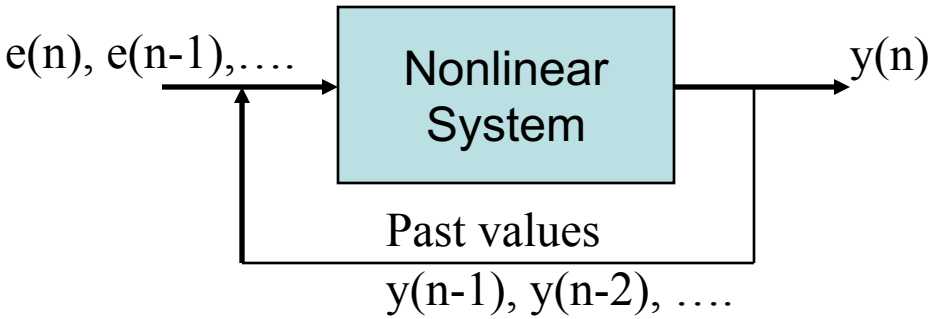
NEURAL NETWORKS FOR PREDICTION

Dr. Yogananda Isukapalli

The Prediction Problem

$y(n) = f(y(n-1), y(n-1), \ldots\ldots , e(n), e(n-1), e(n-2), \ldots\ldots..)$

Given the past values of y, y(n) can be predicted approximately



Thus prediction of y(n) approximates the nonlinear system. If f() is linear, then this becomes linear prediction, and has been extensively used in speech coding. But in practice, systems are nonlinear and thus a nonlinear predictor is expected to do better. Other aplications of times series prediction are in forecasting – such as weather forecasting, load forecasting etc.

If the input e(n) is white noise, then the e(n) component in y(n) cannot be predicted. It is also known as the innovations residual (the new information in y(n))

Conventional Nonlinear Predictors

(a) A Kalman filter based on State Dependent Model

The assumed nonlinear AR Model is,

$$x(n) = \phi_1(x_{n-1})x(n-1) + \phi_2(x_{n-1})x(n-2) + \ldots\ldots \phi_M(x_{n-1})x(n-M) + e(n)$$

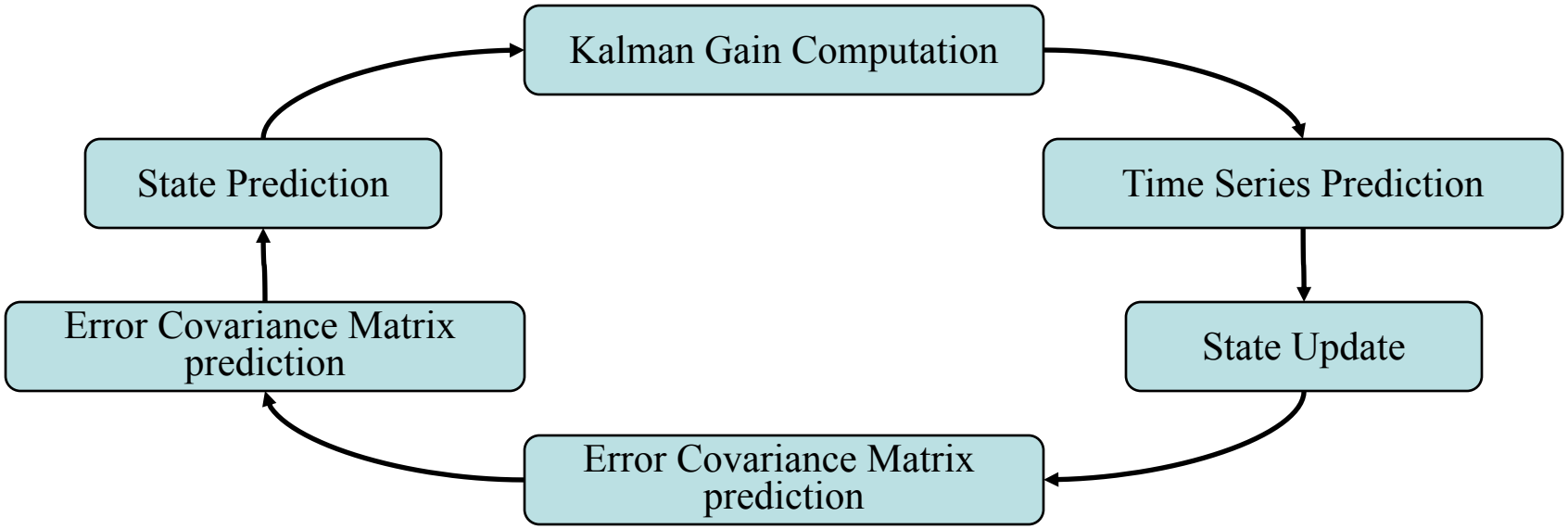
where x_{n-1} = [x(n-1) x(n-2) …. x(n-M)]

