Calôba, L. P., “Redes Neurais em Modelagem de Sistemas” In: Enciclopédia de Automática.1 ed.São Paulo : Edgar Blucher, 2007, v.3, p. 325-344.

Redes Neurais em Modelagem de Sistemas.

Luiz P. Calôba

COPPE & EP – UFRJ

(caloba@ufrj.br)

1 - Introdução

Modelos matemáticos de sistemas físicos são ferramentas essenciais para um imenso número de áreas do conhecimento, inclusive para a automática. Modelos lineares são bastante bem estudados e conhecidos, mas infelizmente a representação precisa da maioria dos fenômenos da natureza inclui não linearidades. O objetivo deste capítulo é mostrar como redes neurais podem ser utilizadas para modelar com precisão sistemas não lineares estáticos ou dinâmicos. Supomos que o leitor já dispõe de algum conhecimento básico sobre modelagem de sistemas e redes neurais, uma vez que farta bibliografia existe sobre estes assuntos. Na sessão II apresentamos a rede neural *feedforward* que será usada ao longo deste trabalho, sua aplicação em modelagem e comentamos alguns detalhes de seu treinamento e operação. Na seção III apresentamos o uso de técnicas neurais para analisar modelos fenomenológicos e introduzimos os sub-modelos híbridos neural-fenomenológicos. Na seção IV apresentamos brevemente alguns aspectos de sistemas dinâmicos e introduzimos a estrutura NARMA, que será usada para modelá-los, discutindo detalhes desta modelagem e de seu treinamento. A seção IV apresentamos as conclusões e algumas referências bibliográficas.

II. Redes Neurais e Modelagem

Redes Neurais são algoritmos que tentam emular de uma forma muito simplificada a maneira como o cérebro animal processa determinadas informações. São baseadas em processadores elementares chamados neurônios, Fig. II.1,

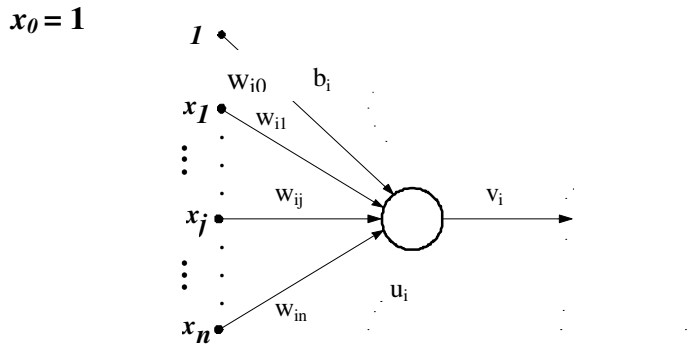


Fig. II.1 – i-ésimo neurônio.

definidos pelas suas funções de excitação u = ψ (x) e ativação v = ϕ (u). Usualmente empregamos:

$$u_i = \sum_{j=0}^{n} w_{ij}\, x_j \qquad x_0 = 1$$

$$v_i = \begin{cases} u_i & \text{neurônio linear} \\ \text{tgh}\, u_i & \text{neurônio tipo tgh} \end{cases}$$

onde x_j são as entradas, w_{ij} são os pesos sinápticos, ou sinapses, que conectam cada entrada x_j ao neurônio i, u_i é a excitação interna do neurônio e v_i a saída do mesmo. A polarização do neurônio é dada por uma sinapse w_{i0} conectada a uma entrada fixa $x_0 = 1$.

II.1 – Estrutura da Rede Neural

As redes neurais que utilizaremos neste trabalho são tipo *feedforward* com duas camadas de neurônios, com múltiplas entradas $z_1, \cdots, z_E$ e uma saída $\tilde{y}$, como apresentado na Fig. II.2. Todos os neurônios tem função de excitação ψ(.) do tipo apresentado na sessão anterior. A camada intermediária tem Q neurônios com função de ativação ϕ(.) = tgh(.) e a camada de saída tem um único neurônio linear. Cada entrada z_j da rede se comunica através de sinapses w_{ij} com todos os neurônios i da camada intermediária, e a saída de cada neurônio i da camada intermediária se comunica através de sinapses t_i com o neurônio de saída, na segunda camada.

Todos os neurônios têm sinapse de polarização, w_{i0} na camada intermediária e t_0 na camada de saída.

Por simplicidade consideraremos redes com apenas uma saída, mas os resultados podem ser facilmente estendidos para redes com múltiplas saídas.

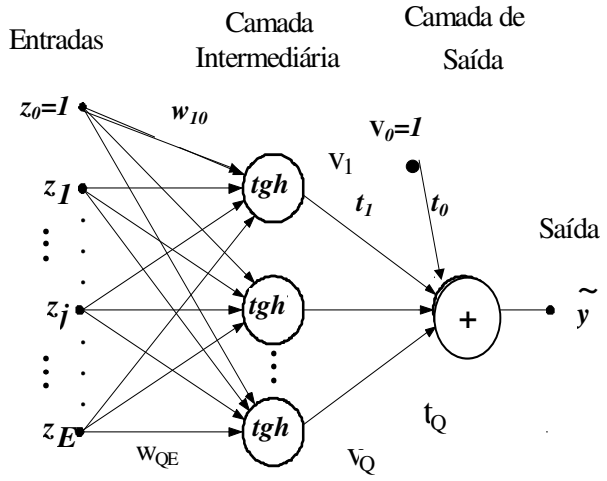


Fig. II.2 – Rede neural *feedforward* básica usada neste trabalho.

A propagação do sinal é dada por

$$v_i = \mathrm{tgh} \sum_{j=0}^{E} w_{ij}\, z_j \qquad \forall i = 1, \cdots, Q \; ; \; z_0 = 1$$

$$\tilde{y} = \sum_{j=0}^{Q} v_j\, t_j \qquad v_0 = 1$$

É possível demonstrar que para um Q suficientemente grande, mas finito, esta rede é um aproximador universal para $\tilde{y} = \varphi(\underline{z})$, desde que φ(.) seja L_2 (e.g., φ(.) admita transformada de Fourier), o que é o caso de praticamente todas as funções de interesse da engenharia.

II.2 – Treinamento

O treinamento da rede é o processo de busca do conjunto de sinapses (w_{ij}, t_j) que minimiza alguma função objetivo F. Usualmente F é escolhida como o erro médio quadrático na saída da rede

$$F = \underset{\forall p}{E}\left(\varepsilon_p^2\right) = \frac{1}{P}\sum_{p=1}^{P}\varepsilon_p^2$$

$$\varepsilon_p = y_p - \tilde{y}_p$$

onde p =1, ... , P são os pares entrada-saída $\left(\underline{z}_p, y_p\right)$ e y_p *e* $\tilde{y}_p$ são respectivamente as saídas desejada e estimada pela rede para o par p. Outras funções objetivo, e.g. erro relativo médio quadrático, erro de ordem mais elevada, etc., podem também ser usadas se necessário.

