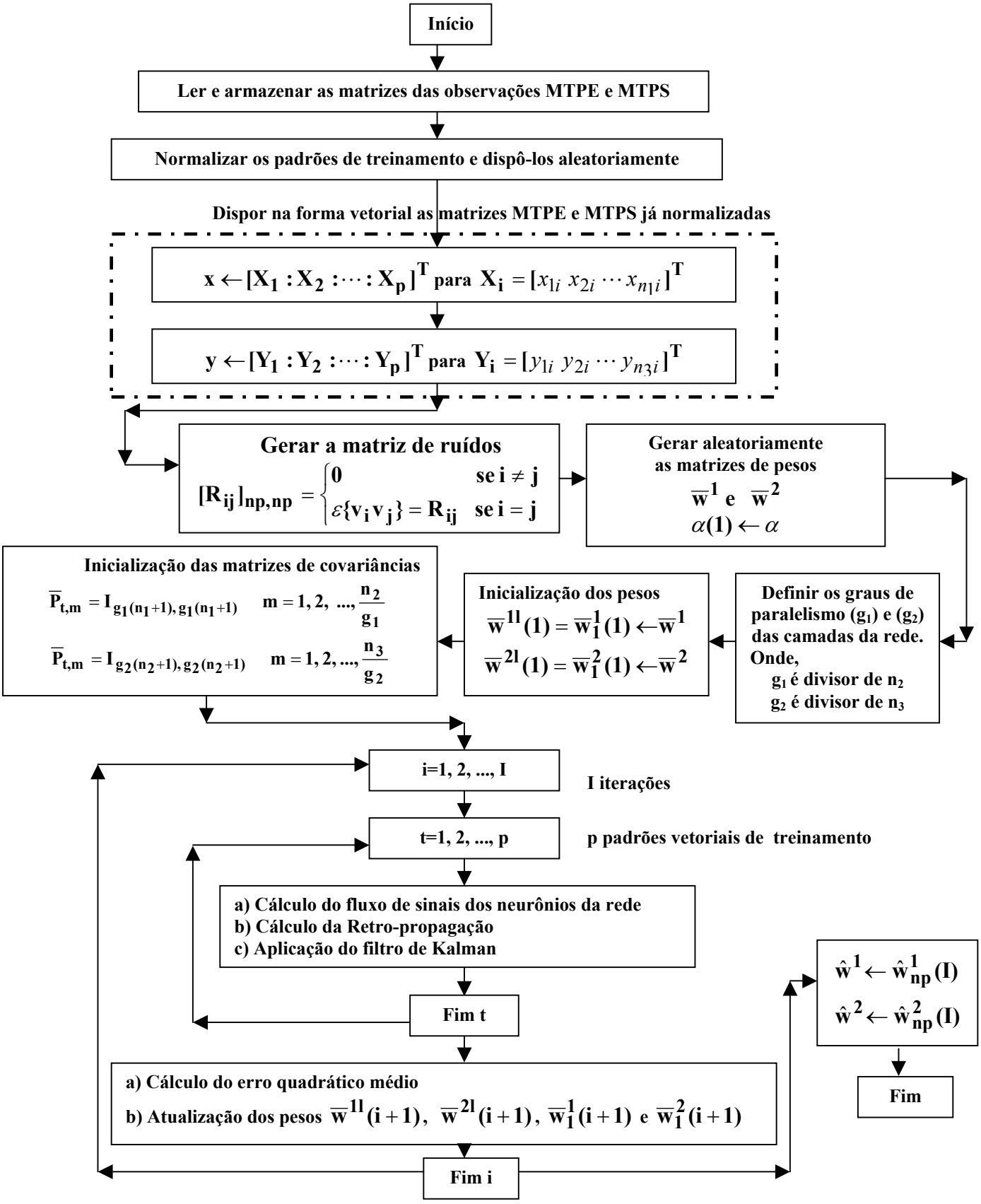


FIGURA B.1 - Fluxograma básico para o algoritmo da filtragem de Kalman com processamento paralelo e recursivo.

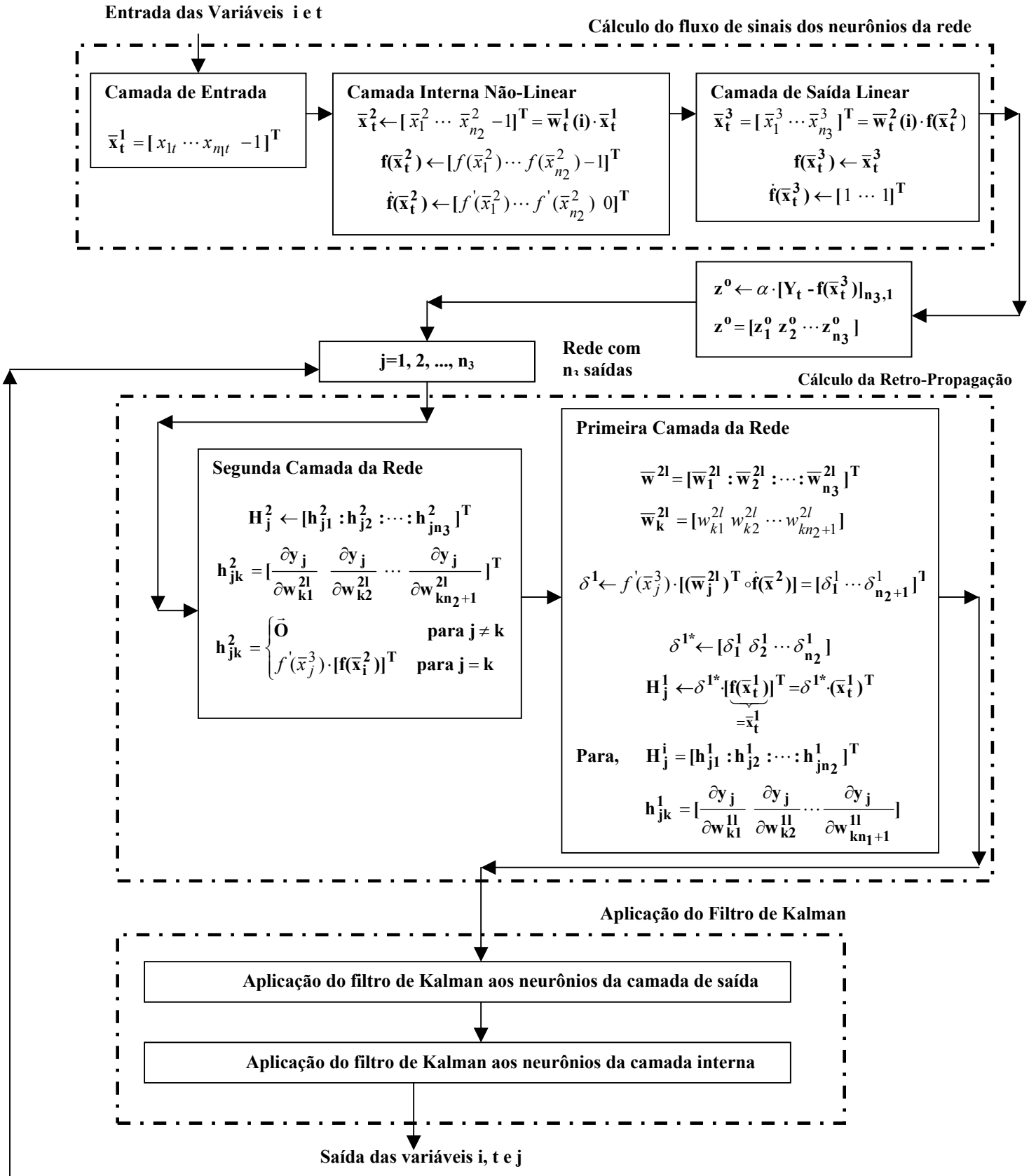


FIGURA B.2 – Continuação do fluxograma da Figura B.1.

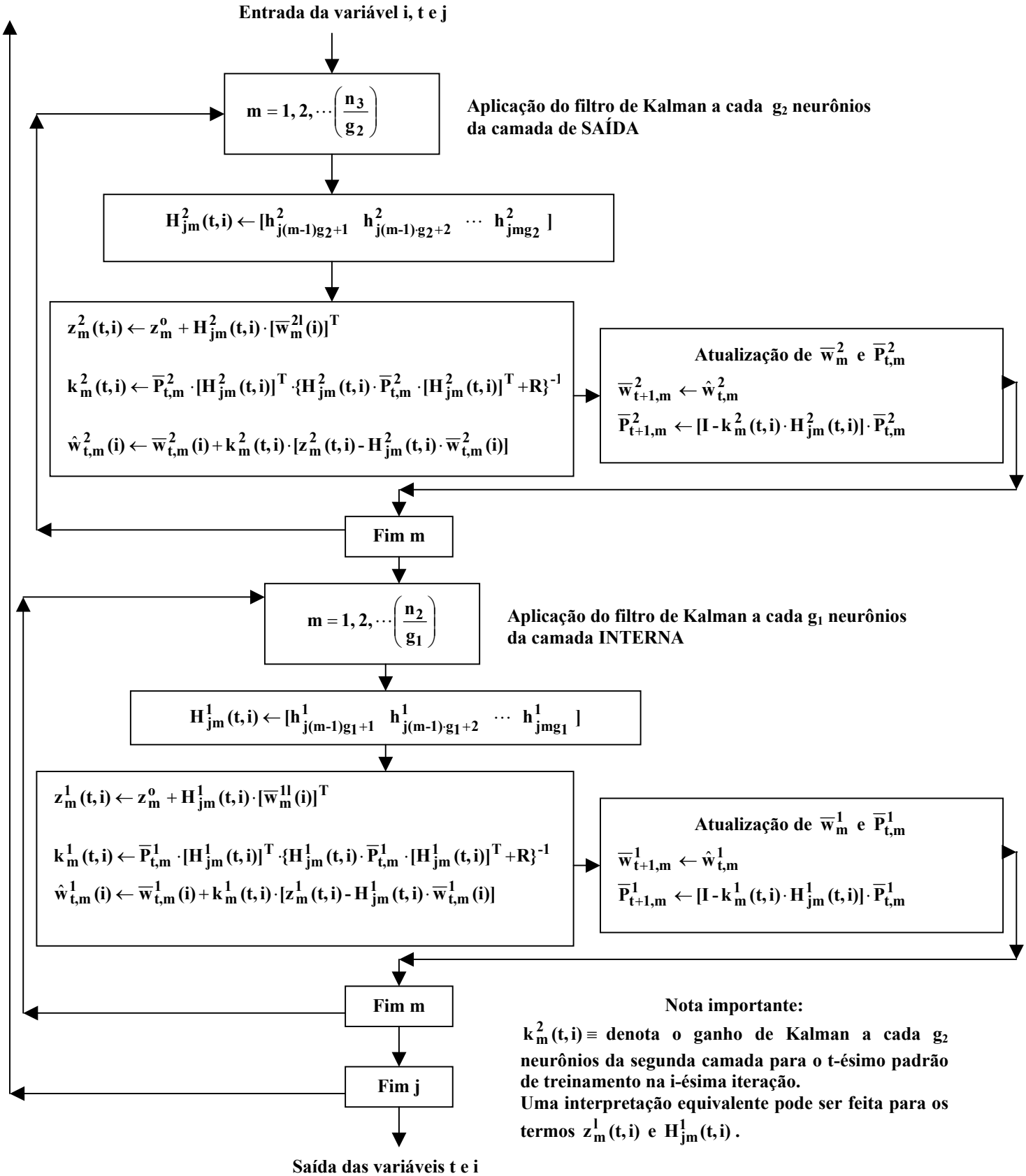


FIGURA B.3 – Continuação do fluxograma da Figura B.2.

APÊNDICE C

A RETRO-PROPAGAÇÃO EM RELAÇÃO ÀS ENTRADAS DOS NEURÔNIOS

Neste apêndice serão desenvolvidas as expressões analíticas da Retro-propagação das saídas da rede *feedforward* com relação às entradas dos neurônios das camadas anteriores. Como já mencionado anteriormente, estas expressões são essenciais para o cálculo das derivadas parciais $\partial\hat{\mathbf{y}}(\mathbf{t_j})/\partial\mathbf{u}(\mathbf{t_k})$ presentes na equação (10) do Capítulo 4. Na verdade, o desenvolvimento analítico a ser apresentado aqui, é um equacionamento mais detalhado daquilo que pode ser encontrado diretamente em Carrara (1997). Na Figura C.1 está representada a arquitetura de uma rede *feedforward,* e o que se deseja determinar nela são as derivadas parciais do vetor $\mathbf{y^k}$ em relação as variáveis vetoriais $\bar{\mathbf{y}}^{\mathbf{k-1}}$ e $\bar{\mathbf{y}}^{\mathbf{k-2}}$.

A representação analítica, como vista no Capítulo 3, da rede *feedforward* que está esquematizada na Figura C.1 é dada por:

$$\mathbf{y_b^k} = \mathbf{f^k}(\mathbf{W_b^k} \cdot \mathbf{y_b^{k-1}}) = \mathbf{f^k}(\bar{\mathbf{y}}_\mathbf{b}^\mathbf{k}) = [\,\mathbf{f^k}(\bar{\mathbf{y}}_\mathbf{1}^\mathbf{k})\;\; \mathbf{f^k}(\bar{\mathbf{y}}_\mathbf{2}^\mathbf{k})\;\; \mathbf{...}\;\; \mathbf{f^k}(\bar{\mathbf{y}}_{\mathbf{n_k}}^\mathbf{k})\;\; \mathbf{-1}\,]^\mathbf{T} \qquad (1.a)$$

onde,

$$\mathbf{W_b^k} = \begin{bmatrix} \mathbf{w_{11}^k} & \mathbf{w_{12}^k} & \cdots & \mathbf{w_{1n_{k-1}}^k} & \mathbf{b_1^k} \\ \mathbf{w_{21}^k} & \mathbf{w_{22}^k} & \cdots & \mathbf{w_{2n_{k-1}}^k} & \mathbf{b_2^k} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{w_{n_k 1}^k} & \mathbf{w_{n_k 2}^k} & \cdots & \mathbf{w_{n_k n_{k-1}}^k} & \mathbf{b_{n_k}^k} \end{bmatrix} \qquad (1.b)$$

Para simplificar o cálculo da Retro-propagação observe que as derivadas das saídas da rede com relação aos neurônios de *viés ou bias* são nulas, como ilustra o cálculo a seguir:

$$\frac{\partial y_1^k}{\partial \bar{y}_{n_{k-1}+1}^{k-1}} = \frac{\partial y_1^k}{\partial \bar{y}_1^k} \cdot \frac{\partial \bar{y}_1^k}{\partial \bar{y}_{n_{k-1}+1}^{k-1}} = \frac{\partial}{\partial \bar{y}_1^k} \cdot \underbrace{\frac{\partial}{\partial \bar{y}_{n_{k-1}+1}^{k-1}} \cdot \left(\sum_{i=1}^{n_{k-1}} w_{1i} \cdot y_i^{k-1} - b_1^k \right)}_{=0,\ \text{pois } y_{n_{k-1}+1}^{k-1} \text{ é igual a -1.}} = 0 \qquad (2)$$

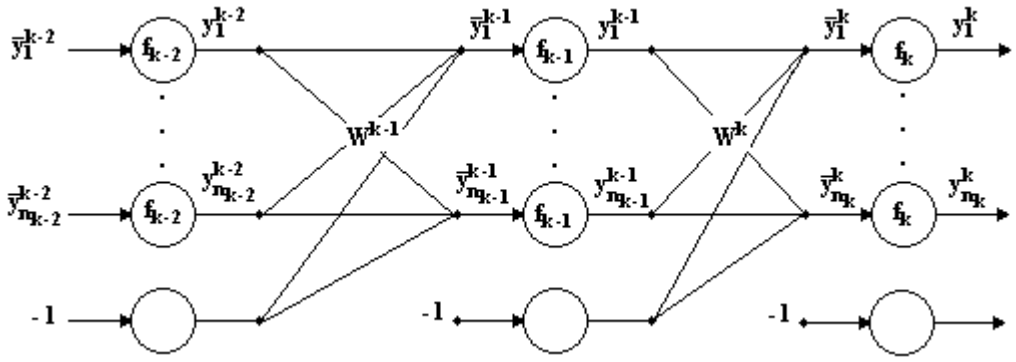


FIGURA C.1 – A Retro-propagação da saída vetorial y^k da rede neural com relação *às entradas dos neurônios* das camadas anteriores.