ϕ is the state vector containing the coefficients.

F_n is the equivalent of the state transition matrix.

H_n is the vector containing the previous observations.

The Kalman filter recursion consists of the following six steps:



M.B. Priestley, "Non-linear and non-stationary time series analysis", Academic Press, Chapter 4 & 5, 1988

Conventional Nonlinear Predictors

(b) Second order Volterra Filter (SVF) as a Predictor

Let x_{n-1} = { x(n), x(n-1), x(n-2), …. x(n-M) }

The Volterra series expansion of x(n+1) is given by,

$$x(n+1) = c + \sum L_i x(n-i) + \sum\sum Q_{ij} x(n-i)x(n-j) + \sum\sum\sum T_{ijk} x(n-i)x(n-j)x(n-k) + ...$$

The SVF is the truncated and finite order representation of this expansion.

$$x(n+1) = C + \mathbf{L}\mathbf{x}^t + \mathbf{x}^t\mathbf{Q}\mathbf{x}$$

where c is a constant, L is a 1 x M vector and Q is an M x M matrix.

The Volterra filter weights are updated as soon as a new sample becomes available

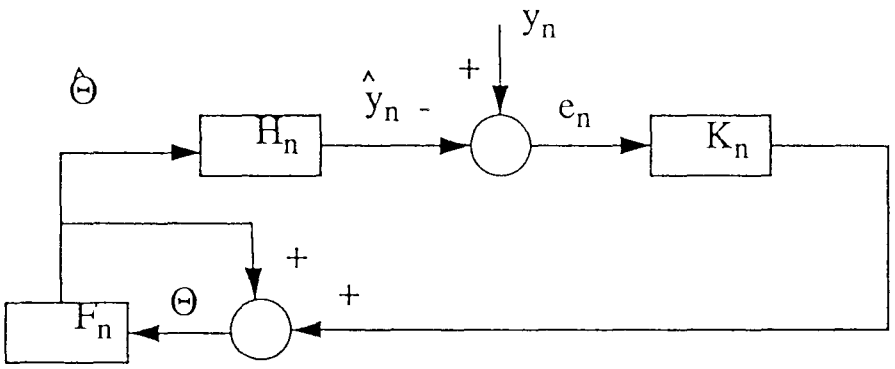
The updated is done using the LMS or fast kalman filter algorithm.

When the Q weights are set to zero, the SVF represents a simple linear filter.

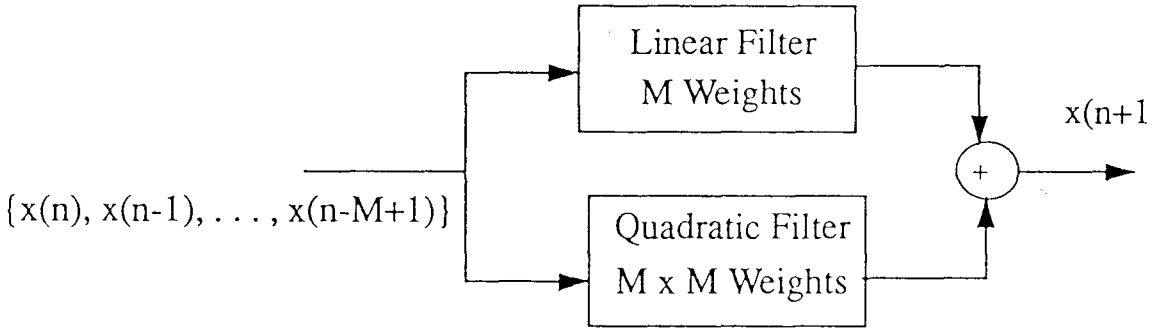
Requires knowledge of the model order of the time series.