O treinamento usual é por épocas. Para a rede da Fig. 2 e $F = \underset{\forall p}{E}\left(\varepsilon_p^2\right)$ os acréscimos nas sinapses calculados para descida contra o gradiente (ou retropropagação do erro, regra delta) para cada par entrada-saída $(\underline{z}_p,\ y_p)$ usando passo de treinamento α são dados por:

$$\varepsilon_p = y_p - \tilde{y}_p$$

$$\Delta t_i = 2\alpha\varepsilon v_i \qquad \forall i = 1,\cdots,Q \ ; \ v_0 = 1$$

$$\Delta w_{ij} = 2\alpha\varepsilon t_i\left(1 - v_i^2\right)z_j \qquad \forall i = 1,\cdots,Q \ ; \ \forall j = 0,\cdots,E \ ; \ z_0 = 1$$

E o acréscimo a ser aplicado após cada época é o valor médio dos acréscimos calculados para cada par entrada-saída pelas equações acima.

II.3 – Detalhes do Treinamento de Redes Neurais

No treinamento da rede neural todos os detalhes convencionais devem ser observados, e.g.: (a) entradas e saídas devem ser normalizadas para média nula e desvio padrão da ordem de 0,5 , (b) as sinapses devem ser inicializadas com valores pequenos e randômicos, tipicamente

$|w_{ij}| < 0{,}2$, (c) no caso do *backpropagation* simples, o passo de treinamento deve ser pequeno, tipicamente $\alpha = 0{,}05$, (d) normalmente é necessário controlar o *overtraining* e obter um teste final independente do aprendizado, o que implica em utilizar três conjuntos de pares entrada-saída: o de treinamento, o de validação e o de teste. Cada conjunto deve representar estatisticamente bem a relação entrada-saída da rede. Para o controle do *overtraining* é necessário fazer o acompanhamento do erro dos conjuntos de treinamento, validação e teste durante o treinamento. O controle do *overtraing* é também o que permite, de modo simples, determinar o número de neurônios adequado na camada intermediária: é aquele que minimiza o erro nos conjuntos de teste e validação, (e) é prudente confirmar a validade dos resultados por validação cruzada, etc.

Mais detalhes específicos do treinamento para modelos de redes dinâmicas serão comentados posteriormente, na respectiva seção.

II.4 – Sistemas com Não Linearidades Fracas

Um grande número de plantas reais pode ser representado por um modelo linear sem um erro muito grande. Isto permite imaginá-los como sistemas lineares em que uma não linearidade não muito forte foi introduzida. Certamente podemos ter interesse em criar um modelo em que as partes linear e não linear que o compõem estejam separadas, principalmente no caso de não linearidades fracas. Isto pode ser conseguido transformando o primeiro neurônio da camada intermediária em um neurônio linear, que passara a ser responsável pela parte linear do mapeamento entrada-saída, enquanto os demais Q – 1 neurônios tipo tgh(.) realizarão a parte não linear. Neste caso, sem perda de generalidade podemos fixar os valores de $w_{10} = 0$ e $t_1 = 1$. Os sistemas linear e não linear serão então independentes, acoplados apenas pelo neurônio de saída, que soma as partes linear e não linear do mapeamento. Obviamente, as equações de operação e treinamento apresentadas anteriormente devem ser convenientemente modificadas. Mais ainda, para que a parte linear do mapeamento seja ótima, é necessário realizar o treinamento em duas fases. Na primeira fase as sinapses responsáveis pela parte linear do mapeamento, $w_{1j} \ \forall j = 1,...,E$, e t_0 serão treinadas. Para isto as sinapses conectadas às entradas e saídas dos neurônios não lineares da camada intermediaria são zeradas, $w_{ij} = 0 \ e \ t_i = 0 \ \ \forall i = 2,...,Q \ \ e \ \ \forall j = 0,...,E$, e as demais sinapses, $w_{1j} \ \forall j = 1,...,E$ e t_0, são então treinadas. Lembre que $w_{10} = 0$ e $t_1 = 1$ são fixas e não são treinadas. Após esta primeira fase do

treinamento as sinapses conectadas à entrada ou saída do neurônio linear da camada intermediária, $w_{1j}\ \forall j=1,...,E$ e t_1, são "congeladas", i.e. seus valores não mais se modificarão.

Na segunda fase as sinapses restantes, responsáveis pela parte não linear do mapeamento, $w_{ij}\ \forall i=2,...,Q\ e\ \forall j=0,...,E$ e $t_i\ \forall i=0,2,3,\ ...,E$ - inclusive t_0 , são treinadas. Para isto são inicializadas com valores aleatórios e pequenos, e o treinamento convencional é aplicado. Note que nesta segunda fase do treinamento, embora as sinapses conectadas ao primeiro neurônio estejam congeladas e não sejam alteradas, elas permanecem atuando na propagação do sinal através da rede.

III - Modelos Fenomenológicos e Sub-modelos Híbridos Neural-Fenomenológicos

Modelos neurais podem ser muito precisos, mas sendo numéricos têm contribuição limitada para o entendimento da fenomenologia do sistema representado, ao contrário dos modelos fenomenológicos. Nesta sessão será introduzida uma técnica que utiliza conceitos de redes neurais para avaliar e fornecer subsídios para aprimorar os sub-modelos fenomenológicos que compõem um modelo global. Será introduzido também um novo tipo de sub-modelo híbrido neural-fenomenológico que preserva o máximo possível o modelo fenomenológico e seus sub-modelos ao mesmo tempo em que fornece a precisão numérica dos modelos neurais.

III.1 Interpretação Neural dos Modelos Fenomenológicos

Modelos matemáticos fenomenológicos consistem usualmente de um sistema não recursivo de equações que são aplicadas sucessivamente, gerando cada qual uma variável interna ao sistema. Cada equação tem, geralmente, um significado fenomenológico, e pode ser considerada como um sub-modelo integrante do modelo. Representando cada equação como um bloco, ou sub-modelo, constrói-se o diagrama de blocos do modelo global. A Figura 3.1 ilustra um bloco ou sub-modelo e a Figura 3.2 representa o diagrama em blocos de um modelo global constituído por treze sub-modelos.

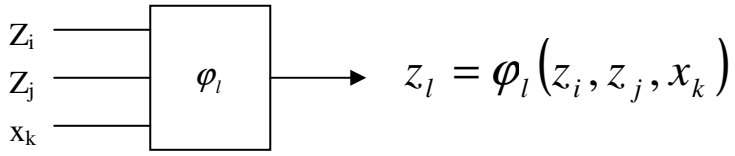


Figura 3.1: Bloco construtivo ou sub-modelo.