Deste modo, as expressões da Retro-propagação serão desenvolvidas sem a presença dos neurônios de *viés*. Com base nas equações anteriores e na Figura C.1 o cálculo de $\partial y^k / \partial \bar{y}^k$ é imediato, resultando que:

$$\frac{\partial y^k}{\partial \bar{y}^k} = \begin{bmatrix} \frac{\partial y_1^k}{\partial \bar{y}_1^k} & \frac{\partial y_1^k}{\partial \bar{y}_2^k} & \cdots & \frac{\partial y_1^k}{\partial \bar{y}_{n_k}^k} \\ \frac{\partial y_2^k}{\partial \bar{y}_1^k} & \frac{\partial y_2^k}{\partial \bar{y}_2^k} & \cdots & \frac{\partial y_2^k}{\partial \bar{y}_{n_k}^k} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_{n_k}^k}{\partial \bar{y}_1^k} & \frac{\partial y_{n_k}^k}{\partial \bar{y}_2^k} & \cdots & \frac{\partial y_{n_k}^k}{\partial \bar{y}_{n_k}^k} \end{bmatrix} = \begin{bmatrix} f'(\bar{y}_1^k) & 0 & 0 & 0 \\ 0 & f'(\bar{y}_2^k) & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & f'(\bar{y}_{n_k}^k) \end{bmatrix} = I_{f'(\bar{y}^k)} \qquad (3)$$

O cálculo de $\frac{\partial \mathbf{y}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}^{\mathbf{k-1}}}$ deve ser obtido de uma forma mais elaborada, como demonstrado a seguir:

$$\frac{\partial \mathbf{y}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}^{\mathbf{k-1}}} = \begin{bmatrix} \frac{\partial \mathbf{y}_{\mathbf{1}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{1}}^{\mathbf{k-1}}} & \frac{\partial \mathbf{y}_{\mathbf{1}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{2}}^{\mathbf{k-1}}} & \cdots & \frac{\partial \mathbf{y}_{\mathbf{1}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{n_{k-1}}}^{\mathbf{k-1}}} \\ \frac{\partial \mathbf{y}_{\mathbf{2}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{1}}^{\mathbf{k-1}}} & \frac{\partial \mathbf{y}_{\mathbf{2}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{2}}^{\mathbf{k-1}}} & \cdots & \frac{\partial \mathbf{y}_{\mathbf{2}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{n_{k-1}}}^{\mathbf{k-1}}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{y}_{\mathbf{n_k}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{1}}^{\mathbf{k-1}}} & \frac{\partial \mathbf{y}_{\mathbf{n_k}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{2}}^{\mathbf{k-1}}} & \cdots & \frac{\partial \mathbf{y}_{\mathbf{n_k}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{n_{k-1}}}^{\mathbf{k-1}}} \end{bmatrix} \tag{4}$$

Pela regra da cadeia, vem:

$$\frac{\partial \mathbf{y}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}^{\mathbf{k-1}}} = \begin{bmatrix} \frac{\partial \mathbf{y}_{\mathbf{1}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{1}}^{\mathbf{k}}} \cdot \frac{\partial \bar{\mathbf{y}}_{\mathbf{1}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{1}}^{\mathbf{k-1}}} & \frac{\partial \mathbf{y}_{\mathbf{1}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{1}}^{\mathbf{k}}} \cdot \frac{\partial \bar{\mathbf{y}}_{\mathbf{1}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{2}}^{\mathbf{k-1}}} & \cdots & \frac{\partial \mathbf{y}_{\mathbf{1}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{1}}^{\mathbf{k}}} \cdot \frac{\partial \bar{\mathbf{y}}_{\mathbf{1}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{n_{k-1}}}^{\mathbf{k-1}}} \\ \frac{\partial \mathbf{y}_{\mathbf{2}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{2}}^{\mathbf{k}}} \cdot \frac{\partial \bar{\mathbf{y}}_{\mathbf{2}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{1}}^{\mathbf{k-1}}} & \frac{\partial \mathbf{y}_{\mathbf{2}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{2}}^{\mathbf{k}}} \cdot \frac{\partial \bar{\mathbf{y}}_{\mathbf{2}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{2}}^{\mathbf{k-1}}} & \cdots & \frac{\partial \mathbf{y}_{\mathbf{2}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{2}}^{\mathbf{k}}} \cdot \frac{\partial \bar{\mathbf{y}}_{\mathbf{2}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{n_{k-1}}}^{\mathbf{k-1}}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{y}_{\mathbf{n_k}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{n_k}}^{\mathbf{k}}} \cdot \frac{\partial \bar{\mathbf{y}}_{\mathbf{n_k}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{1}}^{\mathbf{k-1}}} & \frac{\partial \mathbf{y}_{\mathbf{n_k}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{n_k}}^{\mathbf{k}}} \cdot \frac{\partial \bar{\mathbf{y}}_{\mathbf{n_k}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{2}}^{\mathbf{k-1}}} & \cdots & \frac{\partial \mathbf{y}_{\mathbf{n_k}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{n_k}}^{\mathbf{k}}} \cdot \frac{\partial \bar{\mathbf{y}}_{\mathbf{n_k}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{n_{k-1}}}^{\mathbf{k-1}}} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial \mathbf{y}_{\mathbf{1}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{1}}^{\mathbf{k}}} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \frac{\partial \mathbf{y}_{\mathbf{2}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{2}}^{\mathbf{k}}} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \frac{\partial \mathbf{y}_{\mathbf{n_k}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{n_k}}^{\mathbf{k}}} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial \bar{\mathbf{y}}_{\mathbf{1}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{1}}^{\mathbf{k-1}}} & \frac{\partial \bar{\mathbf{y}}_{\mathbf{1}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{2}}^{\mathbf{k-1}}} & \cdots & \frac{\partial \bar{\mathbf{y}}_{\mathbf{1}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{n_{k-1}}}^{\mathbf{k-1}}} \\ \frac{\partial \bar{\mathbf{y}}_{\mathbf{2}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{1}}^{\mathbf{k-1}}} & \frac{\partial \bar{\mathbf{y}}_{\mathbf{2}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{2}}^{\mathbf{k-1}}} & \cdots & \frac{\partial \bar{\mathbf{y}}_{\mathbf{2}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{n_{k-1}}}^{\mathbf{k-1}}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \bar{\mathbf{y}}_{\mathbf{n_k}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{1}}^{\mathbf{k-1}}} & \frac{\partial \bar{\mathbf{y}}_{\mathbf{n_k}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{2}}^{\mathbf{k-1}}} & \cdots & \frac{\partial \bar{\mathbf{y}}_{\mathbf{n_k}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{n_{k-1}}}^{\mathbf{k-1}}} \end{bmatrix} \tag{5}$$

Logo, tem-se:

$$\frac{\partial \mathbf{y}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}^{\mathbf{k-1}}} = \mathbf{I}_{\mathbf{f}'(\bar{\mathbf{y}}^{\mathbf{k}})} \cdot \begin{bmatrix} \mathbf{w}_{11}^{\mathbf{k}} \cdot \mathbf{f}'(\bar{\mathbf{y}}_1^{\mathbf{k-1}}) & \mathbf{w}_{12}^{\mathbf{k}} \cdot \mathbf{f}'(\bar{\mathbf{y}}_2^{\mathbf{k-1}}) & \cdots & \mathbf{w}_{1\mathbf{n}_{\mathbf{k-1}}}^{\mathbf{k}} \cdot \mathbf{f}'(\bar{\mathbf{y}}_{\mathbf{n}_{\mathbf{k-1}}}^{\mathbf{k-1}}) \\ \mathbf{w}_{21}^{\mathbf{k}} \cdot \mathbf{f}'(\bar{\mathbf{y}}_1^{\mathbf{k-1}}) & \mathbf{w}_{22}^{\mathbf{k}} \cdot \mathbf{f}'(\bar{\mathbf{y}}_2^{\mathbf{k-1}}) & \cdots & \mathbf{w}_{2\mathbf{n}_{\mathbf{k-1}}}^{\mathbf{k}} \cdot \mathbf{f}'(\bar{\mathbf{y}}_{\mathbf{n}_{\mathbf{k-1}}}^{\mathbf{k-1}}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{w}_{\mathbf{n}_{\mathbf{k}}1}^{\mathbf{k}} \cdot \mathbf{f}'(\bar{\mathbf{y}}_1^{\mathbf{k-1}}) & \mathbf{w}_{\mathbf{n}_{\mathbf{k}}2}^{\mathbf{k}} \cdot \mathbf{f}'(\bar{\mathbf{y}}_2^{\mathbf{k-1}}) & \cdots & \mathbf{w}_{\mathbf{n}_{\mathbf{k}}\mathbf{n}_{\mathbf{k-1}}}^{\mathbf{k}} \cdot \mathbf{f}'(\bar{\mathbf{y}}_{\mathbf{n}_{\mathbf{k-1}}}^{\mathbf{k-1}}) \end{bmatrix}$$

$$= \mathbf{I}_{\mathbf{f}'(\bar{\mathbf{y}}^{\mathbf{k}})} \cdot \underbrace{\begin{bmatrix} \mathbf{w}_{11}^{\mathbf{k}} & \mathbf{w}_{12}^{\mathbf{k}} & \cdots & \mathbf{w}_{1\mathbf{n}_{\mathbf{k-1}}}^{\mathbf{k}} \\ \mathbf{w}_{21}^{\mathbf{k}} & \mathbf{w}_{22}^{\mathbf{k}} & \cdots & \mathbf{w}_{2\mathbf{n}_{\mathbf{k-1}}}^{\mathbf{k}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{w}_{\mathbf{n}_{\mathbf{k}}1}^{\mathbf{k}} & \mathbf{w}_{\mathbf{n}_{\mathbf{k}}2}^{\mathbf{k}} & \cdots & \mathbf{w}_{\mathbf{n}_{\mathbf{k}}\mathbf{n}_{\mathbf{k-1}}}^{\mathbf{k}} \end{bmatrix}}_{\mathbf{W}^{\mathbf{k}}} \underbrace{\begin{bmatrix} \mathbf{f}'(\bar{\mathbf{y}}_1^{\mathbf{k-1}}) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{f}'(\bar{\mathbf{y}}_2^{\mathbf{k-1}}) & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{f}'(\bar{\mathbf{y}}_{\mathbf{n}_{\mathbf{k}}}^{\mathbf{k-1}}) \end{bmatrix}}_{\mathbf{I}_{\mathbf{f}'(\bar{\mathbf{y}}^{\mathbf{k-1}})}} \tag{6}$$

Assim,

$$\frac{\partial \mathbf{y}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}^{\mathbf{k-1}}} = \mathbf{I}_{\mathbf{f}'(\bar{\mathbf{y}}^{\mathbf{k}})} \cdot \mathbf{W}^{\mathbf{k}} \cdot \mathbf{I}_{\mathbf{f}'(\bar{\mathbf{y}}^{\mathbf{k-1}})} \tag{7}$$

Substituindo a equação (3) na equação (7), resulta que:

$$\frac{\partial \mathbf{y}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}^{\mathbf{k-1}}} = \frac{\partial \mathbf{y}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}^{\mathbf{k}}} \cdot \mathbf{W}^{\mathbf{k}} \cdot \mathbf{I}_{\mathbf{f}'(\bar{\mathbf{y}}^{\mathbf{k-1}})} \tag{8}$$

A determinação da matriz de derivadas $\frac{\partial \mathbf{y}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}^{\mathbf{k-2}}}$ é efetuada da seguinte forma:

$$\frac{\partial \mathbf{y}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}^{\mathbf{k-2}}} = \begin{bmatrix} \frac{\partial \mathbf{y}_1^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_1^{\mathbf{k}}} \cdot \frac{\partial \bar{\mathbf{y}}_1^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_1^{\mathbf{k-2}}} & \frac{\partial \mathbf{y}_1^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_1^{\mathbf{k}}} \cdot \frac{\partial \bar{\mathbf{y}}_1^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_2^{\mathbf{k-2}}} & \cdots & \frac{\partial \mathbf{y}_1^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_1^{\mathbf{k}}} \cdot \frac{\partial \bar{\mathbf{y}}_1^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{n_{k-2}}}^{\mathbf{k-2}}} \\ \frac{\partial \mathbf{y}_2^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_2^{\mathbf{k}}} \cdot \frac{\partial \bar{\mathbf{y}}_2^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_1^{\mathbf{k-2}}} & \frac{\partial \mathbf{y}_2^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_2^{\mathbf{k}}} \cdot \frac{\partial \bar{\mathbf{y}}_2^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_2^{\mathbf{k-2}}} & \cdots & \frac{\partial \mathbf{y}_2^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_2^{\mathbf{k}}} \cdot \frac{\partial \bar{\mathbf{y}}_2^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{n_{k-2}}}^{\mathbf{k-2}}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{y}_{\mathbf{n_k}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{n_k}}^{\mathbf{k}}} \cdot \frac{\partial \bar{\mathbf{y}}_{\mathbf{n_k}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_1^{\mathbf{k-2}}} & \frac{\partial \mathbf{y}_{\mathbf{n_k}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{n_k}}^{\mathbf{k}}} \cdot \frac{\partial \bar{\mathbf{y}}_{\mathbf{n_k}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_2^{\mathbf{k-2}}} & \cdots & \frac{\partial \mathbf{y}_{\mathbf{n_k}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{n_k}}^{\mathbf{k}}} \cdot \frac{\partial \bar{\mathbf{y}}_{\mathbf{n_k}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{n_{k-2}}}^{\mathbf{k-2}}} \end{bmatrix} \tag{9}$$

A equação (9) pode ser expressa pelo seguinte produto matricial:

$$\frac{\partial \mathbf{y}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}^{\mathbf{k-2}}} = \mathbf{I}_{\mathbf{f}'(\bar{\mathbf{y}}^{\mathbf{k}})} \cdot \underbrace{\begin{bmatrix} \frac{\partial \bar{\mathbf{y}}_1^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_1^{\mathbf{k-2}}} & \frac{\partial \bar{\mathbf{y}}_1^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_2^{\mathbf{k-2}}} & \cdots & \frac{\partial \bar{\mathbf{y}}_1^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{n_{k-2}}}^{\mathbf{k-2}}} \\ \frac{\partial \bar{\mathbf{y}}_2^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_1^{\mathbf{k-2}}} & \frac{\partial \bar{\mathbf{y}}_2^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_2^{\mathbf{k-2}}} & \cdots & \frac{\partial \bar{\mathbf{y}}_2^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{n_{k-2}}}^{\mathbf{k-2}}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \bar{\mathbf{y}}_{\mathbf{n_k}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_1^{\mathbf{k-2}}} & \frac{\partial \bar{\mathbf{y}}_{\mathbf{n_k}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_2^{\mathbf{k-2}}} & \cdots & \frac{\partial \bar{\mathbf{y}}_{\mathbf{n_k}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{n_{k-2}}}^{\mathbf{k-2}}} \end{bmatrix}}_{\mathbf{J_1}} \tag{10}$$