C.E. Davila, et. al., "A second order adaptive volterra filter with rapid convergence", IEEE trans. On ASSP, September 1987.

Conventional Nonlinear Predictors



Schematic of the Kalman filter



Schematic of a Second Order Volterra Filter

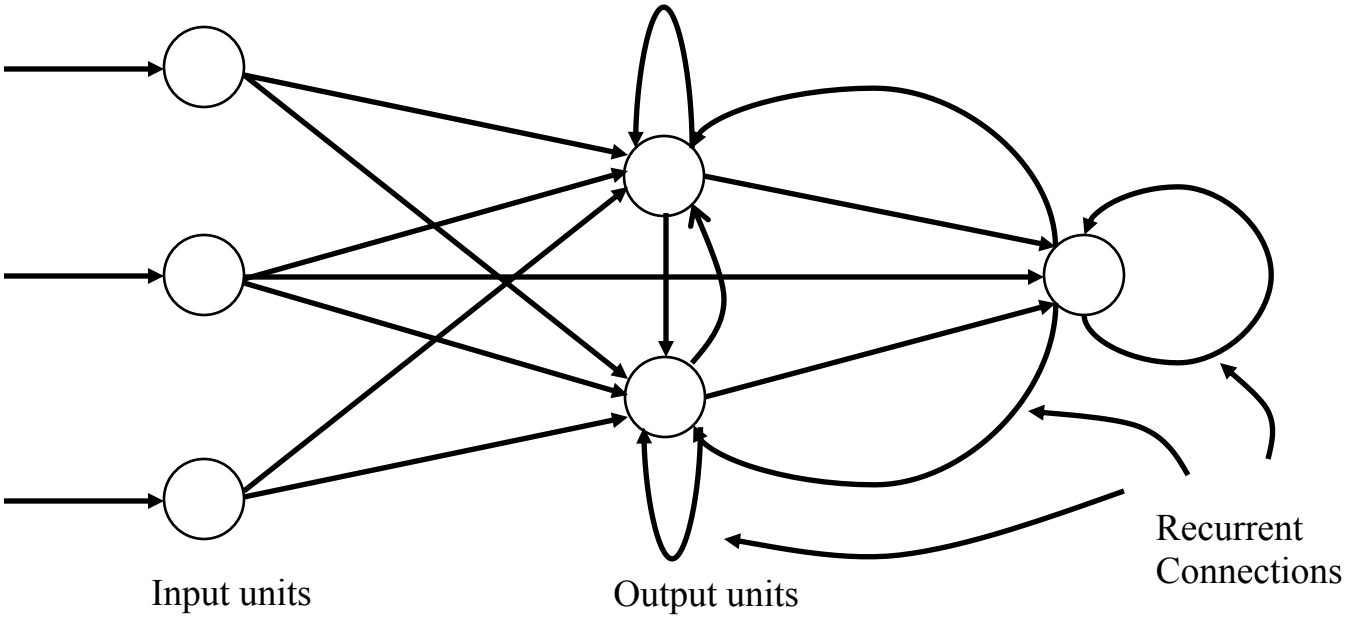
Fully connected Recurrent Neural Networks

Have both forward as well as feedback connections

Information from infinite past can be retained using the recurrent connections.

Trained using the Real Time Recurrent Learning (RTRL) algorithm.

Weight update is by gradient descent & is non-local.