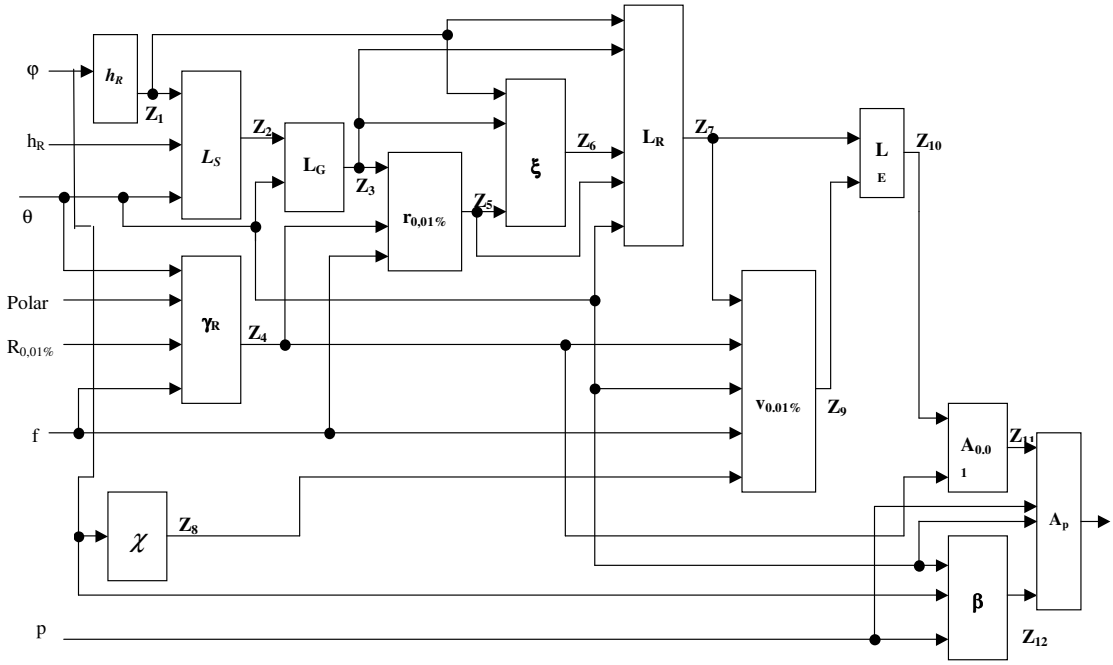


Figura 3.2: Diagrama em blocos de um modelo global.

Na fig. 3.2 φ, h_R, ... , p são as entradas, z_1, ..., z_{12} são as variáveis internas e $\tilde{y}$ é a saída estimada pelo modelo.

Representando as conexões entre os blocos da Figura 3.2 por arcos orientados, tem-se uma espécie de grafo *feedforward*. Interpretando os arcos como sinapses de ganho unitário e os blocos como neurônios com função de excitação/ativação igual as equações que os representam (φ_i), tem-se um modelo fenomenológico equivalente a uma rede neural não convencional do tipo *feedforward*.

Não abordaremos neste capítulo modelos que utilizam equações recursivas, mas eles podem ser representados por redes neurais do tipo NARMA não convencionais seguindo esta mesma linha.

III.2 Erro dos Blocos

A menos que sua equação represente com suficiente precisão o sub-modelo fenomenológico, cada bloco é responsável pela introdução de um erro que comporá o erro da saída do modelo global. É importante classificar os blocos ou sub-modelos função de sua importância para o erro na saída do modelo global. Considere a saída do bloco φ_i adicionada a uma sinapse c_i, conforme ilustrado na Figura 3.3, cuja função é corrigir o erro do bloco.

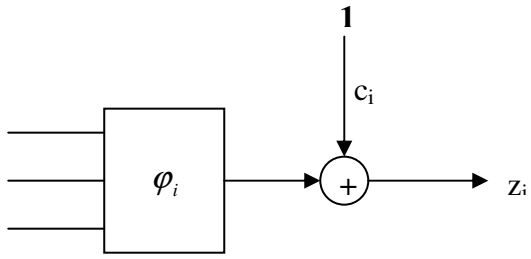


Figura 3.3: Bloco com sinapse de correção de erro.

Para que o diagrama de blocos da Figura 3.2 represente perfeitamente o sistema real, o valor nominal de c_i deve ser nulo para cada sub-modelo ou bloco, e para cada par entrada-saída. Nosso objetivo é medir a contribuição do erro de cada bloco ou sub-modelo na função objetivo F a ser minimizada, usualmente o erro médio quadrático na saída do modelo global.

$$F = \underset{\forall p}{E}(e_p) = \frac{1}{P}\sum_{p=1}^{P} e_p^2$$

$$e_p = y_p - \tilde{y}_p$$

onde E(.) é a função valor esperado para todo par entrada-saída p = 1,...,P, e_p é o erro na saída global para o par entrada-saída p, y_p é o valor desejado na saída e $\tilde{y}_p$ o valor estimado pelo modelo.

Considerando o modelo como uma rede neural não convencional, o valor de c_i pode ser ajustado para o par p pelo método do gradiente descendente aplicado à função e_p^2 em relação a um vetor $\underline{c}$ de componentes c_i. Assim,

$$\Delta c_i(p) = -\alpha \frac{\partial e_p^2}{\partial c_i} = -2\alpha e_p \frac{\partial e_p}{\partial \tilde{y}} \frac{\partial \tilde{y}}{\partial z_i} \frac{\partial z_i}{\partial c_i}$$

$$= 2\,\alpha \frac{\partial \tilde{y}}{\partial z_i} e_p = 2\alpha \delta_i$$

onde $\alpha > 0$, $\partial e_p / \partial \tilde{y} = -1$, $\partial z_i / \partial c_i = 1$ e δ_i é reconhecido como o erro retropropagado da saída da rede até a saída z_i do bloco φ_i.

Como se trata de uma correção por gradiente descendente, para uma aproximação de primeira ordem de F valida para $|\underline{\Delta c}|$ e α constantes e suficientemente pequenos, os valores de $\Delta c_i(p)$ calculados para cada bloco são os que minimizam a função objetivo. Por outro lado, quando $\Delta c_i(p)$ compensa otimamente o erro introduzido por cada bloco, o erro na saída e, conseqüentemente, a função F, são mínimos. Logo, o erro da saída retropropagado até a saída do bloco, δ_i, é um parâmetro bastante adequado para avaliar a importância do erro de cada bloco i na saída da rede.

O ajuste clássico das sinapses de uma rede neural, por batelada, é feito por um valor fixo correspondente a média dos valores calculados para cada par entrada-saída,

$$\Delta c_i = \underset{\forall p}{E}\left[\Delta c_i(p)\right]$$

A correção pode ser feita de maneira simples com a adição de uma polarização na saída do bloco. No entanto, este procedimento é pobre e corrige apenas o erro médio na saída do bloco, que muito provavelmente já é pequeno, se o sub-modelo foi razoavelmente projetado. Neste caso, é mais conveniente corrigir o erro $\Delta c_i(p)$, que varia com cada par p. Assim, uma

medida adequada para a influência do erro de cada bloco no erro da saída global do modelo é a variância de $\delta_i(p)$, σ_i^2, conforme expresso abaixo

$$\sigma_i^2 = \underset{\forall p}{E}\left[\delta_i^2(p)\right] - \underset{\forall p}{E}\left[\delta_i(p)\right]$$

Uma vez calculadas as influências dos erros de cada bloco, σ_i^2, no erro da saída global do modelo, são identificados os blocos críticos, isto é, aqueles com maior σ_i^2, maior influência. Estes blocos devem então ser modificados para melhorar a eficiência dos sub-modelos e, conseqüentemente, do modelo global.