Aplicando novamente a regra da cadeia sobre a matriz $\mathbf{J_1}$, tem-se:

$$\mathbf{J_1} = \begin{bmatrix} \sum_{\mathbf{i}=1}^{\mathbf{n_{k-1}}} \frac{\partial \bar{\mathbf{y}}_1^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{i}}^{\mathbf{k-1}}} \cdot \frac{\partial \bar{\mathbf{y}}_{\mathbf{i}}^{\mathbf{k-1}}}{\partial \bar{\mathbf{y}}_1^{\mathbf{k-2}}} & \sum_{\mathbf{i}=1}^{\mathbf{n_{k-1}}} \frac{\partial \bar{\mathbf{y}}_1^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{i}}^{\mathbf{k-1}}} \cdot \frac{\partial \bar{\mathbf{y}}_{\mathbf{i}}^{\mathbf{k-1}}}{\partial \bar{\mathbf{y}}_2^{\mathbf{k-2}}} & \cdots & \sum_{\mathbf{i}=1}^{\mathbf{n_{k-1}}} \frac{\partial \bar{\mathbf{y}}_1^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{i}}^{\mathbf{k-1}}} \cdot \frac{\partial \bar{\mathbf{y}}_{\mathbf{i}}^{\mathbf{k-1}}}{\partial \bar{\mathbf{y}}_{\mathbf{n_{k-2}}}^{\mathbf{k-2}}} \\ \sum_{\mathbf{i}=1}^{\mathbf{n_{k-1}}} \frac{\partial \bar{\mathbf{y}}_2^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{i}}^{\mathbf{k-1}}} \cdot \frac{\partial \bar{\mathbf{y}}_{\mathbf{i}}^{\mathbf{k-1}}}{\partial \bar{\mathbf{y}}_1^{\mathbf{k-2}}} & \sum_{\mathbf{i}=1}^{\mathbf{n_{k-1}}} \frac{\partial \bar{\mathbf{y}}_2^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{i}}^{\mathbf{k-1}}} \cdot \frac{\partial \bar{\mathbf{y}}_{\mathbf{i}}^{\mathbf{k-1}}}{\partial \bar{\mathbf{y}}_2^{\mathbf{k-2}}} & \cdots & \sum_{\mathbf{i}=1}^{\mathbf{n_{k-1}}} \frac{\partial \bar{\mathbf{y}}_2^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{i}}^{\mathbf{k-1}}} \cdot \frac{\partial \bar{\mathbf{y}}_{\mathbf{i}}^{\mathbf{k-1}}}{\partial \bar{\mathbf{y}}_{\mathbf{n_{k-2}}}^{\mathbf{k-2}}} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{\mathbf{i}=1}^{\mathbf{n_{k-1}}} \frac{\partial \bar{\mathbf{y}}_{\mathbf{n_k}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{i}}^{\mathbf{k-1}}} \cdot \frac{\partial \bar{\mathbf{y}}_{\mathbf{i}}^{\mathbf{k-1}}}{\partial \bar{\mathbf{y}}_1^{\mathbf{k-2}}} & \sum_{\mathbf{i}=1}^{\mathbf{n_{k-1}}} \frac{\partial \bar{\mathbf{y}}_{\mathbf{n_k}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{i}}^{\mathbf{k-1}}} \cdot \frac{\partial \bar{\mathbf{y}}_{\mathbf{i}}^{\mathbf{k-1}}}{\partial \bar{\mathbf{y}}_2^{\mathbf{k-2}}} & \cdots & \sum_{\mathbf{i}=1}^{\mathbf{n_{k-1}}} \frac{\partial \bar{\mathbf{y}}_{\mathbf{n_k}}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}_{\mathbf{i}}^{\mathbf{k-1}}} \cdot \frac{\partial \bar{\mathbf{y}}_{\mathbf{i}}^{\mathbf{k-1}}}{\partial \bar{\mathbf{y}}_{\mathbf{n_{k-2}}}^{\mathbf{k-2}}} \end{bmatrix} \tag{11}$$

A matriz $\mathbf{J_1}$ ainda pode ser expressa pelo seguinte produto matricial:

$$\mathbf{J_1}=\underbrace{\begin{bmatrix} \frac{\partial \bar{y}_1^k}{\partial \bar{y}_1^{k-1}} & \frac{\partial \bar{y}_1^k}{\partial \bar{y}_2^{k-1}} & \cdots & \frac{\partial \bar{y}_1^k}{\partial \bar{y}_{n_{k-1}}^{k-1}} \\ \frac{\partial \bar{y}_2^k}{\partial \bar{y}_1^{k-1}} & \frac{\partial \bar{y}_2^k}{\partial \bar{y}_2^{k-1}} & \cdots & \frac{\partial \bar{y}_2^k}{\partial \bar{y}_{n_{k-1}}^{k-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \bar{y}_{n_k}^k}{\partial \bar{y}_1^{k-1}} & \frac{\partial \bar{y}_{n_k}^k}{\partial \bar{y}_2^{k-1}} & \cdots & \frac{\partial \bar{y}_{n_k}^k}{\partial \bar{y}_{n_{k-1}}^{k-1}} \end{bmatrix}}_{\mathbf{J_2}} \cdot \underbrace{\begin{bmatrix} \frac{\partial \bar{y}_1^{k-1}}{\partial \bar{y}_1^{k-2}} & \frac{\partial \bar{y}_1^{k-1}}{\partial \bar{y}_2^{k-2}} & \cdots & \frac{\partial \bar{y}_1^{k-1}}{\partial \bar{y}_{n_{k-2}}^{k-2}} \\ \frac{\partial \bar{y}_2^{k-1}}{\partial \bar{y}_1^{k-2}} & \frac{\partial \bar{y}_2^{k-1}}{\partial \bar{y}_2^{k-2}} & \cdots & \frac{\partial \bar{y}_2^{k-1}}{\partial \bar{y}_{n_{k-2}}^{k-2}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \bar{y}_{n_{k-1}}^{k-1}}{\partial \bar{y}_1^{k-2}} & \frac{\partial \bar{y}_{n_{k-1}}^{k-1}}{\partial \bar{y}_2^{k-2}} & \cdots & \frac{\partial \bar{y}_{n_{k-1}}^{k-1}}{\partial \bar{y}_{n_{k-2}}^{k-2}} \end{bmatrix}}_{\mathbf{J_3}} \tag{12}$$

Substituindo as equações (5) e (12) na equação (10), vem:

$$\frac{\partial \mathbf{y}^k}{\partial \bar{\mathbf{y}}^{k-2}} = \mathbf{I}_{\mathbf{f}'(\bar{\mathbf{y}}^k)} \cdot \mathbf{J_1} = \underbrace{\mathbf{I}_{\mathbf{f}'(\bar{\mathbf{y}}^k)} \cdot \mathbf{J_2}}_{\partial \mathbf{y}^k / \partial \mathbf{y}^{k-1}} \cdot \mathbf{J_3} = \frac{\partial \mathbf{y}^k}{\partial \bar{\mathbf{y}}^{k-1}} \cdot \mathbf{J_3} \tag{13}$$

Calculando as derivadas da matriz $\mathbf{J_3}$ com base na Figura C.1, tem-se:

$$\mathbf{J_3}=\begin{bmatrix} w_{11}^{k-1} \cdot f'(\bar{y}_1^{k-2}) & w_{12}^{k-1} \cdot f'(\bar{y}_2^{k-2}) & \cdots & w_{1n_{k-2}}^{k-1} \cdot f'(\bar{y}_{n_{k-2}}^{k-2}) \\ w_{21}^{k-1} \cdot f'(\bar{y}_1^{k-2}) & w_{22}^{k-1} \cdot f'(\bar{y}_2^{k-2}) & \cdots & w_{2n_{k-2}}^{k-1} \cdot f'(\bar{y}_{n_{k-2}}^{k-2}) \\ \vdots & \vdots & \ddots & \vdots \\ w_{n_{k-1}1}^{k-1} \cdot f'(\bar{y}_1^{k-2}) & w_{n_{k-1}2}^{k-1} \cdot f'(\bar{y}_2^{k-2}) & \cdots & w_{n_{k-1}n_{k-2}}^{k-1} \cdot f'(\bar{y}_{n_{k-2}}^{k-2}) \end{bmatrix} \tag{14}$$

A expressão (14) finalmente pode ser expressa no seguinte produto matricial:

$$\mathbf{J_3}=\underbrace{\begin{bmatrix} w_{11}^{k-1} & w_{12}^{k-1} & \cdots & w_{1n_{k-2}}^{k-1} \\ w_{21}^{k-1} & w_{22}^{k-1} & \cdots & w_{2n_{k-2}}^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n_{k-1}1}^{k-1} & w_{n_{k-1}2}^{k-1} & \cdots & w_{n_{k-1}n_{k-2}}^{k-1} \end{bmatrix}}_{\mathbf{W}^{k-1}} \cdot \underbrace{\begin{bmatrix} f'(\bar{y}_1^{k-2}) & 0 & \cdots & 0 \\ 0 & f'(\bar{y}_2^{k-2}) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f'(\bar{y}_{n_{k-2}}^{k-2}) \end{bmatrix}}_{\mathbf{I}_{\mathbf{f}'(\bar{\mathbf{y}}^{k-2})}}$$

(15)

Substituindo a equação (15) na equação (13), resulta que:

$$\frac{\partial \mathbf{y}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}^{\mathbf{k-2}}} = \frac{\partial \mathbf{y}^{\mathbf{k}}}{\partial \bar{\mathbf{y}}^{\mathbf{k-1}}} \cdot \mathbf{W}^{\mathbf{k-1}} \cdot \mathbf{I}_{\mathbf{f}'(\bar{\mathbf{y}}^{\mathbf{k-2}})} \tag{16}$$

Analisando as equações (3), (8) e (16) pode-se formular o seguinte conjunto de equações *iterativas* para determinar as derivadas das saídas da rede em relação às entradas dos neurônios das camadas anteriores:

$$\frac{\partial \mathbf{y}^{\mathbf{l}}}{\partial \bar{\mathbf{y}}^{\mathbf{k}}} = \frac{\partial \mathbf{y}^{\mathbf{l}}}{\partial \bar{\mathbf{y}}^{\mathbf{k+1}}} \cdot \mathbf{W}^{\mathbf{k+1}} \cdot \mathbf{I}_{\mathbf{f}'(\bar{\mathbf{y}}^{\mathbf{k}})} \qquad \text{para } \mathbf{k=l-1, l-2} \tag{17.a}$$

onde,

$$\frac{\partial \mathbf{y}^{\mathbf{l}}}{\partial \bar{\mathbf{y}}^{\mathbf{l}}} = \mathbf{I}_{\mathbf{f}'(\bar{\mathbf{y}}^{\mathbf{l}})} \tag{17.b}$$

$$\mathbf{I}_{\mathbf{f}'(\bar{\mathbf{y}}^{\mathbf{l}})} = \begin{bmatrix} \mathbf{f}'(\bar{\mathbf{y}}^{\mathbf{l}}_{\mathbf{1}}) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{f}'(\bar{\mathbf{y}}^{\mathbf{l}}_{\mathbf{2}}) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{f}'(\bar{\mathbf{y}}^{\mathbf{l}}_{\mathbf{n_k}}) \end{bmatrix} \tag{17.c}$$

Ainda com relação às equações (17.a), (17.b) e (17.c) pode-se demonstrar que elas ainda são válidas para redes com mais de três camadas. Deste modo, é possível estender estas equações para o caso onde **k=l-1, l-2, ..., 1** para **l** (número total de camadas da rede) um inteiro qualquer maior ou igual a 3. Observe que $\mathbf{W}^{\mathbf{l}}$ é a matriz de pesos da camada **l**, $\mathbf{W}^{\mathbf{l-1}}$ é a matriz de pesos da camada **l-1**, ..., $\mathbf{W}^{\mathbf{2}}$ é a matriz de pesos da camada **2**. Não há $\mathbf{W}^{\mathbf{1}}$ pois, a camada de entrada da rede não recebe pesos de conexões. É interessante perceber também que às equações de (17.a) à (17.c) são bem semelhantes às equações de (10.a) à (10.c) do Capítulo 3 referentes a Retro-propagação *dos pesos* da rede. Na verdade elas são bem parecidas, mas não devem ser confundidas, pois as primeiras destas equações são utilizadas numa estrutura de controle preditivo para obter uma política de controle através da resolução de um problema de otimização não-linear sobre

a rede *já treinada* e as últimas destas equações são utilizadas no treinamento, propriamente dito, da rede neural que representará o sistema dinâmico.

APÊNDICE D

INTEGRADORES NUMÉRICOS

Neste apêndice se desenvolverão alguns algoritmos computacionais utilizados para resolver *numericamente* equações diferenciais ordinárias ou sistema de equações diferenciais. Inicialmente são definidos (Ames, 1977) vários operadores de diferença que são aplicados sobre uma função **y** de uma variável independente **y=f(x)** sobre um intervalo de dimensão constante $\mathbf{h = x_{i+1} - x_i}$ sendo $\mathbf{y_i = f(x_i)}$.

a) Diferença para frente: $\Delta \mathbf{y_i} = \mathbf{y_{i+1} - y_i}$

b) Diferença para trás: $\nabla \mathbf{y_i} = \mathbf{y_i - y_{i-1}}$

c) Diferença central: $\delta \mathbf{y_i} = \mathbf{y_{i+1/2} - y_{i-1/2}}$

d) Operador média: $\mu \mathbf{y_i} = \dfrac{\mathbf{1}}{\mathbf{2}} \cdot [\mathbf{y_{i+1/2} - y_{i-1/2}}]$

e) Operador troca: $\mathbf{E y_i} = \mathbf{y_{i+1}}$

f) Operador integral: $\mathbf{Jy} = \int_{\mathbf{x}}^{\mathbf{x+h}} \mathbf{y(t)} \cdot \mathbf{dt}$

g) Operador diferencial: $\mathbf{Dy} = \mathbf{dy/dx}$

É bem conhecido que o operador diferencial $\mathbf{D \equiv d/dx}$ e os operadores diferenças Δ, ∇ **e** δ podem ser *utilizados simbolicamente como se eles fossem números*, uma vez que eles satisfazem formalmente as leis da álgebra (Salvadori e Baron, 1961). Por exemplo, para o operador diferença para trás são válidas as seguintes propriedades algébricas:

a) $\nabla^{\mathbf{n}} \mathbf{y_i} = \nabla\left(\nabla^{\mathbf{n-1}} \mathbf{y_i}\right)$

b) $\nabla\left(\mathbf{y_i} + \mathbf{y_j}\right) = \nabla \mathbf{y_i} + \nabla \mathbf{y_j} = \nabla \mathbf{y_j} + \nabla \mathbf{y_i}$

c) $\nabla(\mathbf{c} \cdot \mathbf{y_i}) = \mathbf{c} \cdot \nabla(\mathbf{y_i})$

d) $\nabla^{\mathbf{m}}\left(\nabla^{\mathbf{n}} \mathbf{y_i}\right) = \nabla_{\mathbf{y_i}}^{\mathbf{m+n}}$

e) $\nabla^{\mathbf{n}} \mathbf{y_i} = \nabla^{\mathbf{n-1}} \mathbf{y_i} - \nabla^{\mathbf{n-1}} \mathbf{y_{i-1}}$

Pode-se demonstrar facilmente estas propriedades e que também são válidas para os operadores Δ, δ **e** μ. A Tabela D.1 esquematiza uma maneira prática de se calcular as diferenças sucessivas de $\nabla^{\mathbf{n}} \mathbf{y_i}$.

TABELA D.1 – Determinação de sucessivas diferenças para trás.