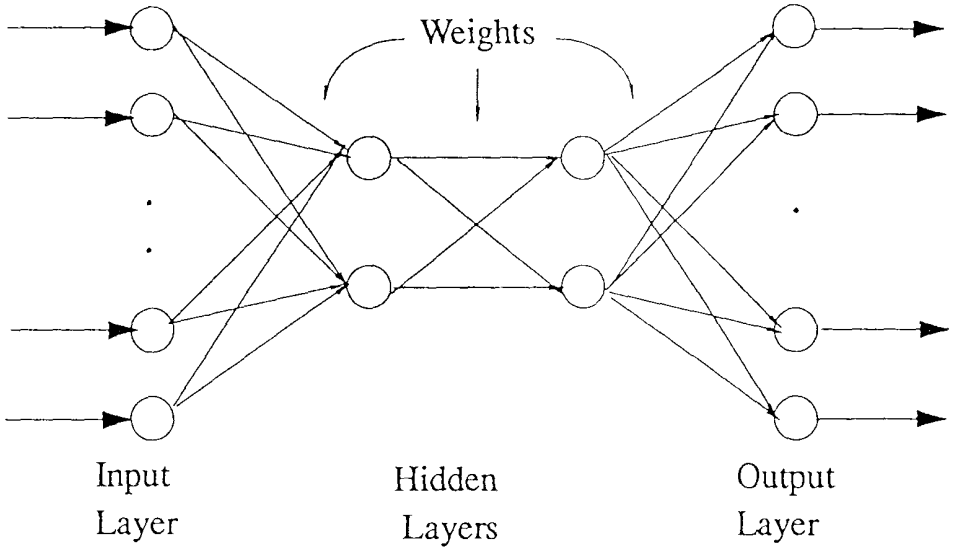


Fully connected Recurrent Neural Network

R.J. Williams & David Zipser, "A learning algorithm for continually running fully recurrent neural networks:, Neural Computation, Vol.1 MIT Press, 1989

Multi-Layered Perception Feed Forward Networks

Trained using the generalized delta rule/error back-propagation, an extension of the Widrow-Hoff LMS algorithm. The weights of the network are updated using gradient descent. The error at the output nodes is propagated backwards through the weights to obtain the errors at the hidden layer node outputs.



Schematic of a Multi-layered Feed Forward network

Neural Network models for Prediction

Feedforward Network for prediction:

Trained using	:	Backpropagation
Inputs	:	A vector of past values of the time series & a bias term
Output	:	The value to be predicted
Hidden Layers	:	1 or 2
Activation function	:	Linear (Output node) Sigmoid (hidden nodes)

Recurrent Neural Network for prediction

Trained using	:	RTRL algorithm
Inputs	:	x(n) & a bias term
Output	:	x(n+1)
Activation function	:	Linear (Output node) Sigmoid (hidden nodes)

Advantages in using RecNN over FFN:

- The model order need not be known, since the RecNN can retain past values
- The number of input nodes is lesser.

Alan Lapedes & robert Farber, "Non-linear Signal processing using Neural networks: Prediction & System modeling", Los Alamos National Laborator Report, LA-UR-87-2662.
J.Connor & L. Atlas, "Recurrent Neural Networks and time series prediction", IJCCN '91, Seattle, pp301-306

Radial Basis Function Network for Prediction

1 layered RBF:

Strict interpolation – use all data points available – good fitting accuracy – higher noise sensitivity – poor prediction in the presence of noise – computationally very intensive – lesser RBCs – how to choose the centers?

2 layered RBF:

Successive approximation & higher smoothing – reasonable fitting accuracy – low noise sensitivity – better prediction – computationally less burdensome

Issues:

- How to choose a kernel? Common choice – Gaussian kernel
- How to choose the spread? A function of the mean squared distance between data points
- How to choose RBF centers? By k means clustering or Orthogonal least squares

Advantages:

There is no iterative training. The calculation of the weights is done in one shot.

Has a good theoretical foundation in function approximation, unlike other NNs.