III.3 Análise e Correção dos Sub-Modelos

Um sub-modelo pode ser inadequado por duas razões principais: a equação que o representa é inadequada ou as entradas do bloco são insuficientes para determinar sua saída, i.e., falta informação na entrada do bloco. Se o erro na saída de cada bloco crítico mantiver correlação significativa com qualquer uma de suas variáveis de entrada, isto significa que a equação do bloco ou sub-modelo não é adequada para representá-lo e, neste caso, necessita ser modificada. Se esta correlação não existe, provavelmente falta informação na entrada do bloco, que pode estar contida em outras variáveis. Quando a saída de um bloco é calculada estão disponíveis, além de suas variáveis de entrada, as variáveis de entrada da rede e outras variáveis internas do modelo global, saídas de outros blocos, já calculadas anteriormente. Se algumas destas variáveis apresentam correlação significativa com o erro na saída do bloco em questão, então elas podem ser utilizadas como entradas adicionais ao bloco para reduzir o erro na saída do mesmo. É possível que o erro na saída do bloco seja causado pelas duas razões citadas anteriormente. Neste caso, as variáveis candidatas a entradas adicionais devem ser descorrelacionadas das variáveis de entrada do bloco que apresentam correlação significativa com o erro em sua saída. Estas variáveis descorrelacionadas devem então ser novamente testadas para verificar se mantém correlação significativa com o erro na saída do bloco. Caso positivo, as variáveis originais, não descorrelacionadas, devem ser aceitas como novas entradas do bloco. Este último passo, envolvendo a descorrelação, não é essencial, mas evita a adição de novas

entradas que não trazem informação adicional ao bloco, e preserva o máximo possível os sub-modelos fenomenológicos.

O processo acima permite determinar os blocos críticos, que necessitam de correções, e eventuais variáveis adicionais que devem ser utilizadas no cálculo destas correções. A correção para manter o modelo como fenomenológico deve ser feita alterando a equação dos sub-modelos críticos, o que não é uma tarefa
fácil na maioria dos casos. Uma alternativa é utilizar um modelo híbrido fenomenológico-neural para o bloco. Uma rede neural é colocada em paralelo com cada bloco crítico, conforme ilustrado na Figura 3.4.

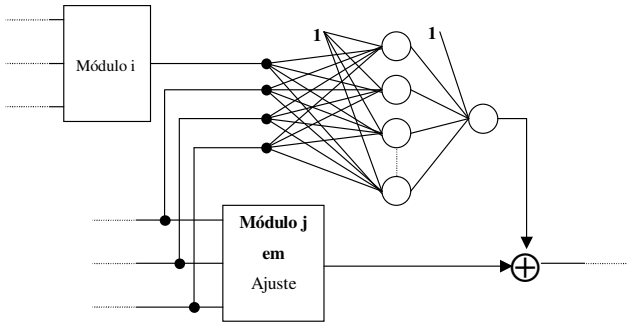


Figura 3.4: Conexão da rede neural com o módulo em processo de correção.

A rede é alimentada pelas entradas do bloco que tem correlação com o erro na saída, caso o erro seja devido à equação inadequada, pelas entradas adicionais que tem correlação com a saída, caso o erro seja devido à falta de informação na entrada ou por ambas, caso o erro tenha ambas as causas.

As redes neurais auxiliares devem ser treinadas simultaneamente, imersas no modelo fenomenológico. Se o treinamento é feito por retropropagação do erro da saída global do modelo, e_p, para cada par entrada-saída p, o erro retropropagado da saída do modelo até a saída da rede neural auxiliar do módulo i é determinado por:

$$\delta_i = e_p \frac{\partial \tilde{y}}{\partial z_i}$$

Daí para trás a retropropagação do erro dentro da rede ocorre de forma convencional. Como usualmente as entradas e saídas dos blocos não são normalizadas, é recomendável normalizar cada uma delas para que apresentem média nula e excursão na faixa (-1,+1), para garantir um treinamento numericamente robusto para as redes neurais corretoras dos blocos, evitando paralisia e problemas correlatos.

III.4 – Exemplo

O método descrito neste capítulo é genérico e pode ser aplicado em qualquer modelo fenomenológico. Alguns detalhes da aplicação no modelo UIT-R de atenuação em enlaces de comunicação terra-satélite são apresentados a seguir. Este modelo utiliza sete variáveis de entrada, uma de saída, e treze blocos ou sub-modelos com equações extremamente não lineares. Seu diagrama de blocos foi apresentado na fig. 3.2 como exemplo.

A tabela abaixo mostra a variância σ_i^2 do erro $\delta_i(p)$ em cada bloco i

Bloco	1	2	3	4	5	6	7	8	9	10	11	12
σ_i^2	0,17	0,07	0,09	0,57	1,09	0,00	0,13	0,01	0,97	0,16	0,13	0,49

A análise desta tabela mostra que os blocos mais críticos são os blocos 5 e 9, seguidos dos blocos 4 e 12. Estes resultados concordam com a informação fenomenológica disponível. Nossa opção foi utilizar redes neurais corretoras apenas nos dois blocos mais críticos, os blocos 5 e 9. A análise das correlações com os erros nas saídas dos blocos mostrou que a rede neural corretora do bloco 5 deveria ser alimentada com φ, h_R, θ, z_2 e z_3, e a do bloco 9 por φ, h_R, z_2, z_3, z_7 e z_8, mostrando que não apenas as equações dos blocos precisam ser revistas, mas também que mais informação parece ser necessária para estes sub-modelos.

O modelo UIT-R apresentou um erro de 32 dB, uma rede neural cuidadosamente projetada apresentou um erro de 20 dB, e o modelo híbrido apresentou um erro de 22 dB. Estes resultados mostram que os modelos neural e híbrido apresentam precisões consideravelmente superiores a do modelo fenomenológico, com uma ligeira vantagem do modelo neural sobre o modelo híbrido. Isto já era esperado porque o modelo híbrido ainda herda os erros contidos no modelo fenomenológico, enquanto que o modelo neural foi ajustado simplesmente com base nas medidas

não carrega os vícios contidos no modelo fenomenológico. Por outro lado, o modelo híbrido preserva a informação fenomenológica e a precisão numérica simultaneamente.

IV- Sistemas Dinâmicos

IV.1 – Sistemas Discretos no Tempo

São aqueles em que as variáveis são discretas no tempo ou, se contínuas, foram amostradas com período ΔT de modo a torná-las discretas. Para garantir a reconstrução do sinal contínuo original é necessário que a amostragem seja feita com uma freqüência $f_s = 1/\Delta T$ maior que duas vezes a máxima freqüência significativa envolvida no processo. Usualmente normalizamos $\Delta T = 1$ e $f_s = 1$.

IV.2 – Sistemas SISO

São sistemas com uma única entrada x(t) e uma única saída y(t), como na Fig. 4.1. São os sistemas que consideraremos neste trabalho, sem perda de generalidade: os resultados que apresentaremos podem com facilidade ser estendidos para sistemas com múltiplas entradas e saídas.

Fig. 4.1 – Sistema SISO

IV.3 – Sistemas Dinâmicos

Sistemas dinâmicos são aqueles em que a saída no instante t depende da entrada no instante t e em T instantes anteriores. Para $\Delta T = 1$,

$$y(t) = \varphi[x(t), x(t-1), \cdots, x(t-T)]$$

IV.4 – Modelos para Sistemas Dinâmicos

A equação anterior pode ser implementada pela estrutura da Fig. 4.2 abaixo, composta de uma cadeia de atrasos e um bloco que realiza φ(.)