I	$\mathbf{y_i}$	$\nabla \mathbf{y_i}$	$\nabla^2 \mathbf{y_i}$	$\nabla^3 \mathbf{y_i}$	$\nabla^4 \mathbf{y_i}$	$\nabla^5 \mathbf{y_i}$
0	$\mathbf{y_o}$					
1	$\mathbf{y_1}$	$\nabla \mathbf{y_1}$				
2	$\mathbf{y_2}$	$\nabla \mathbf{y_2}$	$\nabla^2 \mathbf{y_2}$			
3	$\mathbf{y_3}$	$\nabla \mathbf{y_3}$	$\nabla^2 \mathbf{y_3}$	$\nabla^3 \mathbf{y_3}$		
4	$\mathbf{y_4}$	$\nabla \mathbf{y_4}$	$\nabla^2 \mathbf{y_4}$	$\nabla^3 \mathbf{y_4}$	$\nabla^4 \mathbf{y_4}$	
5	$\mathbf{y_5}$	$\nabla \mathbf{y_5}$	$\nabla^2 \mathbf{y_5}$	$\nabla^3 \mathbf{y_5}$	$\nabla^4 \mathbf{y_5}$	$\nabla^5 \mathbf{y_5}$

Fazendo-se o uso dessas propriedades algébricas é possível expressar as diferenças de uma função y = f(x) em termos de suas sucessivas derivadas e, inversamente, suas derivadas em termos de suas sucessivas diferenças. Através de uma expansão em série de Taylor, tem-se:

$$\mathbf{y_{i+1}} = \mathbf{y} + \frac{\mathbf{h}}{\mathbf{1!}} \cdot \mathbf{D y_i} + \frac{\mathbf{h^2}}{\mathbf{2!}} \cdot \mathbf{D^2 y_i} + \frac{\mathbf{h^3}}{\mathbf{3!}} \cdot \mathbf{D^3 y_i} + \ldots \qquad (1.a)$$

ou

$$\mathbf{y_{i+1}} = \left(\mathbf{1} + \frac{\mathbf{h}}{\mathbf{1!}} \cdot \mathbf{D} + \frac{\mathbf{h^2}}{\mathbf{2!}} \cdot \mathbf{D^2} + \frac{\mathbf{h^3}}{\mathbf{3!}} \cdot \mathbf{D^3} + \ldots \right) \cdot \mathbf{y_i} \tag{1.b}$$

Como os operadores em questão podem ser tratados como números pode-se chegar as seguintes propriedades básicas:

$$\mathbf{y_{i+1}} = \mathbf{e^{h \cdot D}} \mathbf{y_i} \tag{2.a}$$

$$\mathbf{y_{i-1}} = \mathbf{e^{-h \cdot D}} \mathbf{y_i} \tag{2.b}$$

$$\nabla \mathbf{y_i} = \mathbf{y_i} - \mathbf{y_{i-1}} = \mathbf{[1 - e^{-h \cdot D}]} \cdot \mathbf{y_i} \tag{2.c}$$

Da expressão $\nabla = \mathbf{1 - e^{-h \cdot D}}$ pode-se concluir facilmente que:

$$\mathbf{- h \cdot D = ln(1 - \nabla)} \tag{3}$$

Com uma nova expansão em série de Taylor da expressão $\mathbf{ln(1 - \nabla)}$ chega-se finalmente ao seguinte resultado:

$$\mathbf{- h \cdot D} = -\left(\nabla + \frac{\nabla^2}{\mathbf{2}} + \frac{\nabla^3}{\mathbf{3}} + \frac{\nabla^4}{\mathbf{4}} + \ldots \right) \tag{4.a}$$

ou

$$\mathbf{D y_i} = \frac{1}{\mathbf{h}} \cdot \left[\nabla + \frac{\nabla^2}{\mathbf{2}} + \frac{\nabla^3}{\mathbf{3}} + \frac{\nabla^4}{\mathbf{4}} + \ldots \right] \cdot \mathbf{y_i} \tag{4.b}$$

De maneira análoga tem-se:

$$\mathbf{D y_i} = \frac{\mathbf{1}}{\mathbf{h}} \cdot \left[\Delta - \frac{\mathbf{1}}{\mathbf{2}} \cdot \Delta^2 + \frac{\mathbf{1}}{\mathbf{3}} \cdot \Delta^3 - \frac{\mathbf{1}}{\mathbf{4}} \cdot \Delta^4 + \ldots \right] \cdot \mathbf{y_i} \tag{5}$$

A derivada de ordem **k** pode ser dada pelas seguintes expressões (Ames, 1977):

$$\mathbf{D^k y_i} = \frac{\mathbf{1}}{\mathbf{h^k}} \cdot [\mathbf{ln(1+\Delta)}]^{\mathbf{k}} \cdot \mathbf{y_i}$$

$$= \frac{\mathbf{1}}{\mathbf{h^k}} \cdot \left[\Delta^{\mathbf{k}} - \frac{\mathbf{k}}{\mathbf{2}} \cdot \Delta^{\mathbf{k+1}} + \frac{\mathbf{k \cdot (3k+5)}}{\mathbf{24}} \cdot \Delta^{\mathbf{k+2}} - \frac{\mathbf{k \cdot (k+2) \cdot (k+3)}}{\mathbf{48}} \cdot \Delta^{\mathbf{k+3}} + \ldots \right] \cdot \mathbf{y_i} \qquad (6.a)$$

ou

$$\mathbf{D^k y_i} = -\frac{\mathbf{1}}{\mathbf{h^k}} \cdot [\mathbf{ln(1-\nabla)}]^{\mathbf{k}} \cdot \mathbf{y_i} = \frac{\mathbf{1}}{\mathbf{h^k}} \cdot \left(\nabla + \frac{\mathbf{1}}{\mathbf{2}} \cdot \nabla^{\mathbf{2}} + \frac{\mathbf{1}}{\mathbf{3}} \cdot \nabla^{\mathbf{3}} + \ldots \right)^{\mathbf{k}} \cdot \mathbf{y_i}$$

$$= \frac{\mathbf{1}}{\mathbf{h^k}} \cdot \left[\nabla^{\mathbf{k}} + \frac{\mathbf{k}}{\mathbf{2}} \cdot \nabla^{\mathbf{k+1}} + \frac{\mathbf{k \cdot (3k+5)}}{\mathbf{24}} \cdot \nabla^{\mathbf{k+2}} + \frac{\mathbf{k \cdot (k+2) \cdot (k+3)}}{\mathbf{48}} \cdot \nabla^{\mathbf{k+3}} + \ldots \right] \cdot \mathbf{y_i} \qquad (6.b)$$

As equações (6.a) e (6.b) expressam, respectivamente, as derivadas de ordem **k** em função dos operadores diferença finita para frente e diferença finita para trás. Desta forma, por exemplo, a segunda derivada de $\mathbf{y_i}$ $\left(\frac{\mathbf{d^2 y_i}}{\mathbf{dx_i^2}} \right)$ pode ser obtida numericamente pela equação (6.a) até o termo de segunda ordem como:

$$\frac{\mathbf{d^2 y_i}}{\mathbf{dx_i^2}} = \frac{\mathbf{1}}{\mathbf{h^2}} \cdot \left[\Delta^{\mathbf{2}} - \Delta^{\mathbf{3}} + \frac{\mathbf{11}}{\mathbf{12}} \cdot \Delta^{\mathbf{4}} - \frac{\mathbf{5}}{\mathbf{6}} \cdot \Delta^{\mathbf{5}} + \ldots \right] \cdot \mathbf{y_i} \qquad (7.a)$$

$$\approx \frac{\mathbf{1}}{\mathbf{h^2}} \cdot \left[\Delta^{\mathbf{2}} - \Delta^{\mathbf{3}} \right] \cdot \mathbf{y_i} + \mathbf{O(h^2)} \qquad (7.b)$$

$$\approx \frac{\mathbf{1}}{\mathbf{h^2}} \cdot \left(\mathbf{y_{i+1} - 2 \cdot y_i + y_{i-1}} \right) - \frac{\mathbf{1}}{\mathbf{h^2}} \cdot \left(\mathbf{y_{i+2} - 3 \cdot y_{i+1} + 3 \cdot y_i - y_{i-1}} \right) + \mathbf{O(h^2)} \qquad (7.c)$$

O operador diferença central δ também pode ser operado de maneira análoga a que foi realizada anteriormente. No entanto, é importante perceber que:

$$\delta \mathbf{y_i} = \mathbf{y}\left(\mathbf{x_i} + \frac{\mathbf{h}}{\mathbf{2}}\right) - \mathbf{y}\left(\mathbf{x_i} - \frac{\mathbf{h}}{\mathbf{2}}\right) = \mathbf{y_{i+1/2}} - \mathbf{y_{i-1/2}} \tag{8.a}$$

$$\mu \mathbf{y_i} = \frac{\mathbf{1}}{\mathbf{2}} \cdot \left[\mathbf{y_{i+1/2}} - \mathbf{y_{i-1/2}}\right] \tag{8.b}$$

$$\mu^{\mathbf{2}}\mathbf{y_i} = \left(\mathbf{1} + \frac{\delta^{\mathbf{2}}}{\mathbf{4}}\right) \cdot \mathbf{y_i} = \frac{\mathbf{1}}{\mathbf{4}} \cdot \left(\mathbf{y_{i+1}} + \mathbf{2} \cdot \mathbf{y_i} + \mathbf{y_{i-1}}\right) \tag{8.c}$$

$$\mu\delta \mathbf{y_i} = \mathbf{sinh(h.D)y_i} = \frac{\mathbf{1}}{\mathbf{2}} \cdot \left(\mathbf{y_{i+1}} - \mathbf{y_{i-1}}\right) \tag{8.d}$$

$$\mathbf{h.D} = \mathbf{sinh}^{\mathbf{-1}}\left(\mu\delta\right) \tag{8.e}$$

Molécula Computacional

Molécula computacional é uma maneira eficiente e prática de obter graficamente o desenvolvimento em série de Taylor dos operadores diferença finita e diferencial.

Por exemplo, o desenvolvimento das primeiras, segundas e terceiras derivadas de $\mathbf{y_i}$ com erros de primeira ordem obtidas em termos dos operadores diferença finita para trás são dados, respectivamente, por:

$$\mathbf{Dy_i} = \frac{\mathbf{1}}{\mathbf{h}} \cdot \left(\mathbf{y_i} - \mathbf{y_{i-1}}\right) + \mathbf{O(h)} \tag{9.a}$$

$$\mathbf{D^2 y_i} = \frac{\mathbf{1}}{\mathbf{h^2}} \cdot \left(\mathbf{y_i} - \mathbf{2} \cdot \mathbf{y_{i-1}} + \mathbf{y_{i-2}}\right) + \mathbf{O(h)} \tag{9.b}$$

$$\mathbf{D^3 y_i} = \frac{\mathbf{1}}{\mathbf{h^3}} \cdot \left(\mathbf{y_i} - \mathbf{3} \cdot \mathbf{y_{i-1}} + \mathbf{3} \cdot \mathbf{y_{i-2}} - \mathbf{y_{i-3}}\right) + \mathbf{O(h)} \tag{9.c}$$

A representação em molécula computacional das exptressões acima pode ser vista na Figura D.1. Esta representação gráfica facilita o entendimento das expresões analíticas e

consequentemente da implementação computacional dos respectivos algoritmos numéricos.

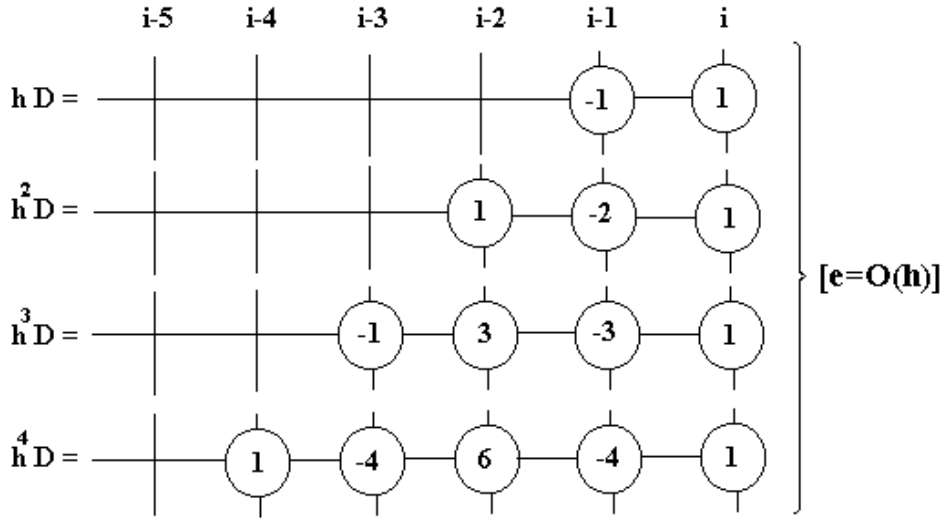


FIGURA D.1 – Representação gráfica das derivadas como molécula computacional.

Polinômio de Newton

Um polinômio de grau n que passa através de n-1 pontos $(x_o, y_o), (x_1, y_1), ..., (x_n, y_n)$ onde os valores de **x** são igualmente espaçados em intervalos de comprimento **h** pode ser escrito como (Pennington, 1970):

$$y = y_o + \frac{\Delta y_o}{h}.(x - x_o) + \frac{\Delta^2 y_o}{2h^2} \cdot (x - x_o) \cdot (x - x_1) + ... \quad \frac{\Delta^n y_o}{n! h^n} \cdot (x - x_o) \cdot (x - x_1) ... (x - x_{n-1}) \tag{10}$$

A equação acima é conhecida como fórmula de interpolação de Newton (*forward*). Se for admitido a seguinte igualdade:

$$\mathbf{r} = \frac{(\mathbf{x} - \mathbf{x_o})}{\mathbf{h}} \tag{11}$$

Então, o polinômio da equação (10) poderá ser escrito da seguinte forma:

$$\mathbf{y} = \mathbf{y_o} + \Delta \mathbf{y_o} \mathbf{r} + \frac{\Delta^2 \mathbf{y_o}}{\mathbf{2!}} \mathbf{r}(\mathbf{r} - \mathbf{1}) + \ldots + \frac{\Delta^n \mathbf{y_o}}{\mathbf{n!}} \mathbf{r}(\mathbf{r} - \mathbf{1})(\mathbf{r} - \mathbf{2}) \ldots (\mathbf{r} - \mathbf{n} + \mathbf{1}) \tag{12}$$

De maneira similar também existe a fórmula de interpolação de Newton (*backward*) dado por:

$$\mathbf{y} = \mathbf{y_n} + \Delta \mathbf{y_{n-1}} \mathbf{r} + \frac{\Delta^2 \mathbf{y_{n-2}}}{\mathbf{2!}} \mathbf{r}(\mathbf{r} + \mathbf{1}) + \ldots + \frac{\Delta^n \mathbf{y_o}}{\mathbf{n!}} \mathbf{r}(\mathbf{r} + \mathbf{1}) \ldots (\mathbf{r} + \mathbf{n} - \mathbf{1}) \tag{13.a}$$

e

$$\mathbf{r} = \frac{(\mathbf{x} - \mathbf{x_n})}{\mathbf{h}} \tag{13.b}$$

Tipos de integradores mais comuns

Admita **f=f(x,y)** ser uma função real de duas variáveis reais definidas para $\mathbf{a} \le \mathbf{x} \le \mathbf{b}$ onde **a** e **b** são finitos, e para todo valor real de **y**. A equação

$$\mathbf{y}' = \mathbf{f(x,y)} \tag{14}$$

é chamada uma equação diferencial ordinária de primeira ordem; ela simboliza o seguinte problema: achar uma função **y = y(x)**, contínua e diferenciável para $\mathbf{x} \in \mathbf{[a,b]}$, tal que

$$\mathbf{y}'\mathbf{(x)} = \mathbf{f(x,y(x))} \tag{15}$$

para todo $\mathbf{x} \in \mathbf{[a, b]}$. A função **y** com esta propriedade é chamada a solução da equação diferencial (14). Em geral uma equação diferencial pode ter muitas soluções. Para evitar este tipo de problema deve-se especificar o valor da função **y(x)** em um dado ponto, por exemplo, $\mathbf{y(x) = \eta}$ em $\mathbf{x = a}$. Desta maneira, tem-se os chamados *problemas de valor inicial*. O seguinte teorema estabelece a unicidade para este tipo particular de equação diferencial (e. g., Sotomayor Tello, 1979, Lambert, 1973).