They readily suit parallel implementation

NEURAL NET SIGNAL MODELING

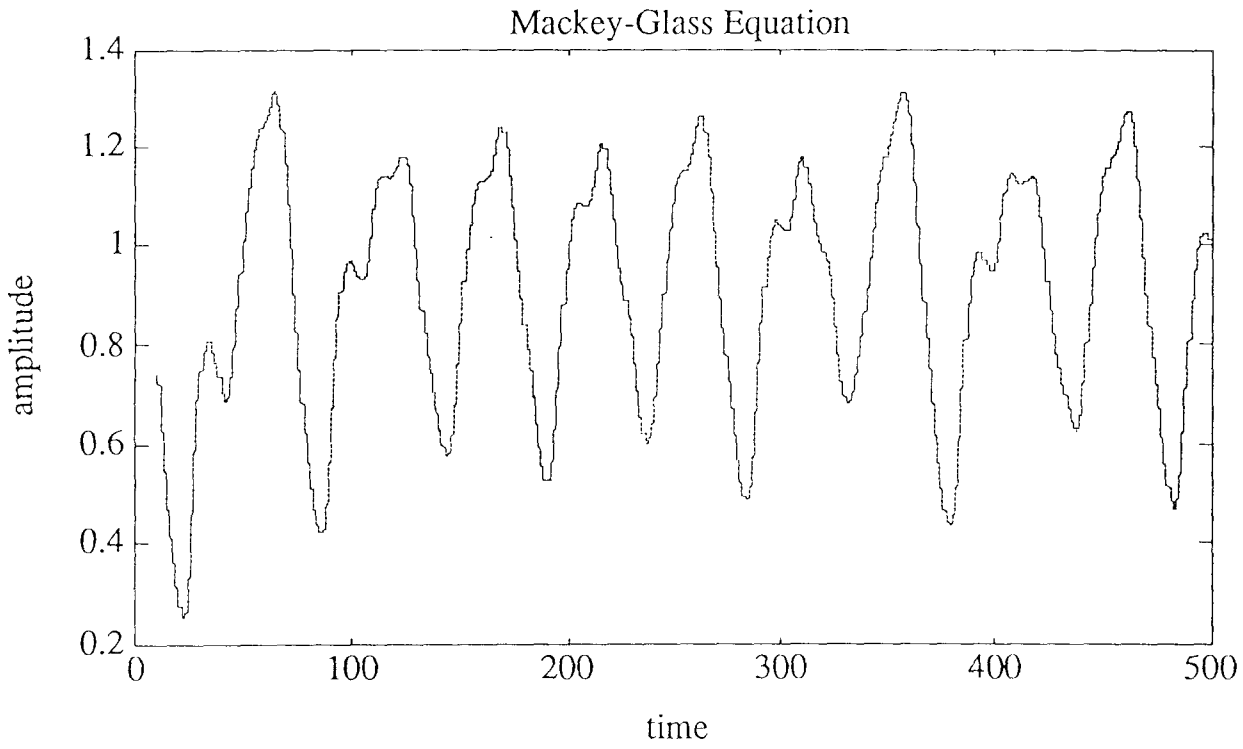
The Mackey-Glass equation is defined by the following equation

$$\frac{dx(t)}{dt} = \frac{ax(t-\tau)}{a + x^{10}(t-\tau)} - bx(t)$$

- set a = 0.2, b = 0.1
- for small solution is either fixed point or limit cycle
- for τ = 17 solution is chaotic with a strange attractor and fractal dimension of 2.1
- for τ = 30 solution yields a strange attractor with fractal dimension of 3.5