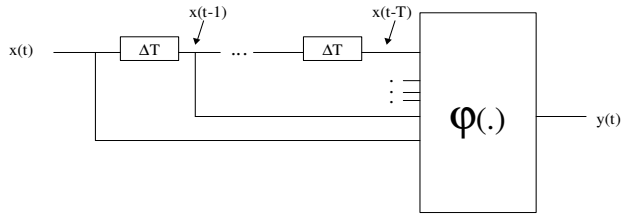


Fig. 4.2 – Modelo NMA

Se o sistema é linear a equação acima se torna

$$y(t) = \sum_{i=0}^{T} b_i \; x(t - i)$$

e o bloco φ(.) é um mero somador ponderado das variáveis independentes. A estrutura é conhecida como um filtro FIR, *Finite Impulse Response*, em processamento de sinais ou filtro média móvel, MA, em estatística.

Se o sistema é não linear o bloco φ(.) pode ser implementado por uma rede neural feedforward de duas camadas como a da Fig. II.2, e a estrutura é conhecida como filtro média móvel não linear, NMA.

A equação do sistema também pode ser escrita com a saída atual como função de entrada atual e de N saídas atrasadas.

$$y(t) = \varphi[x(t), y(t-1), \cdots, y(t-N)]$$

onde N é a ordem do sistema. A estrutura da Fig. 4.3 abaixo implementa a equação acima se substituirmos na equação os valores de y(.) por $\tilde{y}(.)$, isto é, em vez das saídas da planta usarmos a aproximação das mesmas calculadas pelo modelo.

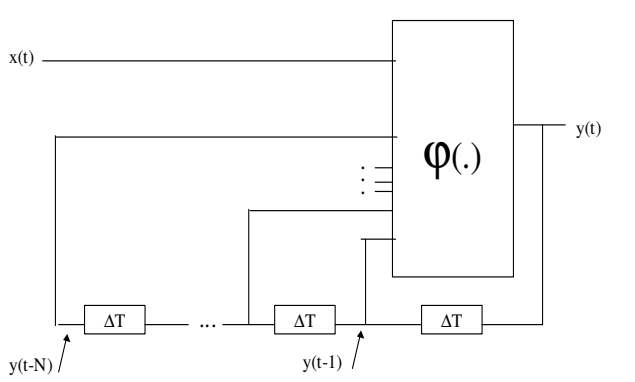


Fig. 4.3 – Modelo NAR

Se o sistema é linear a função φ(.) é um mero somador ponderado das variáveis independentes, e o modelo é conhecido em estatística como autoregressivo, AR. Se o sistema é não linear φ(.) pode ser implementado por uma rede neural, e o modelo é auto regressivo não linear, NAR.

Geralmente é mais eficiente associarmos os dois modelos, fazendo a saída função das entradas atual e atrasadas até M intervalos de tempo, e das saída atrasadas até N intervalos de tempo, onde usualmente $M \leq N$.

$$y(t) = \varphi[x(t), x(t-1), \cdots, x(t-M), y(t-1), \cdots, y(t-N)]$$

A equação acima pode ser implementada pela estrutura da Fig. 4.4 abaixo se substituirmos na equação os valores de y(.) por $\tilde{y}(.)$, isto é, em vez das saídas da planta usarmos a aproximação das mesmas calculadas pelo modelo, como no caso anterior.

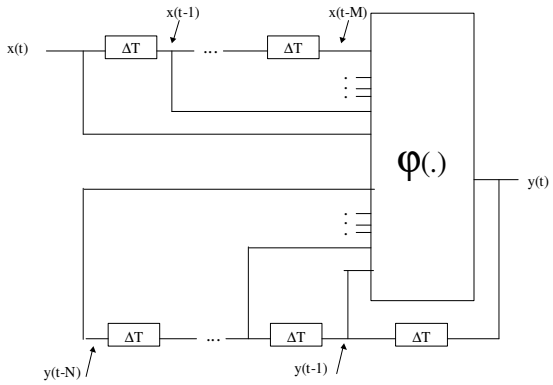


Fig. 4.4 – Modelo NARMA

Novamente, se o sistema é linear φ(.) é um simples ponderador e a estrutura é conhecida como um filtro IIR, *Infinite Impulse Response*, em processamento de sinais, e filtro ARMA em estatística. Se o sistema é não linear o bloco φ(.) pode ser implementado por uma rede neural e o modelo recebe o nome de ARMA não linear, NARMA.

O modelo NARMA (e suas simplificações NAR e NMA) é o mais utilizado em conjunto com redes neurais, talvez porque pode ser facilmente reinicializado em qualquer instante de tempo, porque seu estado é composto por variáveis do sistema facilmente observáveis, suas entradas e saídas atrasadas.

Existem diversos outros modelos com estados não facilmente observáveis, mas não os consideraremos neste trabalho.

IV.5 – Dimensão dos Modelos

Um sistema dinâmico real estável tem freqüências naturais $s_i = \sigma_i + jw_i$ com respectivas constantes de tempo $\tau_i = -1/\sigma_i$. Neste trabalho, no caso de sistemas não lineares, chamaremos "freqüências naturais" as freqüências naturais do sistema linearizado nos diversos pontos de operação. O mesmo vale para "singularidades", "constante de tempo", etc. Do ponto de vista prático a saída é independente de entradas atrasadas mais do que quatro vezes a maior constante de tempo do sistema. Isto nos fornece um limite inferior para número de atrasos T no filtro MA ou NMA,

$$T \cdot \Delta T \geq 4 \, \mathrm{Max}\,(\tau_i)$$

Como usualmente as baixas freqüências do sistema são reais,

$$T \cdot \Delta T \geq \frac{2}{\pi f_{min}}$$

onde $f_{min} = 1 / (2\pi \, \mathrm{Max}(\tau_i))$ é a mais baixa freqüência natural do sistema, em Hz.

Por outro lado, a freqüência de amostragem f_s deve ser maior que o dobro da maior freqüência operada pelo sistema, f_{max}. Como o sistema é não linear, é razoável considerar esta freqüência f_{max} como pelo menos a maior freqüência natural do sistema ou se for o caso (não provável) alguma freqüência ainda maior inserida na entrada.

$$f_s = \frac{1}{\Delta T} \geq 2 f_{max}$$

Das duas inequações anteriores verificamos que o número de atrasos T necessários no filtro MA é aproximadamente

$$T \cong \frac{4}{\pi} \frac{f_{max}}{f_{min}}$$

que pode ser razoavelmente grande na prática. Nos modelos ARMA e NARMA, N é a ordem estimada da planta, e usualmente $M \leq N$. Geralmente estes modelos são de dimensão muito menor que os MA e NMA, e por isto são mais utilizados na prática.

IV.6 – Estabilidade e Precisão dos Modelos

O modelo NMA apresentado na Fig. 4.2 não inclui realimentação e é portanto estruturalmente estável, e mantém o erro entre a saída da planta e a saída do modelo $\varepsilon(t) = y(t) - \tilde{y}(t)$ com módulo aproximadamente da mesma ordem de grandeza a medida que o tempo evolui.

O modelo NARMA apresentado na Fig. 4.4 inclui realimentação e é potencialmente instável. Praticamente, se a função $\varphi(.)$ não for determinada muito precisamente a saída $\tilde{y}(t)$ do modelo diverge da da planta y(t) após um certo número de passos de operação.