Teorema 1. Admita **f(x,y)** ser definida e contínua para todos os pontos **(x,y)** na região **D** definida por $\mathbf{a \le x \le b}$, $\mathbf{-\infty \le y \le \infty}$, **a** e **b** finitos, e admita existir uma constante **L** tal que, para todo **x**, **y**, $\mathbf{y^*}$ tal que **(x,y)** e $\mathbf{(x,y^*)}$ são ambos internos a região **D**,

$$\left|\mathbf{f(x,y) - f(x,y^*)}\right| \le \mathbf{L} \cdot \left|\mathbf{y - y^*}\right| \tag{16}$$

Então, se η é qualquer dado número, existirá uma solução única **y(x)** de valor inicial $\mathbf{y(a)} = \eta$, onde **y(x)** é contínua e diferenciável para todo **(x,y)** interno a **D.** A inequação (16) é conhecida como condição de Lipschitz e a constante **L** como a constante de Lipschitz. Em particular se **f(x,y)** possui uma derivada contínua com respeito a **y** para todo **(x,y)** interno a região **D**, então pelo teorema do valor médio:

$$\mathbf{f(x,y) - f(x,y^*)} = \frac{\partial \mathbf{f(x,\bar{y})}}{\partial \mathbf{y}} \cdot \left(\mathbf{y - y^*}\right) \tag{17}$$

onde $\bar{\mathbf{y}}$ é um ponto no interior do intervalo cujos os pontos extremos são **y** e $\mathbf{y^*}$, e **(x,y)** e $\mathbf{(x,y^*)}$ são ambos interiores a **D**. Claramente, a condição de Lipschitz é então satisfeita se **L** for dada pela seguinte expressão:

$$\mathbf{L} = \sup_{\mathbf{(x,y) \in D}} \left|\frac{\partial\, \mathbf{f(x,y)}}{\partial \mathbf{y}}\right| \tag{18}$$

Em muitos problemas de aplicação pode-se deparar não com uma simples equação diferencial, mas com um sistema de **m** equações simultâneas de primeira ordem com variáveis dependentes $\mathbf{^1y, \ ^2y, \ldots, \ ^my}$. Se cada uma dessas variáveis satisfaz uma dada

condição inicial para o mesmo valor **a** de **x**, então nós temos um problema de valor inicial para um *sistema* de primeira ordem, que pode-se escrever:

$$\begin{aligned} {}^{1}\mathbf{y}' &= {}^{1}\mathbf{f}(\mathbf{x}, {}^{1}\mathbf{y}, {}^{2}\mathbf{y}, \ldots, {}^{m}\mathbf{y}), & {}^{1}\mathbf{y}(\mathbf{a}) &= {}^{1}\eta \\ {}^{2}\mathbf{y}' &= {}^{2}\mathbf{f}(\mathbf{x}, {}^{1}\mathbf{y}, {}^{2}\mathbf{y}, \ldots, {}^{m}\mathbf{y}), & {}^{2}\mathbf{y}(\mathbf{a}) &= {}^{2}\eta \\ &\vdots & &\vdots \\ {}^{m}\mathbf{y}' &= {}^{m}\mathbf{f}(\mathbf{x}, {}^{1}\mathbf{y}, {}^{2}\mathbf{y}, \ldots, {}^{m}\mathbf{y}), & {}^{m}\mathbf{y}(\mathbf{a}) &= {}^{m}\eta \end{aligned} \tag{19}$$

Introduzindo a notação vetorial $\vec{\mathbf{y}} = [{}^{1}\mathbf{y}, {}^{2}\mathbf{y}, \ldots, {}^{m}\mathbf{y}]^{T}$, $\vec{\mathbf{f}} = [{}^{1}\mathbf{f}, {}^{2}\mathbf{f}, \ldots, {}^{m}\mathbf{f}]^{T} = \vec{\mathbf{f}}(\mathbf{x}, \vec{\mathbf{y}})$ e $\vec{\mathbf{n}} = [{}^{1}\mathbf{n}, {}^{2}\mathbf{n}, \ldots, {}^{m}\mathbf{n}]^{T}$ pode-se então escrever o problema de valor inicial (19) na forma:

$$\vec{\mathbf{y}}' = \vec{\mathbf{f}}(\mathbf{x}, \vec{\mathbf{y}}), \qquad \vec{\mathbf{y}}(\mathbf{a}) = \eta \tag{20}$$

A condição de Lipschitz, para o caso vetorial, toma a forma:

$$\left\| \vec{\mathbf{f}}(\mathbf{x}, \vec{\mathbf{y}}) - \vec{\mathbf{f}}(\mathbf{n}, \vec{\mathbf{y}}^{*}) \right\| \leq \mathbf{L} \cdot \left\| \vec{\mathbf{y}} - \vec{\mathbf{y}}^{*} \right\| \tag{21.a}$$

$$\mathbf{L} = \sup_{(\mathbf{x},\mathbf{y}) \in \mathbf{D}} \left\| \frac{\partial \vec{\mathbf{f}}}{\partial \vec{\mathbf{y}}} \right\| \tag{21.b}$$

onde $\partial \vec{\mathbf{f}} / \partial \vec{\mathbf{y}}$ é o Jacobiano de **f** com respeito a $\vec{\mathbf{y}}$ que é uma matriz de dimensão **m x m** e $\| \cdot \|$ denota a norma da matriz. Pode-se demonstrar que uma equação diferencial ordinária de segunda ordem pode ser transformada num sistema de duas equações diferenciais de primeira ordem, uma equação de terceira ordem em um sistema de duas equações diferenciais de segunda ordem e assim por diante. Desta forma, qualquer sistema de alta ordem pode ser transformado num sistema de primeira ordem e recair na situação dada pelas equações (19) ou (20).

Garantida as unicidades da equação diferencial (14) e do sistema de equações diferenciais de primeira ordem (19) ou (20), então pode-se afirmar que nem sempre eles possuirão uma solução analítica, principalmente ser forem do tipo não-linear. Nestes casos somente soluções numéricas poderão ser obtidas. Desta forma, basicamente há

três classes de métodos para resolver um sistema de equações diferenciais de primeira ordem ordinário:

a) método de passo simples, que computa o valor de **y** em apenas um passo $\Delta\mathbf{x}(=\mathbf{h})$ na variável independente **x** por utilizar somente a informação vinda das funções feitas entre **x** e $\mathbf{x}+\Delta\mathbf{x}$ (ponto de vista local);
b) método de múltiplos passos, que computa um passo por utilizar a informação ao longo de vários passos precedentes (ponto de vista global);
c) método de extrapolação, que primeiramente utiliza-se de um método de passo simples para integrar a equação diferencial para vários tamanhos de passos $\mathbf{h_j}$, e então, uma interpolação polinomial é construída através dos valores calculados para gerar os valores desejados de y.

A aproximação geral para uma solução numérica de uma equação diferencial envolve uma seqüência de, igualmente espaçados ou não, pontos discretos $\mathbf{x_o, x_1, x_2 ...}$. Em cada ponto $\mathbf{x_n}$, a solução $\mathbf{y(x_n)}$ é aproximada por um número $\mathbf{y_n}$ que é computado de valores mais anteriores. Isto é, em cada passo, uma aproximação é feita utilizando os valores das aproximações realizadas anteriormente. Em outras palavras, um erro em um dado ponto propaga-se com progresso da integração. Assim, $\mathbf{h_n = x_{n+1} - x_n}$ deve ser ajustado e o erro em $\mathbf{y(x_n)}$ deve ser *estimado.*

Existem basicamente quatro tipos de erros a serem considerados na utilização de integradores numéricos: erro de truncamento local, erro global, erro local e os *round-off* erros, este último relacionado com a precisão alcançada com os computadores digitais.

a) Erro de Truncamento Local

O Erro de Truncamento Local (ETL), τ_{n+1}, em um ponto $\mathbf{x_{n+1}}$ é definido como

$$\tau_{n+1} = \mathbf{y(x_{n+1})} - \mathbf{y_{n+1}} \tag{22}$$

onde $\mathbf{y_{n+1}}$ é o valor computado em $\mathbf{x_{n+1}}$ e $\mathbf{y(x_{n+1})}$ é o valor teórico de **y** em $\mathbf{x_{n+1}}$ calculado por utilizar o mesmo dado utilizado na computação de $\mathbf{y_{n+1}}$. Isto significa que

o ETL é o erro cometido em um passo assumindo que todos os valores anteriores são exatos.

b) Erro Global

O Erro Global (EG), υ_{n+1}, em um ponto $\mathbf{x_{n+1}}$ é definido como

$$\upsilon_{n+1} = \mathbf{y^*(x_{n+1}) - y_{n+1}} \tag{23}$$

onde $\mathbf{y^*(x_{n+1})}$ é o valor verdadeiro de **y** em $\mathbf{x_{n+1}}$ computado por utilizar o dado inicial original.

c) Erro Local

O Erro Local (EL), λ_{n+1}, no ponto $\mathbf{x_{n+1}}$ é definido como

$$\lambda_{n+1} = \mathbf{u_n(x_{n+1}) - y_{n+1}} \tag{24.a}$$

onde $\mathbf{u_n(x)}$ é a solução da equação

$$\mathbf{u_n'(x) = f(x, u_n(x))} \tag{24.b}$$

$$\mathbf{u_n(x) = y_n} \tag{24.c}$$

Métodos de Passo Simples

O método de passo simples é um algoritmo que descreve uma técnica numérica para calcular a aproximação para a solução em $\mathbf{x_{n+1}}$, em termos do valor em um passo anterior ($\mathbf{x_n}$). Os métodos de passo simples são principalmente divididos em duas categorias: métodos em séries de Taylor e métodos de Runge-Kutta.

a) Métodos em Série de Taylor

A idéia básica do método em séries de Taylor é aproximar o valor de $\mathbf{y_{n+1}}$ a partir do ponto $\mathbf{y_n}$ por um polinômio do tipo

$$\mathbf{y_{n+1} = y_n + h \cdot y'_n + \frac{h^2}{2!} \cdot y''_n + ... + \frac{h^k}{k!} \cdot y_n^{(k)}} \qquad (25.a)$$

onde,

$$\mathbf{H = x_{n+1} - x_n} \qquad (25.b)$$

Se a equação acima é utilizada para a aproximação, o método é dito ser de ordem **k.** Por exemplo, o método de Euler é baseado nas séries de Taylor para **k=1** (método de primeira ordem):

$$\mathbf{y_{n+1} = y_n + h \cdot y'_n = y_n + h \cdot f(x_n, y_n)} \qquad (26)$$

O grande inconveniente deste método é determinar as derivadas de ordem superior $\mathbf{y''_n, y'''_n, ...}$ e assim por diante. Por exemplo, $\mathbf{y''_n}$ e $\mathbf{y'''_n}$ são dadas, respectivamente, por (Lapidus et al, 1971):

$$\mathbf{y''_n = f' = f_x + f_y \cdot f} \qquad (27.a)$$

$$\mathbf{y'''_n = f'' = f_{xx} + 2 \cdot f_{xy} \cdot f + f_{yy} \cdot f^2 + (f_x + f_y \cdot f) \cdot f_y} \qquad (27.b)$$

Calcular as derivadas acima nem sempre é uma tarefa fácil, e desta forma, o método das séries de Taylor não possui muita praticidade computacional para integradores de *alta ordem*. Para evitar os problemas das derivadas superiores $\mathbf{f_x, f_y, f_{xx}, f_{yy}, f_{xy}, ...}$ foram desenvolvidos os integradores de Runge-Kutta (R-K) como será visto a seguir.

b) Os Métodos de Runge-Kutta.

Com base nas argumentações levantadas na seção anterior, é muito mais fácil ter um polinômio que concorda com **y** e $\mathbf{y'}$ em $\mathbf{x_n}$ e com $\mathbf{y'}$ (ou **f**) em vários outro argumentos próximos de $\mathbf{x_n}$ do que um polinômio que concorda com **y** e suas **k** derivadas em $\mathbf{x_n}$. Os métodos de R-K são baseados neste princípio. Por exemplo, seja a equação (25.a) expandida até os termos de segunda ordem:

$$\mathbf{y_{n+1} \cong y_n + h \cdot y_n^{'} + \frac{h^2}{2!} \cdot y_n^{''}} \tag{28}$$

Substituindo a equação (27.a) em (28), vem:

$$\mathbf{y \cong y_n + h \cdot f(x_n, y_n) + \frac{h^2}{2} \cdot [f_x(x_n, y_n) + f_y(x_n, y_n) \cdot f(x_n, y_n)]} \tag{29}$$

Para que as derivadas $\mathbf{f_x}$ e $\mathbf{f_y}$ sejam eliminadas recorre-se as seguintes substituições analíticas:

$$\mathbf{k_1 = h \cdot f(x_n, y_n)} \tag{30.a}$$

$$\mathbf{k_2 = h \cdot f(x_n + 1/2, y_n + 1/2)} \tag{30.b}$$

Expandindo novamente em série de Taylor a equação (30.b), vem:

$$\mathbf{k_2 = h \cdot f(x_x, y_n) + h \cdot \left[f_x(x_n, y_n) \cdot \left(\frac{h}{2} \right) + f_y(x_n, y_n) \cdot \left(\frac{k_1}{2} \right) \right]} \tag{31}$$

Substituindo (30.a) em (31) resulta:

$$\mathbf{k_2 = h \cdot f(x_n, y_n) + \frac{h^2}{2} \cdot [f_x(x_n, y_n) + f_y(x_n, y_n) \cdot f(x_n, y_n)]} \tag{32}$$

Fazendo-se nova substituição da equação (32) na equação (29), tem-se:

$$\mathbf{k_1 = h \cdot f(x_n, y_n)} \tag{33.a}$$

$$\mathbf{k_2 = h \cdot f(x_n + h/2, y_n + k_1/2)} \tag{33.b}$$

$$\mathbf{y_{n+1} \cong y_n + k_2} \tag{33.c}$$

As equações (33.a), (33.b) e (33.c) formam, desta maneira, um esquema para obter $\mathbf{y_{n+1}}$ conhecidos os valores de $\mathbf{x_n}$ e $\mathbf{y_n}$, para uma precisão $\mathbf{O(h^2)}$ e envolvendo **f(x,y)** em dois

pontos distintos $(\mathbf{x_n}, \mathbf{y_n})$ e $(\mathbf{x_n} + \mathbf{h/2}, \mathbf{y_n} + \mathbf{k_n/2})$. A equação descrita é o processo de Runge-Kutta de segunda ordem.