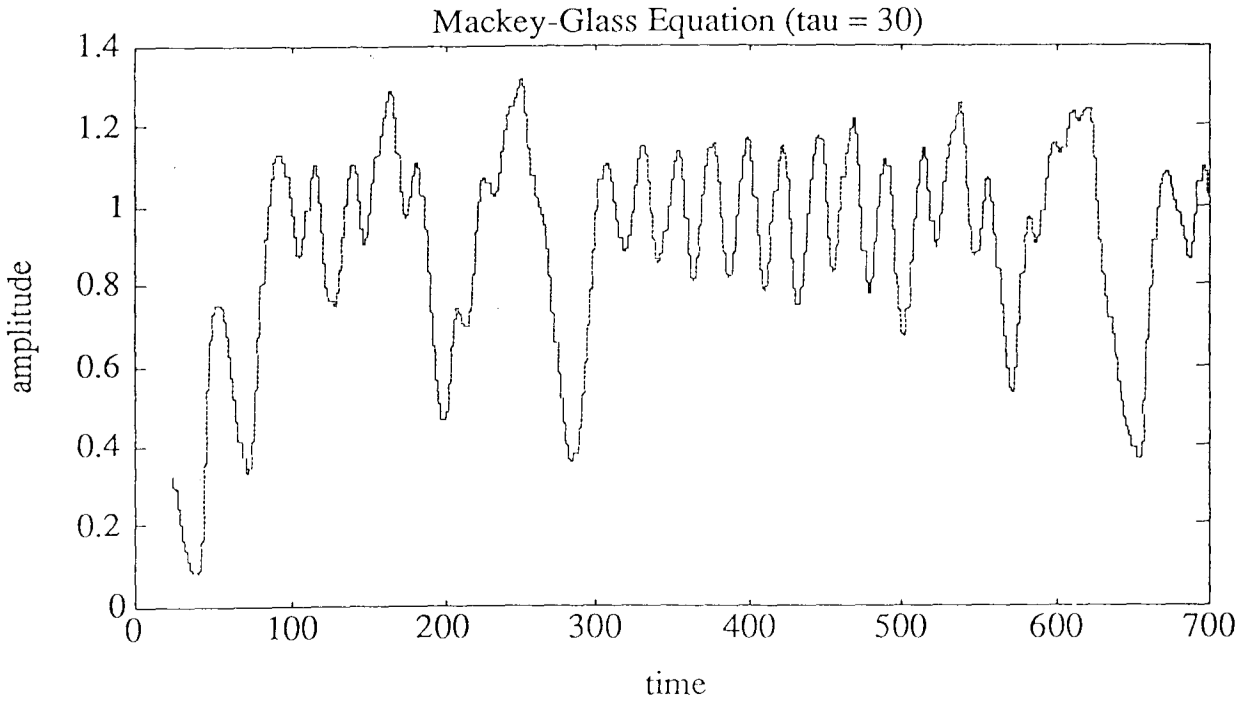
NEURAL NET SIGNAL MODELING

$\tau = 17$ solution



NEURAL NET SIGNAL MODELING

$\tau = 30$ solution



NEURAL NET SIGNAL MODELING

Performance Comparison for the Prediction Problem

Mean Squared Error	Logistic Map	Mackey-Glass Time Series	Sunspot Series *	Canadian Lynx Series *
SDM – Kalman Filter	1.77 E-05	0.0031	0.0364	0.0375
II Order Volterra Filter	4.48 E-04	0.0057	0.0758	0.0842
RBF Network	5.99 E-10	0.0025	0.0362	0.0240
Feedforward Network	0.0019	0.0081	0.0512	0.0448
Recurrent Network	0.1304	0.0007	0.0342	0.0338
Parallel Combination	0.0012	0.0017	0.0312	0.0371

* The values indicate the MSE when the Sunspot Series was scaled down by 100 and the Canadian Lynx Series was scaled down by a factor of 5000.

NEURAL NET SIGNAL MODELING

Performance Comparison for the Prediction Problem (contd.,)

Mean Squared Error	Exp. AR TS1
SDM – Kalman Filter	0.1503
II Order Volterra Filter	0.2319
Recurrent Network	0.1938
Parallel Combination	0.1212

Mean Squared Error	Exp. AR TS2
SDM – Kalman Filter	0.0323
II Order Volterra Filter	0.0608
Recurrent Network	0.0273

TS1: $$x(n) = \sum_{i=1}^{5} \left(\Gamma_i + \lambda_i e^{-(\gamma_i x(n-1)^2)} \right) x(n-i) + e(n)$$

$\Gamma 1 = 0.4$; $\lambda 1 = 0.8$; $\gamma 2 = -0.3$; $\Gamma 2 = -0.7$; $\lambda 2 = 0.9$; $\gamma 2 = 0.4$; $\Gamma 3 = 0$; $\lambda 3 = 0.1$; $\gamma 3 = 0.1$;
$\Gamma 4 = 0.4$; $\lambda 4 = 0.8$; $\gamma 4 = 0.4$; $\Gamma 5 = 0.4$; $\lambda 5 = 0.8$; $\gamma 5 = 0.4$;