Isto limita a duração da simulação, e nos obriga a reinicializar a rede após um período de tempo, e retomar a simulação a partir deste ponto. Note que embora inconveniente, no caso do modelo NARMA isto é simples de realizar, porque seu estado real é facilmente determinado a cada instante.

IV.7 – Preditores e Operações em Paralelo e em Série-Paralelo

Há casos, e.g. em controle preditivo ou em previsão de séries temporais, em que desejamos que a saída do modelo no instante atual t aproxime a saída do sistema em um instante futuro $t + k$, $k > 0$. Neste caso a equação do modelo NARMA pode ser escrita:

$$\tilde{y}(t + k) = \varphi\left[x(t), \cdots, x(t - M), y(t), \cdots, y(t - N)\right]$$

e implementada por uma estrutura análoga à da Fig. 4.4, com os valores de y(.) substituídos pelos de $\tilde{y}(.)$ (operação com planta e modelo em paralelo) ou pela estrutura da Fig. 4.5, (operação com planta e modelo em uma estrutura série-paralelo).

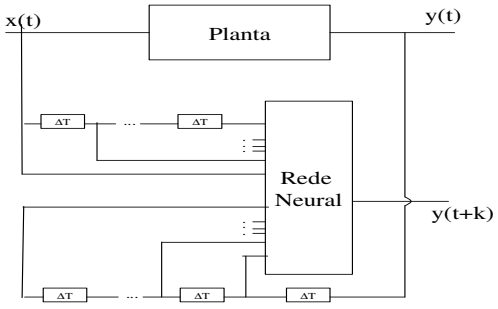


Fig. 4.5 - Operação em configuração série paralelo.

Na operação em paralelo a informação das saídas atrasadas é fornecida à rede neural pela aproximação feita pela própria rede, eventualmente – provavelmente - realimentando o erro e provocando perda da precisão ao longo do tempo, e possivelmente instabilidade. A vantagem é que o modelo é completamente independente da planta. Este é o modo de operação adequado para simulações, mas é necessário que os estados – saídas atrasadas – sejam corrigidos periodicamente.

Na operação em série paralelo a informação dos valores de saída atrasados é fornecida à rede pela própria planta, eliminando a realimentação do erro. A estrutura da Fig. 4.5 não é realimentada, logo é estruturalmente estável. A desvantagem é que exige as informações da planta para operação, limitando sua utilização em simulações. Este é o modo de operação a ser usado junto à planta, na implementação do sistema de controle.

IV.8 – Detalhes Específicos do Treinamento de Modelos Neurais para Sistemas Dinâmicos.

Uma das vantagens do modelo NARMA é que as partes dinâmica e não linear podem ser facilmente separadas. Com isto a rede pode ser treinada fora do modelo, de forma estática. Os pares entrada-saída para o treinamento são obtidos a partir de uma tabela com três colunas, t, x(t) e y(t) que lista a evolução das variáveis de entrada e saída do sistema ao longo do tempo. Cada instante de tempo t maior que N podendo gerar um par entrada-saída (e, s):

$\underline{e} = [x(t), x(t-1), ..., x(t-M), y(t-1), ..., y(t-N)]^{t}$

$s = y(t)$

cujos valores numéricos são tirados da tabela com 3 colunas, t, x(t) e y(t).

IV.8.1 – Seleção de Pares Entrada-Saída

Uma planta industrial pode permanecer longo tempo no entorno do *set point*, e pouco tempo transicionando de um *set point* para outro. Este processo gerará um grande número de pares entrada-saída no entorno dos *set points* e poucos nas transições, fazendo com que estas não sejam corretamente aprendidas. Usualmente é necessário equilibrar as populações de pares nas diversas regiões ao construir os conjuntos de treinamento e validação. Regiões mal treinadas podem ser detectadas após o treinamento devido aos erros anormalmente grandes que apresentam. Novos pares entrada-saída deverão então ser criados nestas regiões para reforçar a importância dos erros nas mesmas, ou os erros retropropagados ponderados por região.

IV.8.2 – Univocidade

Um outro detalhe importante a considerar é que o mapeamento entrada-saída a ser aprendido pela rede neural deve obrigatoriamente ser unívoco, o que pode não ser o caso para o modelo inverso de algumas plantas.

IV.8.3 – Ordem da Planta e do Modelo

É necessário ter uma idéia à priori da ordem e do tipo da planta para arbitrar N e M. O conhecimento da fenomenologia da planta, a autocorrelação da saída e os correlogramas entre as entradas e saídas podem dar uma idéia desta ordem. Se escolhermos N (e M) pequeno a rede não consegue aprender o mapeamento, e se escolhermos N grande "freqüências naturais" usualmente altas e instáveis serão criadas artificialmente. Se não temos uma idéia muito precisa da ordem, uma alternativa é utilizar N e M um pouco maior que o previsto. Após o treinamento examina-se a relevância de cada entrada da rede neural, especialmente aquelas que correspondem aos

maiores atrasos. Eliminam-se as entradas de baixa relevância, se houver, e refaz-se o treinamento usando apenas com as entradas mais relevantes.

IV.8.4- Singularidades em Altas Freqüências

Plantas reais são normalmente operadas em *set points* escolhidos para maximizar a eficácia no processo, o que geralmente implica em trabalhar em regiões não lineares e com tempo de transição da ordem das maiores "constantes de tempo", para evitar *ringing* ou *overshoot* fortes. Isto é, trabalham com grandes amplitudes de sinal e faixa de freqüências limitada.

Plantas reais podem ter "singularidades" em baixas e altas freqüências, e não linearidades. Os pares entrada-saída devem explorar todas estas características para que o modelo neural aprenda a generalizar a planta verdadeira. As não linearidades e a região de baixas freqüências são normalmente bem exploradas pelos sinais de operação da planta, mas isto muitas vezes não ocorre com a região de altas freqüências.

A solução normalmente permitida é adicionar ao sinal de entrada da planta um pequeno ruído r(t) contendo altas freqüências, usualmente uma onda quadrada com pequena amplitude e *duty cycle* de duração aleatória, que tem um espectro de freqüência mais ou menos constante. Entretanto, como este sinal é muito pequeno comparado ao sinal de entrada, quase não afeta o sinal e o erro na saída da rede, e quase não é percebido pelo algoritmo de treinamento *backpropagation*.

Uma possibilidade é destacar as altas freqüências do sinal de erro. O sinal de erro $\varepsilon(t)$ é separado em dois por filtros com bandas passantes diferentes, o da faixa de baixas freqüência, $\varepsilon_{LF}(t)$, cujas componentes estão na faixa de freqüência do sinal de operação normal da planta, e o da faixa de altas freqüências, $\varepsilon_{HF}(t)$, gerado praticamente pelas altas freqüências componentes do ruído r(t) injetado na planta junto com o sinal de entrada. Um novo sinal de erro $\varepsilon_M(t)$, onde $\varepsilon_{HF}(t)$ é relevante, é gerado e utilizado no algoritmo de treinamento backpropagation, Fig. 4.6.

$$\varepsilon_M(t) = \varepsilon_{LF}(t) + k\, \varepsilon_{HF}(t)$$

onde k ≥ 1 inicia grande e é reduzido até 1 ao longo do treinamento.