O método geral de Runge-Kutta é definido como (Rao, 1984):

$$\mathbf{y_{n+1}} = \mathbf{y_n} + \mathbf{h} \cdot \Phi(\mathbf{x_n}, \mathbf{y_n}, \mathbf{h}) \tag{34.a}$$

onde,

$$\Phi(\mathbf{x_n}, \mathbf{y_n}, \mathbf{h}) = \sum_{\mathbf{r=1}}^{\mathbf{R}} \mathbf{C_r} \cdot \mathbf{k_r} \tag{34.b}$$

$$\mathbf{k_1} = \mathbf{f}(\mathbf{x_n}, \mathbf{y_n}) \tag{34.c}$$

$$\mathbf{k_r} = \mathbf{f}(\mathbf{x_n} + \mathbf{h} \cdot \mathbf{a_r}, \mathbf{y_n} + \mathbf{h} \cdot \sum_{\mathbf{s=1}}^{\mathbf{r-1}} \mathbf{b_{rs}} \cdot \mathbf{k_s}) \tag{34.d}$$

$$\mathbf{a_r} = \sum_{\mathbf{s=1}}^{\mathbf{r-1}} \mathbf{b_{rs}} \qquad \mathbf{r = 2, 3, ..., R} \tag{34.e}$$

Os parâmetros $\mathbf{C_r}$, $\mathbf{a_r}$ e $\mathbf{b_{rs}}$ são determinados por comparar os termos das potências sucessivas de **h** entre (34.a) e o algoritmo de Taylor. Deve ser observado que, em geral, o número de equações não-lineares das condições que surgem de (34.a) à (34.e) em comparação ao algoritmo de Taylor é diferente do número de parâmetros a ser determinados, para uma dada ordem pré-fixada. Com isto, há muitos conjuntos de equações de R-K de mesma ordem propostos por vários autores. A seguir listam-se algumas das expressões mais importantes:

• Fórmula de Heun de terceira ordem (Rao, 1984) ou (Lambert, 1973).

$$\mathbf{y_{n+1}} = \mathbf{y_n} + \frac{\mathbf{h}}{\mathbf{4}} \cdot (\mathbf{k_1} + \mathbf{3} \cdot \mathbf{k_3}) \tag{35.a}$$

$$\mathbf{k_1} = \mathbf{f}(\mathbf{x_n}, \mathbf{y_n}) \tag{35.b}$$

$$\mathbf{k_2 = f(x_n + \frac{1}{3} \cdot h, y_n + \frac{1}{3} \cdot h \cdot k_1)} \quad (35.c)$$

$$\mathbf{k_3 = f(x_n + \frac{2}{3} \cdot h, y_n + \frac{2}{3} \cdot h \cdot k_2)} \quad (35.d)$$

- Fórmula de Kutta de terceira ordem (Rao, 1984).

$$\mathbf{y_{n+1} = y_n + \frac{h}{6} \cdot (k_1 + 4 \cdot k_2 + k_3)} \quad (36.a)$$

$$\mathbf{k_1 = f(x_n, y_n)} \quad (36.b)$$

$$\mathbf{k_2 = f(x_n + \frac{1}{2} \cdot h, y_n + \frac{1}{2} \cdot h \cdot k_1)} \quad (36.c)$$

$$\mathbf{k_3 = f(x_n + h, y_n - h \cdot k_1 + 2 \cdot h \cdot k_2)} \quad (36.d)$$

- Fórmula de R-K de quarta ordem (Lambert, 1973) ou (Henrici, 1964).

$$\mathbf{y_{n+1} = y_n + \frac{h}{6} \cdot (k_1 + 2 \cdot k_2 + 2 \cdot k_3 + k_4)} \quad (37.a)$$

$$\mathbf{k_1 = f(x_n, y_n)} \quad (37.b)$$

$$\mathbf{k_2 = f(x_n + \frac{1}{2} \cdot h, y_n + \frac{1}{2} \cdot h \cdot k_1)} \quad (37.c)$$

$$\mathbf{k_3 = f(x_n + \frac{1}{2} \cdot h, y_n + \frac{1}{2} \cdot h \cdot k_2)} \quad (37.d)$$

$$\mathbf{k_4 = f(x_n + h, y_n + h \cdot k_3)} \quad (37.e)$$

• Método de Kutta-Nystrom de sexta ordem (Lambert, 1973).

$$\mathbf{y_{n+1} = y_n + \frac{h}{192} \cdot (23 \cdot k_1 + 125 \cdot k_3 - 81 \cdot k_5 + 125 \cdot k_6)} \quad (38.a)$$

$$\mathbf{k_1 = f(x_n, y_n)} \quad (38.b)$$

$$\mathbf{k_2 = f(x_n + \frac{1}{3} \cdot h, y_n + \frac{1}{3} \cdot h \cdot k_1)} \quad (38.c)$$

$$\mathbf{k_3 = f(x_n + \frac{2}{5} \cdot h, y_n + \frac{1}{25} \cdot h \cdot (4 \cdot k_1 + 6 \cdot k_2))} \quad (38.d)$$

$$\mathbf{k_4 = f(x_n + h, y_n + \frac{1}{4} \cdot h \cdot (k_1 - 12 \cdot k_2 + 15 \cdot k_3))} \quad (38.e)$$

$$\mathbf{k_5 = f(x_n + \frac{2}{3} \cdot h, y_n + \frac{1}{81} \cdot h \cdot (6 \cdot k_1 + 90 \cdot k_2 - 50 \cdot k_3 + 8 \cdot k_4))} \quad (38.f)$$

$$\mathbf{k_6 = f(x_n + \frac{4}{5} \cdot h, y_n + \frac{1}{75} \cdot h \cdot (6 \cdot k_1 + 36 \cdot k_2 + 10 \cdot k_3 + 8 \cdot k_4))} \quad (38.g)$$

Métodos Lineares de Múltiplos Passos

Há muitos esquemas de integração que dependem de valores em mais do que um ponto prévio para estimar o próximo ponto sobre a curva de solução. Entre estes métodos estão os de Adams-Bashforth, Adams-Moulton, regra de Simpson e Preditivo-Corretor.

Existem basicamente três maneiras de se derivar essas espressões numéricas (Lambert,1973): através de uma expansão em série de Taylor, por integração numérica e através de interpolações.

Neste apêndice será desenvolvido somente os integradores obtidos por integração numérica, pois esta é a metodologia mais empregada. Suponha que para valores igualmente espaçados $\mathbf{x_1, x_2, \ldots, x_n}$ obteve-se as soluções numéricas $\mathbf{y_1, y_2, \ldots, y_n}$ por algum método de passo simples e agora deseja-se obter a solução em $\mathbf{y_{n+1}}$ utilizando-se

um ou mais valores anteriores de **y(x)**. A maneira de se resolver este problema através da *integração numérica* é representar a equação (14) por:

$$\mathbf{y(x_{n+p}) = y(x_n) + \int_{x_n}^{x_{n+p}} f(t, y(t)) \cdot dt} \tag{39}$$

onde, **f(t,y(t))** é a *equação de derivadas* da expressão (14).

Suponha conhecer os pontos $\mathbf{(x, y'_n), (x_{n+1}, y'_{n+1})}$ e $\mathbf{(x_{n+2}, y'_{n+2})}$. Através do polinômio interpolador de Newton que passa através destes três pontos pode-se expressar $\mathbf{y'(t) = f(t, y(t))}$ em termos de diferenças finitas, ou seja:

$$\mathbf{y'(t) = f(t, y(t)) = y'_n + r \cdot \Delta y'_n + \frac{r \cdot (r - 1)}{2!} \cdot \Delta^2 y'_n} \tag{40.a}$$

Ou

$$\mathbf{y'(t) = f(t, y(t)) = f_n + r \cdot \Delta f_n + \frac{r \cdot (r - 1)}{2!} \cdot \Delta^2 f_n} \tag{40.b}$$

onde,

$$\mathbf{r = \frac{t - t_n}{h}} \tag{40.c}$$

Substituindo-se (40.b) em (39), resulta:

$$\mathbf{y(x_{n+2}) = y(x_n) + \int_{x_n}^{x_{n+2}} [f_n + r \cdot \Delta f_n + \frac{r \cdot (r - 1)}{2!} \cdot \Delta^2 f_n] \cdot dt} \tag{41}$$

Sabendo-se que $\mathbf{t = r \cdot h + t_n}$ ou $\mathbf{dt = h \cdot dr}$, então a seguinte substituição pode ser realizada na integral acima:

$$y(x_{n+2}) = y(x_n) + \int_0^2 [f_n + r \cdot \Delta f_n + \frac{r \cdot (r-1)}{2!} \cdot \Delta^2 f_n] \cdot h \cdot dr \tag{42}$$

Resolvendo esta integral em termos da variável r, tem-se:

$$y(x_{n+2}) = y(x_n) + h \cdot (2 \cdot f_n + 2 \cdot \Delta f_n + \frac{1}{3} \cdot \Delta^2 f_n) \tag{43}$$

Expandindo Δf_n e $\Delta^2 f_n$ em termos de f_n, f_{n+1} e f_{n+2} resulta:

$$y(x_{n+2}) = y(x_n) + \frac{h}{3} \cdot (f_{n+2} + 4 \cdot f_{n+1} + f_n) \tag{44}$$

A equação (44) é a famosa regra de Simpson, ou seja, um dos métodos de *dois passos* mais preciso que há. Se a equação (39) for expressa na forma:

$$y(x_{n+2}) - y(x_{n+1}) = \int_{x_{n+1}}^{x_{n+2}} y'(t) \cdot dt \tag{45}$$

Então, se $y'(t)$ for substituida por (40.b) resultará na seguinte identidade (Lambert, 1973):

$$y(x_{n+2}) = y(x_{n+1}) + \frac{h}{12} \cdot (5 \cdot f_{n+2} + 8 \cdot f_{n+1} - f_n) \tag{46}$$

A identidade acima representa o método de *dois passos* de Adams-Moulton. As fórmulas correspondentes para os algoritmos de Adams-Moulton (Vidyasagar, 1978) em termos de $y(x_{n+1})$ com ordens de 1 a 4 são, respectivamente, dadas por:

$$y(x_{n+1}) = y(x_n) + h \cdot f_n \tag{47.a}$$

$$y(x_{n+1}) = y(x_n) + \frac{h}{2} \cdot [f_{n+1} + f_n] \tag{47.b}$$

$$y(x_{n+1}) = y(x_n) + \frac{h}{12} \cdot [\,5 \cdot f_{n+1} + 8 \cdot f_n - f_{n-1}\,] \tag{47.c}$$

$$y(x_{n+1}) = y(x_n) + \frac{h}{24} \cdot [\,9 \cdot f_{n+1} + 19 \cdot f_n - 5 \cdot f_{n-1} + f_{n-2}\,] \tag{47.d}$$

Ainda em (Vidyasagar, 1978) podem ser encontradas as fórmulas dos algoritmos de Adams-Bashforth de ordens de 1 a 4 como listadas abaixo:

$$y(x_{n+1}) = y(x_n) + h \cdot f_n \tag{48.a}$$

$$y(x_{n+1}) = y(x_n) + \frac{h}{2} \cdot [\,3 \cdot f_n - f_{n-1}\,] \tag{48.b}$$

$$y(x_{n+1}) = y(x_n) + \frac{h}{12} \cdot [\,21 \cdot f_n - 16 \cdot f_{n-1} + 5 \cdot f_{n-2}\,] \tag{48.c}$$

$$y(x_{n+1}) = y(x_n) + \frac{h}{24} \cdot [\,55 \cdot f_n - 59 \cdot f_{n-1} + 37 \cdot f_{n-2} - 9 \cdot f_{n-3}\,] \tag{48.d}$$

Para obter a fórmula do método preditor utiliza-se a fórmula de interpolação de Newton com diferenças para trás (Δ), ou seja:

$$y'(t) = f_n + \Delta f_{n-1} \cdot r + \frac{\Delta^2 f_{n-2}}{2!} \cdot r \cdot (r-1) + \frac{\Delta^3 f_{n-3}}{3!} \cdot r \cdot (r+1) \cdot (r+2) + \ldots \tag{49}$$

Substituindo a equação (49) na integral $\int_{x_n}^{x_{n+1}} y'(t) \cdot dt$ resulta que (Pennington, 1970):

$$y(x_{n+1}) - y(x_n) = h \cdot [f_n + \frac{1}{2} \cdot \Delta f_{n-1} + \frac{5}{12} \cdot \Delta^2 f_{n-2} + \left(\frac{3}{8}\right) \cdot \Delta^3 f_{n-3} + \ldots] \tag{50}$$

Para se obter a fórmula do método corretor rescreve-se a equação (49) em termos de x_{n+1} ao invés de x_n. A teoria dos integradores numéricos vão muito além daquilo que foi exposto neste apêndice, mas isto é mais do que suficiente para o desenvolvimento do trabalho que se pretende realizar nesta tese. Entretanto, apenas para fins ilustrativos,

ainda existem os integradores de passo variável combinados com as técnicas numéricas já apresentadas, parcialmente, neste apêndice.

APÊNDICE E

CÁLCULO DAS DERIVADAS PARCIAIS PARA UMA ESTRUTURA DE CONTROLE PREDITIVO TRABALHANDO COM AS DERIVADAS MÉDIAS

Nesta apêndice serão deduzidas as expressões das derivadas parciais $\frac{\partial\, {}^{k+n}\mathbf{y}^{i}}{\partial\, {}^{k}\mathbf{u}}$ para o caso geral onde ${}^{k+n}\mathbf{y}^{i}$ e ${}^{k}\mathbf{u}$ são grandezas vetoriais e **n** o horizonte de previsão adotado pela estrutura de controle preditivo e como visto nos Capítulos 4 e 5. Estas derivadas são requeridas pela equação (10) apresentada no Capítulo 4 e a determinação destas derivadas parciais é também funções da retro-propagação, como expressa pelas equações de (16.a) à (16.c) também do Capítulo 4 ou encontradas deduzidas no Apêndice C, e da estrutura de integração adotada para discretizar a dinâmica do sistema não-linear. O que se fará nesta seção é aplicar a regra da cadeia combinada com a retro-propagação sobre a estrutura particular de um integrador para se obter as expressões matriciais para $\frac{\partial\, {}^{k+n}\mathbf{y}^{i}}{\partial\, {}^{k}\mathbf{u}}$. O integrador adotado é o com estrutura de Euler, pois este é que foi utilizado para desenvolver o método das derivadas médias, como visto no Capítulo 5, como uma nova metodologia para a representação de sistemas dinâmicos através de uma estrutura neural de integração.

As expressões analíticas para a estrutura de integração do tipo Euler adotadas pelo método das derivadas médias são:

$$\begin{Bmatrix} {}^{k+1}\mathbf{y}_1^{i} \\ {}^{k+1}\mathbf{y}_2^{i} \\ \vdots \\ {}^{k+1}\mathbf{y}_{n_{y'}}^{i} \end{Bmatrix} = \begin{Bmatrix} \tan_{\Delta t}\, {}^{k}\alpha_1^{i} \cong \hat{\mathbf{f}}_1({}^{k}\mathbf{y}^{i}, {}^{k}\mathbf{u}, \hat{\mathbf{w}}) \\ \tan_{\Delta t}\, {}^{k}\alpha_2^{i} \cong \hat{\mathbf{f}}_2({}^{k}\mathbf{y}^{i}, {}^{k}\mathbf{u}, \hat{\mathbf{w}}) \\ \vdots \\ \tan_{\Delta t}\, {}^{k}\alpha_{n_{y'}}^{i} \cong \hat{\mathbf{f}}_{n_{y'}}({}^{k}\mathbf{y}^{i}, {}^{k}\mathbf{u}, \hat{\mathbf{w}}) \end{Bmatrix} \Delta t + \begin{Bmatrix} {}^{k}\mathbf{y}_1^{i} \\ {}^{k}\mathbf{y}_2^{i} \\ \vdots \\ {}^{k}\mathbf{y}_{n_y}^{i} \end{Bmatrix} \tag{1}$$

A equação (1) ainda pode ser expressa pela seguinte notação compacta:

$${}^{k+1}\mathbf{y}^{i} = \tan_{\Delta t}\, {}^{k}\alpha^{i} \cdot \Delta t + {}^{k}\mathbf{y}^{i} \cong \hat{\mathbf{f}}({}^{k}\mathbf{y}^{i}, {}^{k}\mathbf{u}, \hat{\mathbf{w}}) \cdot \Delta t + {}^{k}\mathbf{y}^{i} \tag{2}$$

onde,

$\hat{\mathbf{f}}(^{\mathbf{k}}\mathbf{y}^{\mathbf{i}}, {}^{\mathbf{k}}\mathbf{u}, \hat{\mathbf{w}})$ ... é uma estimativa de $\mathbf{tan}_{\Delta \mathbf{t}}{}^{\mathbf{k}}\alpha^{\mathbf{i}}$ obtida pela rede neural;

$\mathbf{n_{y'}}$ ... é o número total de variáveis de estado e que não pode ser confundido com $\mathbf{n_y}$ que é o número de entradas atrasadas da rede neural.

As expressões que serão desenvolvidas a seguir serão válidas somente para o caso de redes neurais projetadas com apenas uma entrada atrasada, ou seja, $\mathbf{n_y} = \mathbf{1}$. Seja então, inicialmente o caso onde **n=1**. Assim, tem-se de imediato que:

Se $^{\mathbf{k+1}}\mathbf{y}^{\mathbf{i}} \cong \hat{\mathbf{f}}(^{\mathbf{k}}\mathbf{y}^{\mathbf{i}}, {}^{\mathbf{k}}\mathbf{u}, \hat{\mathbf{w}}) \cdot \Delta \mathbf{t} + {}^{\mathbf{k}}\mathbf{y}^{\mathbf{i}}$ então,

$$\frac{\partial^{\mathbf{k+1}}\mathbf{y}^{\mathbf{i}}}{\partial\, {}^{\mathbf{k}}\mathbf{u}} = \frac{\partial\, \hat{\mathbf{f}}(^{\mathbf{k}}\mathbf{y}^{\mathbf{i}}, {}^{\mathbf{k}}\mathbf{u}, \hat{\mathbf{w}})}{\partial\, {}^{\mathbf{k}}\mathbf{u}} \cdot \Delta \mathbf{t} \cong \frac{\partial\, \mathbf{tan}_{\Delta \mathbf{t}}{}^{\mathbf{k}}\alpha^{\mathbf{i}}}{\partial\, {}^{\mathbf{k}}\mathbf{u}} \cdot \Delta \mathbf{t} \qquad (3)$$

para,

$$\frac{\partial\, \hat{\mathbf{f}}(^{\mathbf{k}}\mathbf{y}^{\mathbf{i}}, {}^{\mathbf{k}}\mathbf{u}, \hat{\mathbf{w}})}{\partial\, {}^{\mathbf{k}}\mathbf{u}} \cong \frac{\partial\, \mathbf{tan}_{\Delta \mathbf{t}}{}^{\mathbf{k}}\alpha^{\mathbf{i}}}{\partial\, {}^{\mathbf{k}}\mathbf{u}} = \begin{bmatrix} \frac{\partial\, \mathbf{tan}_{\Delta \mathbf{t}}{}^{\mathbf{k}}\alpha_{\mathbf{1}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{1}}} & \frac{\partial\, \mathbf{tan}_{\Delta \mathbf{t}}{}^{\mathbf{k}}\alpha_{\mathbf{1}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{2}}} & \cdots & \frac{\partial\, \mathbf{tan}_{\Delta \mathbf{t}}{}^{\mathbf{k}}\alpha_{\mathbf{1}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{n_{u'}}}} \\ \frac{\partial\, \mathbf{tan}_{\Delta \mathbf{t}}{}^{\mathbf{k}}\alpha_{\mathbf{2}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{1}}} & \frac{\partial\, \mathbf{tan}_{\Delta \mathbf{t}}{}^{\mathbf{k}}\alpha_{\mathbf{2}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{2}}} & \cdots & \frac{\partial\, \mathbf{tan}_{\Delta \mathbf{t}}{}^{\mathbf{k}}\alpha_{\mathbf{2}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{n_{u'}}}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\, \mathbf{tan}_{\Delta \mathbf{t}}{}^{\mathbf{k}}\alpha_{\mathbf{n_{y'}}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{1}}} & \frac{\partial\, \mathbf{tan}_{\Delta \mathbf{t}}{}^{\mathbf{k}}\alpha_{\mathbf{n_{y'}}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{2}}} & \cdots & \frac{\partial\, \mathbf{tan}_{\Delta \mathbf{t}}{}^{\mathbf{k}}\alpha_{\mathbf{n_{y'}}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{n_{u'}}}} \end{bmatrix}_{\mathbf{n_{y'}\ x\ n_{u'}}}$$

(4)

onde,

$\mathbf{n_{y'}}$ ... é o número total de variáveis de estado;

$\mathbf{n_{u'}}$ ... é o número total de variáveis de controle.

A equação (4) ainda pode ser expressa na forma escalar, como segue:

$$\frac{\partial^{\mathbf{k+1}}\mathbf{y}_{\mathbf{j}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{l}}} = \frac{\partial\,\mathbf{tan}_{\Delta\mathbf{t}}{}^{\mathbf{k}}\alpha_{\mathbf{j}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{l}}}\cdot\Delta\mathbf{t} \quad \text{para } \mathbf{j=1, 2, ..., n_y}\text{, e } \mathbf{l=1, 2, ..., n_u}, \tag{5}$$

Cada derivada presente na matriz dada pela equação (4) pode ser calculada através das expressões de (16.a) à (16.c) do Capítulo 4 referentes a retro-propagação. Seja agora então, o caso para **n=2**. Assim, tem-se:

$$\mathbf{k+2}\mathbf{y}^{\mathbf{i}} = \left(\mathbf{tan}_{\Delta\mathbf{t}}{}^{\mathbf{k+1}}\alpha^{\mathbf{i}} + \mathbf{tan}_{\Delta\mathbf{t}}{}^{\mathbf{k}}\alpha^{\mathbf{i}}\right)\cdot\Delta\mathbf{t} + {}^{\mathbf{k}}\mathbf{y}^{\mathbf{i}} \tag{6.a}$$

ou

$$\mathbf{k+2}\mathbf{y}^{\mathbf{i}} \cong \hat{\mathbf{f}}({}^{\mathbf{k+1}}\mathbf{y}^{\mathbf{i}}, {}^{\mathbf{k+1}}\mathbf{u}, \hat{\mathbf{w}})\cdot\Delta\mathbf{t} + \hat{\mathbf{f}}({}^{\mathbf{k}}\mathbf{y}^{\mathbf{i}}, {}^{\mathbf{k}}\mathbf{u}, \hat{\mathbf{w}})\cdot\Delta\mathbf{t} + {}^{\mathbf{k}}\mathbf{y}^{\mathbf{i}} \tag{6.b}$$

A equação (6.b) pode também ser expressa na forma escalar como segue:

$$\mathbf{k+2}\mathbf{y}_{\mathbf{j}}^{\mathbf{i}} \cong \hat{\mathbf{f}}_{\mathbf{j}}[\underbrace{\hat{\mathbf{f}}({}^{\mathbf{k}}\mathbf{y}^{\mathbf{i}}, {}^{\mathbf{k}}\mathbf{u}, \hat{\mathbf{w}})\cdot\Delta\mathbf{t} + {}^{\mathbf{k}}\mathbf{y}^{\mathbf{i}}}_{\mathbf{k+1}\mathbf{y}^{\mathbf{i}}}, {}^{\mathbf{k+1}}\mathbf{u}, \hat{\mathbf{w}}]\cdot\Delta\mathbf{t} + \hat{\mathbf{f}}_{\mathbf{j}}({}^{\mathbf{k}}\mathbf{y}^{\mathbf{i}}, {}^{\mathbf{k}}\mathbf{u}, \hat{\mathbf{w}})\cdot\Delta\mathbf{t} + {}^{\mathbf{k}}\mathbf{y}_{\mathbf{j}}^{\mathbf{i}}$$

(7)

para, $\mathbf{j=1, 2, ..., n_y}$.

Com base na equação anterior, tem-se:

$$\frac{\partial\,{}^{\mathbf{k+2}}\mathbf{y}_{\mathbf{j}}^{\mathbf{i}}}{\partial\,{}^{\mathbf{k}}\mathbf{u}_{\mathbf{l}}} = \frac{\partial}{\partial\,{}^{\mathbf{k}}\mathbf{u}_{\mathbf{l}}}\cdot\underbrace{\{\hat{\mathbf{f}}_{\mathbf{j}}[\hat{\mathbf{f}}({}^{\mathbf{k}}\mathbf{y}^{\mathbf{i}}, {}^{\mathbf{k}}\mathbf{u}, \hat{\mathbf{w}})\cdot\Delta\mathbf{t} + {}^{\mathbf{k}}\mathbf{y}^{\mathbf{i}}, {}^{\mathbf{k+1}}\mathbf{u}, \hat{\mathbf{w}}]\}}_{\mathbf{J}_{\mathbf{j}}^{\mathbf{1}}}\cdot\Delta\mathbf{t}$$

$$+\underbrace{\frac{\partial}{\partial\,{}^{\mathbf{k}}\mathbf{u}_{\mathbf{l}}}\cdot\hat{\mathbf{f}}_{\mathbf{j}}({}^{\mathbf{k}}\mathbf{y}^{\mathbf{i}}, {}^{\mathbf{k}}\mathbf{u}, \hat{\mathbf{w}})\cdot\Delta\mathbf{t}}_{\frac{\partial\,{}^{\mathbf{k+1}}\mathbf{y}_{\mathbf{j}}^{\mathbf{i}}}{\partial\,{}^{\mathbf{k}}\mathbf{u}_{\mathbf{l}}}}} + \mathbf{0} \tag{8}$$

A derivada $\frac{\partial \mathbf{J}_{\mathbf{j}}^{\mathbf{1}}}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{l}}}$ pode ser determinada pelo emprego da regra da cadeia, ou seja:

$$\frac{\partial \mathbf{J}_{\mathbf{j}}^{\mathbf{1}}}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{l}}} = \frac{\partial}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{l}}}\left\{\overbrace{\hat{\mathbf{f}}_{\mathbf{j}}[\underbrace{\hat{\mathbf{f}}(^{\mathbf{k}}\mathbf{y}^{\mathbf{i}}, {}^{\mathbf{k}}\mathbf{u}, \hat{\mathbf{w}}) \cdot \Delta\mathbf{t} + {}^{\mathbf{k}}\mathbf{y}^{\mathbf{i}}}_{{}^{\mathbf{k+1}}\mathbf{y}^{\mathbf{i}}}, {}^{\mathbf{k+1}}\mathbf{u}, \hat{\mathbf{w}}]}^{\mathbf{J}_{\mathbf{j}}^{\mathbf{1}} \cong \tan_{\Delta\mathbf{t}}{}^{\mathbf{k+1}}\alpha_{\mathbf{j}}^{\mathbf{i}}}\right\} =$$

$$\frac{\partial}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{l}}} \cdot \left\{\hat{\mathbf{f}}_{\mathbf{j}}[\overbrace{\hat{\mathbf{f}}_{\mathbf{1}}(^{\mathbf{k}}\mathbf{y}^{\mathbf{i}}, {}^{\mathbf{k}}\mathbf{u}, \hat{\mathbf{w}}) \cdot \Delta\mathbf{t} + {}^{\mathbf{k}}\mathbf{y}_{\mathbf{1}}^{\mathbf{i}}}^{{}^{\mathbf{k+1}}\mathbf{y}_{\mathbf{1}}^{\mathbf{i}}}, \cdots, \overbrace{\hat{\mathbf{f}}_{\mathbf{n_{y'}}}(^{\mathbf{k}}\mathbf{y}^{\mathbf{i}}, {}^{\mathbf{k}}\mathbf{u}, \hat{\mathbf{w}}) \cdot \Delta\mathbf{t} + {}^{\mathbf{k}}\mathbf{y}_{\mathbf{n_{y'}}}^{\mathbf{i}}}^{{}^{\mathbf{k+1}}\mathbf{y}_{\mathbf{n_{y'}}}^{\mathbf{i}}}, {}^{\mathbf{k+1}}\mathbf{u}, \hat{\mathbf{w}}]\right\} =$$

$$\frac{\partial \mathbf{J}_{\mathbf{j}}^{\mathbf{1}}}{\partial {}^{\mathbf{k+1}}\mathbf{y}_{\mathbf{1}}^{\mathbf{i}}} \cdot \frac{\partial {}^{\mathbf{k+1}}\mathbf{y}_{\mathbf{1}}^{\mathbf{i}}}{\partial {}^{\mathbf{k}}\mathbf{u}_{\mathbf{l}}} + \frac{\partial \mathbf{J}_{\mathbf{j}}^{\mathbf{1}}}{\partial {}^{\mathbf{k+1}}\mathbf{y}_{\mathbf{2}}^{\mathbf{i}}} \cdot \frac{\partial {}^{\mathbf{k+1}}\mathbf{y}_{\mathbf{2}}^{\mathbf{i}}}{\partial {}^{\mathbf{k}}\mathbf{u}_{\mathbf{l}}} + \cdots + \frac{\partial \mathbf{J}_{\mathbf{j}}^{\mathbf{1}}}{\partial {}^{\mathbf{k+1}}\mathbf{y}_{\mathbf{n_{y'}}}^{\mathbf{i}}} \cdot \frac{\partial {}^{\mathbf{k+1}}\mathbf{y}_{\mathbf{n_{y'}}}^{\mathbf{i}}}{\partial {}^{\mathbf{k}}\mathbf{u}_{\mathbf{l}}} \qquad (9)$$

Na forma de somatório a equação (9) se reduz a:

$$\frac{\partial \mathbf{J}_{\mathbf{j}}^{\mathbf{1}}}{\partial {}^{\mathbf{k}}\mathbf{u}_{\mathbf{l}}} = \sum_{\mathbf{p=1}}^{\mathbf{n_{y'}}} \frac{\partial \mathbf{J}_{\mathbf{j}}^{\mathbf{1}}}{\partial {}^{\mathbf{k+1}}\mathbf{y}_{\mathbf{p}}^{\mathbf{i}}} \cdot \frac{\partial {}^{\mathbf{k+1}}\mathbf{y}_{\mathbf{p}}^{\mathbf{i}}}{\partial {}^{\mathbf{k}}\mathbf{u}_{\mathbf{l}}} = \sum_{\mathbf{p=1}}^{\mathbf{n_{y'}}} \frac{\partial \tan_{\Delta\mathbf{t}}{}^{\mathbf{k+1}}\alpha_{\mathbf{j}}^{\mathbf{i}}}{\partial {}^{\mathbf{k+1}}\mathbf{y}_{\mathbf{p}}^{\mathbf{i}}} \cdot \frac{\partial {}^{\mathbf{k+1}}\mathbf{y}_{\mathbf{p}}^{\mathbf{i}}}{\partial {}^{\mathbf{k}}\mathbf{u}_{\mathbf{l}}} = \sum_{\mathbf{p=1}}^{\mathbf{n_{y'}}} \frac{\partial \tan_{\Delta\mathbf{t}}{}^{\mathbf{k+1}}\alpha_{\mathbf{j}}^{\mathbf{i}}}{\partial {}^{\mathbf{k+1}}\mathbf{y}_{\mathbf{p}}^{\mathbf{i}}} \cdot \frac{\partial \tan_{\Delta\mathbf{t}}{}^{\mathbf{k}}\alpha_{\mathbf{p}}^{\mathbf{i}}}{\partial {}^{\mathbf{k}}\mathbf{u}_{\mathbf{l}}} \cdot \Delta\mathbf{t}$$

(10)

Substituindo a equação (10) na equação (8), tem-se finalmente que:

$$\frac{\partial {}^{\mathbf{k+2}}\mathbf{y}_{\mathbf{j}}^{\mathbf{i}}}{\partial {}^{\mathbf{k}}\mathbf{u}_{\mathbf{l}}} = \left(\sum_{\mathbf{p=1}}^{\mathbf{n_{y'}}} \frac{\partial \tan_{\Delta\mathbf{t}}{}^{\mathbf{k+1}}\alpha_{\mathbf{j}}^{\mathbf{i}}}{\partial {}^{\mathbf{k+1}}\mathbf{y}_{\mathbf{p}}^{\mathbf{i}}} \cdot \frac{\partial \tan_{\Delta\mathbf{t}}{}^{\mathbf{k}}\alpha_{\mathbf{p}}^{\mathbf{i}}}{\partial {}^{\mathbf{k}}\mathbf{u}_{\mathbf{l}}}\right) \cdot \Delta\mathbf{t}^{\mathbf{2}} + \frac{\partial \tan_{\Delta\mathbf{t}}{}^{\mathbf{k}}\alpha_{\mathbf{j}}^{\mathbf{i}}}{\partial {}^{\mathbf{k}}\mathbf{u}_{\mathbf{l}}} \cdot \Delta\mathbf{t} \qquad (11)$$

para, **j=1, 2, ..., $\mathbf{n_y}$,** e **l=1, 2, ..., $\mathbf{n_u}$,**.