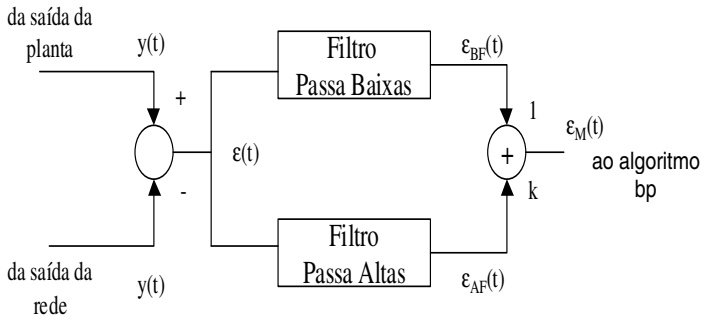


Fig. 4.6 – Novo sinal de erro

Uma desvantagem deste processo é que os pares entrada-saída devem estar ordenados no tempo, ou os estados internos dos filtros devem ser atualizados antes da aplicação de cada par entrada-saída, o que pode ser trabalhoso.

IV.9 – Exemplos

IV.9.1 – Exemplo 1

Considere uma base retangular apoiada em quatro molas e com uma massa desbalanceada sobre ela, Fig. 4.7.

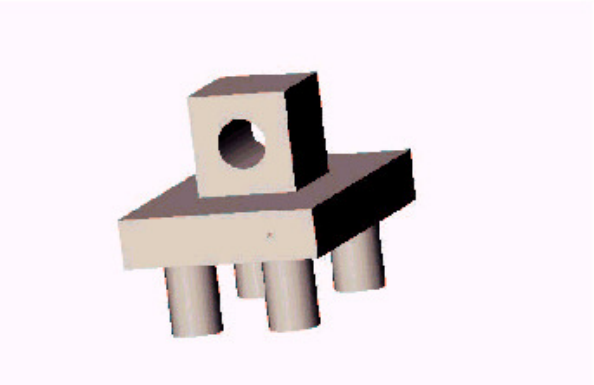
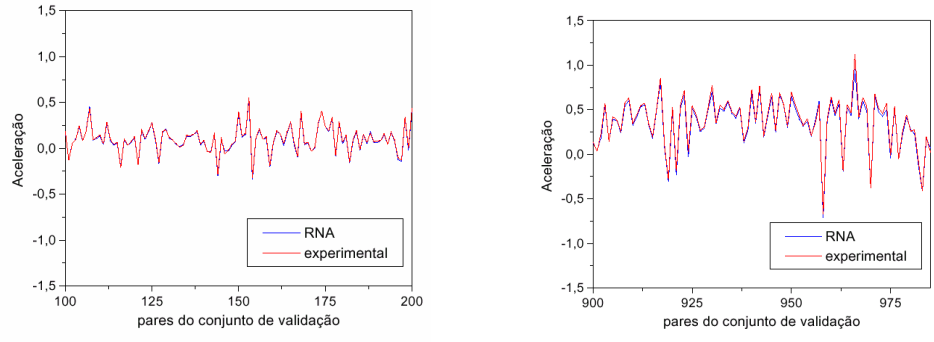


Fig. 4.7 – Experimento.

Como excitação utilizou-se um martelete de impacto que gera uma entrada x(t) muito breve, praticamente um impulso em nossa escala de tempo, $x(t) = \delta(t)$. A saída y(t) é a resposta de um acelerômetro acoplado à placa.

Após o instante inicial $t = 0$ a entrada é nula, $x(t) = 0$, e o nosso modelo NARMA reduz-se ao modelo NAR da Fig. 4.3 em que até mesmo a entrada x(t) é eliminada, porque nula para $t>0$. Após alguns experimentos escolhemos $N = 5$ e uma rede neural com $Q = 3$ neurônios na camada intermediária. A rede foi então treinada e testada com os sinais obtidos diretamente do experimento.

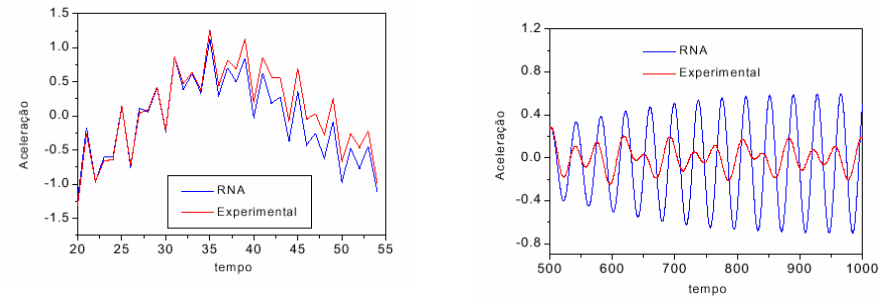
Montada no modo série paralelo, em que as saídas atrasadas são fornecidas pela planta, o modelo representa muito bem a planta em todos os intervalos de tempo, como pode ser visto na Fig. 4.8



(a) (b)

Fig. 4.8 - Resposta da planta e do modelo operando em série paralelo para dois intervalos distintos de tempo, a) 100 < t < 200 e b) 900 < t < 1000

Montado em modo paralelo, em que o modelo se auto-realimenta, a precisão da modelagem somente é boa durante cerca dos 35 primeiros passos da operação, mostrando inclusive tendência à instabilidade para longos tempos de operação, conforme pode ser visto na Fig. 4.9.



(a) (b)

Fig. 4.9 - Resposta da planta e do modelo operando em modo paralelo para dois intervalos distintos de tempo a) 20 < t < 55 e b) 500 < t < 1000

Este exemplo destaca o maior problema de uma modelagem do sistema com uma aproximação apenas boa da não linearidade, realizada com dados obtidos diretamente do experimento. O modelo diverge da planta após alguns passos de simulação, sendo necessário atualizar os estados de tempos em tempos. O que não lhe retira a utilidade.

IV.9.2 –Exemplo 2

Considere um sistema dinâmico regido pela seguinte equação a diferença.

$$y(t) = \frac{y(t-1)\,y(t-2)\,y(t-3)\,x(t-2)\,[y(t-3)-1] + x(t-1)}{1 + y(t-3)^2 + y(t-2)^3}$$

Para este exemplo foi utilizado o processo de destacar o erro em altas freqüências descrito na sessão IV.8.4. O sistema foi excitado por uma entrada x(t) composta por um simples sinal senoidal com freqüência baixa, dentro da banda da planta, e amplitude alta, cobrindo toda a faixa dinâmica da saída da planta. A esta senoide foi adicionado um pequeno ruído "branco": uma onda quadrada com amplitude da ordem de 2% da amplitude da senoide e *duty cycle* aleatório. Na saída da planta também foi adicionado um ruído branco com amplitude similar.

As condições acima são desconhecidas do modelista do sistema, e foram utilizadas apenas para gerar as tabelas numéricas com as três colunas, o tempo t, a seqüência de entrada x(t) e a seqüência de saída y(t). Desta tabela foram extraídos os pares entrada-saída dos conjuntos de treinamento e validação.