A equação (11) está na forma escalar. Entretanto, é possível colocá-la na forma matricial, como segue:

$$\frac{\partial\,^{k+2}\mathbf{y}^{i}}{\partial\,^{k}\mathbf{u}} = \frac{\partial\,\tan_{\Delta t}{}^{k+1}\alpha^{i}}{\partial\,^{k+1}\mathbf{y}^{i}} \cdot \frac{\partial\,\tan_{\Delta t}{}^{k}\alpha^{i}}{\partial^{k}\mathbf{u}} \cdot \Delta t^{2} + \underbrace{\frac{\partial\,\tan_{\Delta t}{}^{k}\alpha^{i}}{\partial^{k}\mathbf{u}}}_{\textbf{equação (71)}} \cdot \Delta t \tag{12.a}$$

ou

$$\frac{\partial\,^{k+2}\mathbf{y}^{i}}{\partial\,^{k}\mathbf{u}} = \left(\frac{\partial\,\tan_{\Delta t}{}^{k+1}\alpha^{i}}{\partial\,^{k+1}\mathbf{y}^{i}} \cdot \Delta t + \mathbf{I} \right) \cdot \underbrace{\frac{\partial\,\tan_{\Delta t}{}^{k}\alpha^{i}}{\partial\,^{k}\mathbf{u}} \cdot \Delta t}_{\frac{\partial\,^{k+1}\mathbf{y}^{i}}{\partial\,^{k}\mathbf{u}}} \tag{12.b}$$

A matriz $\frac{\partial\,^{k+1}\mathbf{y}^{i}}{\partial^{k}\mathbf{u}}$, ao lado direita da equação (12.b), pode ser obtida através das equações (3) e (4) e a outra matriz $\frac{\partial\,\tan_{\Delta t}{}^{k+1}\alpha^{i}}{\partial\,^{k+1}\mathbf{y}^{i}}$ é definida como:

$$\frac{\partial\,\tan_{\Delta t}{}^{k+1}\alpha^{i}}{\partial\,^{k+1}\mathbf{y}^{i}} = \begin{bmatrix} \frac{\partial\,\tan_{\Delta t}{}^{k+1}\alpha_{1}^{i}}{\partial^{k+1}\mathbf{y}_{1}^{i}} & \frac{\partial\,\tan_{\Delta t}{}^{k+1}\alpha_{1}^{i}}{\partial^{k+1}\mathbf{y}_{2}^{i}} & \cdots & \frac{\partial\,\tan_{\Delta t}{}^{k+1}\alpha_{1}^{i}}{\partial^{k+1}\mathbf{y}_{n_{y'}}^{i}} \\ \frac{\partial\,\tan_{\Delta t}{}^{k+1}\alpha_{2}^{i}}{\partial^{k+1}\mathbf{y}_{1}^{i}} & \frac{\partial\,\tan_{\Delta t}{}^{k+1}\alpha_{2}^{i}}{\partial^{k+1}\mathbf{y}_{2}^{i}} & \cdots & \frac{\partial\,\tan_{\Delta t}{}^{k+1}\alpha_{2}^{i}}{\partial^{k+1}\mathbf{y}_{n_{y'}}^{i}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\,\tan_{\Delta t}{}^{k+1}\alpha_{n_{y'}}^{i}}{\partial^{k+1}\mathbf{y}_{1}^{i}} & \frac{\partial\,\tan_{\Delta t}{}^{k+1}\alpha_{n_{y'}}^{i}}{\partial^{k+1}\mathbf{y}_{2}^{i}} & \cdots & \frac{\partial\,\tan_{\Delta t}{}^{k+1}\alpha_{n_{y'}}^{i}}{\partial^{k+1}\mathbf{y}_{n_{y'}}^{i}} \end{bmatrix}_{\mathbf{n_{y'}\ x\ n_{y'}}} \tag{13}$$

para **n=3**, tem-se:

$$\frac{\partial\,^{k+3}\mathbf{y}^{i}}{\partial\,^{k}\mathbf{u}} = \frac{\partial\,\tan_{\Delta t}{}^{k+2}\alpha^{i}}{\partial\,^{k}\mathbf{u}} \cdot \Delta t + \frac{\partial\,\tan_{\Delta t}{}^{k+1}\alpha^{i}}{\partial\,^{k}\mathbf{u}} \cdot \Delta t + \frac{\partial\,\tan_{\Delta t}{}^{k}\alpha^{i}}{\partial\,^{k}\mathbf{u}} \cdot \Delta t \tag{14.a}$$

ou

$$\frac{\partial\,{}^{k+3}\mathbf{y}^{i}}{\partial\,{}^{k}\mathbf{u}} \cong \frac{\partial\,\hat{\mathbf{f}}({}^{k+2}\mathbf{y}^{i},{}^{k+2}\mathbf{u},\hat{\mathbf{w}})}{\partial\,{}^{k}\mathbf{u}}\cdot\Delta\mathbf{t} + \frac{\partial\,\hat{\mathbf{f}}({}^{k+1}\mathbf{y}^{i},{}^{k+1}\mathbf{u},\hat{\mathbf{w}})}{\partial\,{}^{k}\mathbf{u}}\cdot\Delta\mathbf{t} + \frac{\partial\,\hat{\mathbf{f}}({}^{k}\mathbf{y}^{i},{}^{k}\mathbf{u},\hat{\mathbf{w}})}{\partial\,{}^{k}\mathbf{u}}\cdot\Delta\mathbf{t}$$

(14.b)

A matriz de derivadas $\frac{\partial\,\mathbf{tan}_{\Delta\mathbf{t}}{}^{k+2}\alpha^{i}}{\partial\,{}^{k}\mathbf{u}}$ presente na equação (14.a) pode ser obtida da seguinte forma:

$$\frac{\partial\,\mathbf{tan}_{\Delta\mathbf{t}}{}^{k+2}\alpha^{i}}{\partial\,{}^{k}\mathbf{u}} \cong \frac{\partial\,\hat{\mathbf{f}}({}^{k+2}\mathbf{y}^{i},{}^{k+2}\mathbf{u},\hat{\mathbf{w}})}{\partial\,{}^{k}\mathbf{u}} = \frac{\partial\,\hat{\mathbf{f}}[\hat{\mathbf{f}}({}^{k+1}\mathbf{y}^{i},{}^{k+1}\mathbf{u},\hat{\mathbf{w}})\cdot\Delta\mathbf{t} + {}^{k+1}\mathbf{y}^{i}]}{\partial\,{}^{k}\mathbf{u}} \quad (15)$$

Aplicando a regra da cadeia sobre a equação (15), vem:

$$\frac{\partial\,\hat{\mathbf{f}}({}^{k+2}\mathbf{y}^{i},{}^{k+2}\mathbf{u},\hat{\mathbf{w}})}{\partial\,{}^{k}\mathbf{u}} \cong \frac{\partial\,\mathbf{tan}_{\Delta\mathbf{t}}{}^{k+2}\alpha^{i}}{\partial\,{}^{k}\mathbf{u}} = \frac{\partial\,\mathbf{tan}_{\Delta\mathbf{t}}{}^{k+2}\alpha^{i}}{\partial\,{}^{k+2}\mathbf{y}^{i}}\cdot\frac{\partial\,{}^{k+2}\mathbf{y}^{i}}{\partial\,{}^{k}\mathbf{u}} \quad (16)$$

A demonstração da equação (16) pode ser realizada de maneira análoga àquela feita para deduzir a equação (10). Logo, substituindo a equação (16) na equação (14.a), vem:

$$\frac{\partial\,{}^{k+3}\mathbf{y}^{i}}{\partial\,{}^{k}\mathbf{u}} = \frac{\partial\,\mathbf{tan}_{\Delta\mathbf{t}}{}^{k+2}\alpha^{i}}{\partial\,{}^{k+2}\mathbf{y}^{i}}\cdot\frac{\partial\,{}^{k+2}\mathbf{y}^{i}}{\partial\,{}^{k}\mathbf{u}}\cdot\Delta\mathbf{t} + \underbrace{\frac{\partial\,\mathbf{tan}_{\Delta\mathbf{t}}{}^{k+1}\alpha^{i}}{\partial\,{}^{k}\mathbf{u}}\cdot\Delta\mathbf{t} + \frac{\partial\mathbf{tan}_{\Delta\mathbf{t}}{}^{k}\alpha^{i}}{\partial\,{}^{k}\mathbf{u}}\cdot\Delta\mathbf{t}}_{\frac{\partial\,{}^{k+2}\mathbf{y}^{i}}{\partial\,{}^{k}\mathbf{u}}\ \{\text{ver a equação (75)}\}}$$

(17)

Assim, observando as equações (17) e (12.b) pode-se deduzir a seguinte equação de recorrência para o cálculo das derivadas parciais $\frac{\partial\,{}^{k+3}\mathbf{y}^{i}}{\partial\,{}^{k}\mathbf{u}}$:

$$\frac{\partial\,{}^{k+p}\mathbf{y}^{i}}{\partial\,{}^{k}\mathbf{u}} = \left(\frac{\partial\,\mathbf{tan}_{\Delta\mathbf{t}}{}^{k+p-1}\alpha^{i}}{\partial^{k+p-1}\mathbf{y}^{i}}\cdot\Delta\mathbf{t} + \mathbf{I}\right)\cdot\frac{\partial^{k+p-1}\mathbf{y}^{i}}{\partial^{k}\mathbf{u}} \quad \text{para } \mathbf{q=2, 3} \quad (18)$$

A equação (18) sugere então, as seguintes equações recorrentes para o cálculo de $\frac{\partial^{\mathbf{k+n}}\mathbf{y}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}}$:

$$\frac{\partial^{\mathbf{k+q}}\mathbf{y}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}} = \left(\frac{\partial \tan_{\Delta \mathbf{t}}{}^{\mathbf{k+q-1}}\alpha^{\mathbf{i}}}{\partial^{\mathbf{k+q-1}}\mathbf{y}^{\mathbf{i}}} \cdot \Delta \mathbf{t} + \mathbf{I} \right) \cdot \frac{\partial^{\mathbf{k+q-1}}\mathbf{y}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}} \quad \text{para } \mathbf{q=2, 3, ..., n} \qquad (19.a)$$

onde,

$$\frac{\partial^{\mathbf{k+q-1}}\mathbf{y}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}} = \Delta \mathbf{t} \cdot \begin{bmatrix} \frac{\partial \tan_{\Delta \mathbf{t}}{}^{\mathbf{k+q-2}}\alpha_{\mathbf{1}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u_1}} & \frac{\partial \tan_{\Delta \mathbf{t}}{}^{\mathbf{k+q-2}}\alpha^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u_2}} & \cdots & \frac{\partial \tan_{\Delta \mathbf{t}}{}^{\mathbf{k+q-2}}\alpha_{\mathbf{1}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{n_{u'}}}} \\ \frac{\partial \tan_{\Delta \mathbf{t}}{}^{\mathbf{k+q-2}}\alpha_{\mathbf{2}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u_1}} & \frac{\partial \tan_{\Delta \mathbf{t}}{}^{\mathbf{k+q-2}}\alpha_{\mathbf{2}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u_2}} & \cdots & \frac{\partial \tan_{\Delta \mathbf{t}}{}^{\mathbf{k+q-2}}\alpha_{\mathbf{2}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{n_{u'}}}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \tan_{\Delta \mathbf{t}}{}^{\mathbf{k+q-2}}\alpha_{\mathbf{n_{y'}}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u_1}} & \frac{\partial \tan_{\Delta \mathbf{t}}{}^{\mathbf{k+q-2}}\alpha_{\mathbf{n_{y'}}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u_2}} & \cdots & \frac{\partial \tan_{\Delta \mathbf{t}}{}^{\mathbf{k+q-2}}\alpha_{\mathbf{n_{y'}}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{n_{u'}}}} \end{bmatrix}_{\mathbf{n_{y'}\ x\ n_{u'}}} \qquad (19.b)$$

$$\frac{\partial \tan_{\Delta \mathbf{t}}{}^{\mathbf{k+q-1}}\alpha^{\mathbf{i}}}{\partial^{\mathbf{k+q-1}}\mathbf{y}^{\mathbf{i}}} = \begin{bmatrix} \frac{\partial \tan_{\Delta \mathbf{t}}{}^{\mathbf{k+q-1}}\alpha_{\mathbf{1}}^{\mathbf{i}}}{\partial^{\mathbf{k+q-1}}\mathbf{y}_{\mathbf{1}}^{\mathbf{i}}} & \frac{\partial \tan_{\Delta \mathbf{t}}{}^{\mathbf{k+q-1}}\alpha_{\mathbf{1}}^{\mathbf{i}}}{\partial^{\mathbf{k+q-1}}\mathbf{y}_{\mathbf{2}}^{\mathbf{i}}} & \cdots & \frac{\partial \tan_{\Delta \mathbf{t}}{}^{\mathbf{k+q-1}}\alpha_{\mathbf{1}}^{\mathbf{i}}}{\partial^{\mathbf{k+q-1}}\mathbf{y}_{\mathbf{n_{y'}}}^{\mathbf{i}}} \\ \frac{\partial \tan_{\Delta \mathbf{t}}{}^{\mathbf{k+q-1}}\alpha_{\mathbf{2}}^{\mathbf{i}}}{\partial^{\mathbf{k+q-1}}\mathbf{y}_{\mathbf{1}}^{\mathbf{i}}} & \frac{\partial \tan_{\Delta \mathbf{t}}{}^{\mathbf{k+q-1}}\alpha_{\mathbf{2}}^{\mathbf{i}}}{\partial^{\mathbf{k+q-1}}\mathbf{y}_{\mathbf{2}}^{\mathbf{i}}} & \cdots & \frac{\partial \tan_{\Delta \mathbf{t}}{}^{\mathbf{k+q-1}}\alpha_{\mathbf{2}}^{\mathbf{i}}}{\partial^{\mathbf{k+q-1}}\mathbf{y}_{\mathbf{n_{y'}}}^{\mathbf{i}}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \tan_{\Delta \mathbf{t}}{}^{\mathbf{k+q-1}}\alpha_{\mathbf{n_{y'}}}^{\mathbf{i}}}{\partial^{\mathbf{k+q-1}}\mathbf{y}_{\mathbf{1}}^{\mathbf{i}}} & \frac{\partial \tan_{\Delta \mathbf{t}}{}^{\mathbf{k+q-1}}\alpha_{\mathbf{n_{y'}}}^{\mathbf{i}}}{\partial^{\mathbf{k+q-1}}\mathbf{y}_{\mathbf{2}}^{\mathbf{i}}} & \cdots & \frac{\partial \tan_{\Delta \mathbf{t}}{}^{\mathbf{k+q-1}}\alpha_{\mathbf{n_{y'}}}^{\mathbf{i}}}{\partial^{\mathbf{k+q-1}}\mathbf{y}_{\mathbf{n_{y'}}}^{\mathbf{i}}} \end{bmatrix}_{\mathbf{n_{y'}\ x\ n_{y'}}} \qquad (19.c)$$

$$\frac{\partial\,{}^{k+1}\mathbf{y}^{i}}{\partial\,{}^{k}\mathbf{u}} = \Delta\mathbf{t}\cdot\begin{bmatrix} \dfrac{\partial\,\mathbf{tan}_{\Delta\mathbf{t}}{}^{\mathbf{k}}\alpha_{\mathbf{1}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{1}}} & \dfrac{\partial\,\mathbf{tan}_{\Delta\mathbf{t}}{}^{\mathbf{k}}\alpha_{\mathbf{1}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{2}}} & \cdots & \dfrac{\partial\,\mathbf{tan}_{\Delta\mathbf{t}}{}^{\mathbf{k}}\alpha_{\mathbf{1}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{n_{u'}}}} \\ \dfrac{\partial\,\mathbf{tan}_{\Delta\mathbf{t}}{}^{\mathbf{k}}\alpha_{\mathbf{2}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{1}}} & \dfrac{\partial\,\mathbf{tan}_{\Delta\mathbf{t}}{}^{\mathbf{k}}\alpha_{\mathbf{2}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{2}}} & \cdots & \dfrac{\partial\,\mathbf{tan}_{\Delta\mathbf{t}}{}^{\mathbf{k}}\alpha_{\mathbf{2}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{n_{u'}}}} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial\,\mathbf{tan}_{\Delta\mathbf{t}}{}^{\mathbf{k}}\alpha_{\mathbf{n_{y'}}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{1}}} & \dfrac{\partial\,\mathbf{tan}_{\Delta\mathbf{t}}{}^{\mathbf{k}}\alpha_{\mathbf{n_{y'}}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{2}}} & \cdots & \dfrac{\partial\,\mathbf{tan}_{\Delta\mathbf{t}}{}^{\mathbf{k}}\alpha_{\mathbf{n_{y'}}}^{\mathbf{i}}}{\partial^{\mathbf{k}}\mathbf{u}_{\mathbf{n_{u'}}}} \end{bmatrix}_{\mathbf{n_{y'}\ x\ n_{u'}}} \tag{19.d}$$

Para se determinar as matrizes (19.b), (19.c) e (19.d) recorre-se as equações de (16.a) à (16.c) do Capítulo 4 referentes a retro-propagação.