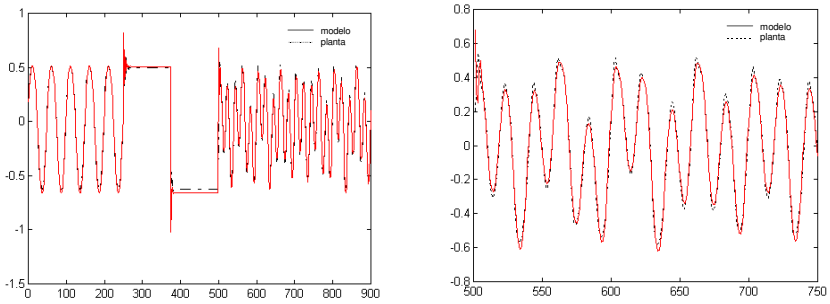
Após algumas observações e simulações iniciais arbitramos para o modelo NARMA que deveria simular o sistema M = 4, N = 5 atrasos e Q = 7 neurônios para a camada intermediária da rede neural.

O treinamento da rede neural foi bastante cuidadoso no tocante à precisão da aproximação. Após dois treinamentos e eliminação das entradas pouco relevantes foram encontrados os valores corretos de N = 3 e M = 2.

Para dificultar a tarefa de modelagem, para o conjunto de teste foi usado como entrada um sinal complexo:

$$x(k) = \begin{cases} \text{sen}(\pi k / 25) & , k < 250 \\ +1 & , 250 \le k < 375 \\ -1 & , 375 \le k < 500 \\ 0{,}3\,\text{sen}(\pi k / 25) + 0{,}1\,\text{sen}(\pi k / 32) + 0{,}6\,\text{sen}(\pi k / 10) & , 500 \le k < 900 \end{cases}$$

A Fig. 4.10 mostra a resposta da planta e do modelo operando em modo paralelo, isto é, auto-realimentado. O modelo preserva a estabilidade e precisão mesmo após 900 passos de operação, devido à boa qualidade da aproximação provida pela rede neural.



(a) (b)

Fig. 4.10 - (a) Reposta da planta e do modelo.
(b) Detalhe dos últimos passos da operação

É interessante notar que as soluções não são obrigatoriamente únicas. Repetindo o processo de treinamento encontramos M = 1 e N = 2, com resposta surpreendentemente muito semelhante a da Fig. 4.10. Estados correspondentes a freqüências mais altas podem, em alguns casos, ter pouca relevância na resposta na operação, como foi mencionado anteriormente.

V – Outros Trabalhos na Área:

Modelos baseados em redes neurais, ou simplesmente modelos neurais, podem ser uma importante ferramenta auxiliar na modelagem de sistemas não lineares estáticos ou dinâmicos.

Modelos neurais diretos podem ser bastante precisos numericamente. Mas se pretendemos precisão numérica e também preservar a fenomenologia do sistema, então podemos usar modelos e sub-modelos híbridos neural-fenomenológicos, que alem de boa precisão numérica preservam também suas características fenomenológicas, permitindo inclusive expandir os conhecimentos sobre a mesma.

No caso de sistemas dinâmicos, o modelo discutido neste trabalho é o NARMA, contendo uma rede neural. Operando em modo série-paralelo o NARMA modela fácil e precisamente sistemas dinâmicos não lineares independente do tempo de simulação, mas necessita das saídas passadas da planta para operar. Operando em modo paralelo o modelo se torna autônomo, independente da planta, mas costuma divergir da mesma depois de um certo número de passos de operação, quando é necessário atualizar seus estados para que a simulação possa prosseguir com precisão.

Como dito anteriormente existe farta bibliografia sobre redes neurais em geral, da qual destacamos dois livros, Haykin (1999), um dos trabalhos mais completos sobre o assunto, e Fawsett (1994) por sua clareza e aplicabilidade. Sobre modelagem destacamos Aguirre (2004), um livro recente e muito completo.

Especificamente sobre aplicação de redes neurais em modelagem e controle destacamos dois livros, Luo e Unbehauen (1998) e Nascimento e Yoneyama (2000). Algumas revistas publicaram números especiais sobre esta aplicacão, e.g. o IEEE Tr. on Neural Networks vol 5, # 2 (1994) e a Automática vol 37, #8 (2001), e o IEEE Tr. on Neural Networks publica desde 1999 a sessão "Control and Estimation". Um artigo seminal sobre o assunto é Narendra e Parthasarathy (1990).

Referências Bibliográficas:

Aguirre, L. A., "Introdução à Identificação de Sistemas", UFMG, 2004, Belo Horizonte.

Alencar, G. A., "Previsão de Atenuação por Chuvas em Enlaces Terra-Satélite Utilizando Redes Neurais Artificiais", Tese D.Sc., COPPE/UFRJ, PEE, 2003.

Calôba, L.P., "Introdução ao Uso de Redes Neurais na Modelagem de Sistemas Dinâmicos e Séries Temporais.", Livro de Minicursos do XIV Congresso Brasileiro de Automática, Natal, 2002.

Calôba, L.P., " Introdução à Computação Neuronal", Livro de Minicursos do IX Congresso Brasileiro de Automática, 1992.

Calôba, L.P.; Alencar, G.A.; Assis, M., “Hybrid Neural-Phenomenological Sub-Models and its Application to Earth-space Path Signal Attenuation Prediction“ Proc. Int. Joint Conf. on Neural Networks, Montreal, 2005.

Chen, L.; Narendra, K. S., “Identification and Control of a Nonlinear Discrete Time System based on its Linearization: An Unified Framework”, IEEE Transaction on Neural Networks vol 15, # 3, pp 663-673, May 2004.

Fawsett, L., “Fundamentals of Neural Networks”, Prentice Hall, 1994.

Haykin, S., “Neural Networks, A Comprehensive Foundation”, Ch. 13 - 15, Prentice Hall, 1999. Também em português, “Redes Neurais, Teoria e Prática”, Bookman, 2001.

Luo, F. L.; Unbehauen, R., “Applied Neural Networks for Signal Processing”, Ch. 8, Cambridge Univ. Press, 1998.

Monteiro, J.B., “Identificação de Sistemas Dinâmicos Não Lineares Usando Redes Neurais”, Tese M.Sc., COPPE-UFRJ, PEE, 1997.

Monteiro, J.B.; Calôba, L.P., “Redes Neurais: Identificação de Sistemas Dinâmicos Não-Lineares com Interferência Reduzida na Operação”, XII Congresso Brasileiro de Automática, Uberlândia, Set. 1998, 6 pg.

Narendra, K. S.; Parthasarathy, K., “Identification and Control of Dynamic Systems Using Neural Networks”, IEEE Trans. on Neural Networks, Vol. 1, no. 1, pg. 4-27, 1990.

Nascimento, C. L.; Yoneyama, T., “Inteligência Artificial em Controle e Automação”, Cap. 13, Ed. Edgard Blucher, São Paulo, 2000.

Silva, M.F.T., “Identificação de um Sistema Dinâmico através de Redes Neurais Artificiais”, Trabalho de fim de disciplina, EP-UFRJ, 2001.

“Special Issue on Dynamic Recurrent Neural Systems”, IEEE Trans. on Neural Networks, vol. 5, no. 2, March 94.

“Special Issue on Neural Networks Feedback Control”, Automatica, vol 37, #8, Aug 2001.