TS1: $$x(n) = -0.8 e^{-(x(n-2)^2)} x(n-2) + e(n)$$

Simulation Results

Time series data considered to test the proposed scheme were –

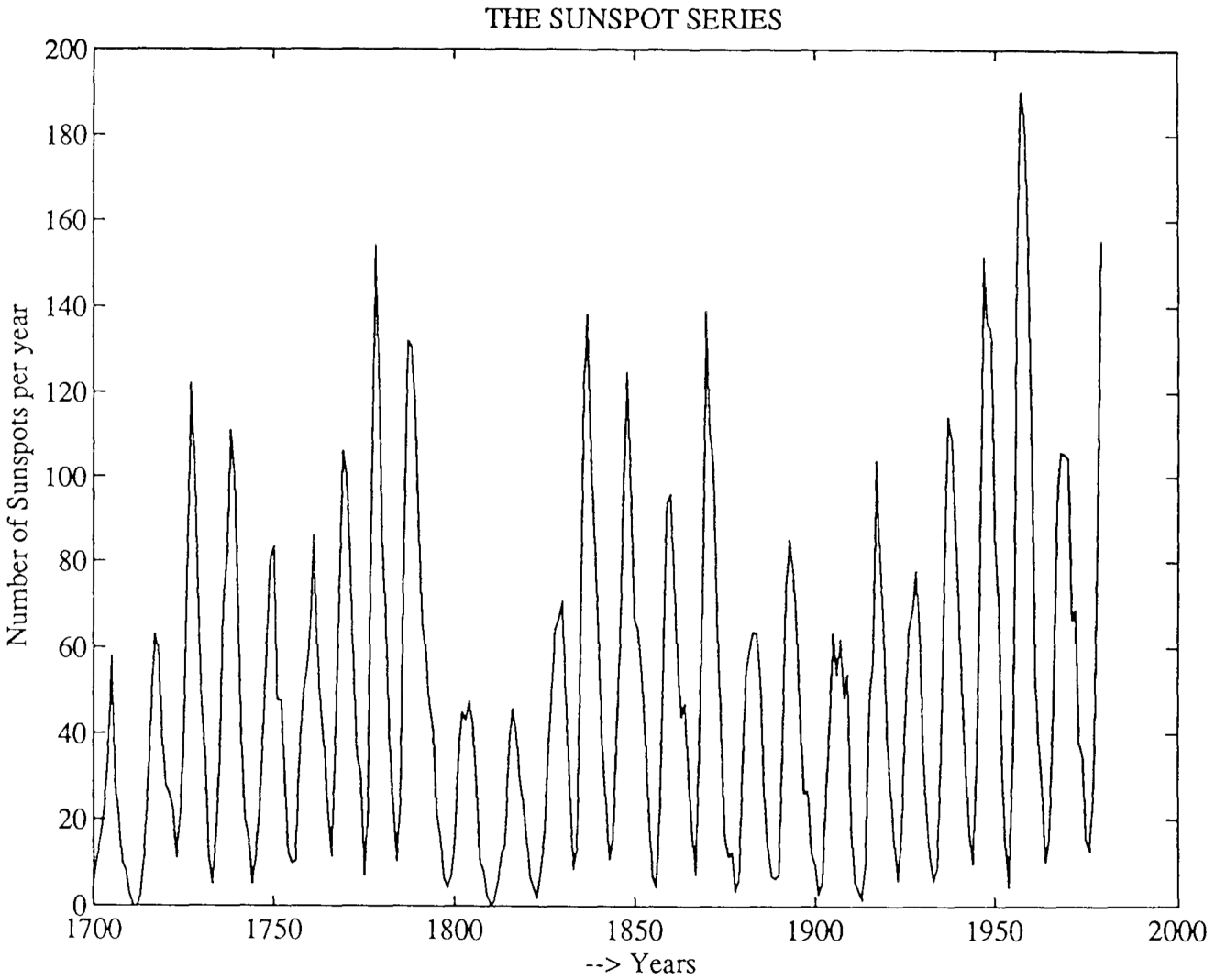
- Sunspot Series.
- Exponential autoregressive (AR) series based on the model

$$x(n) = \sum_{i=1}^{N} \left(\Gamma_i + \lambda_i e^{-(\gamma x(n-1)^2)} \right) x(n-i) + e(n)$$

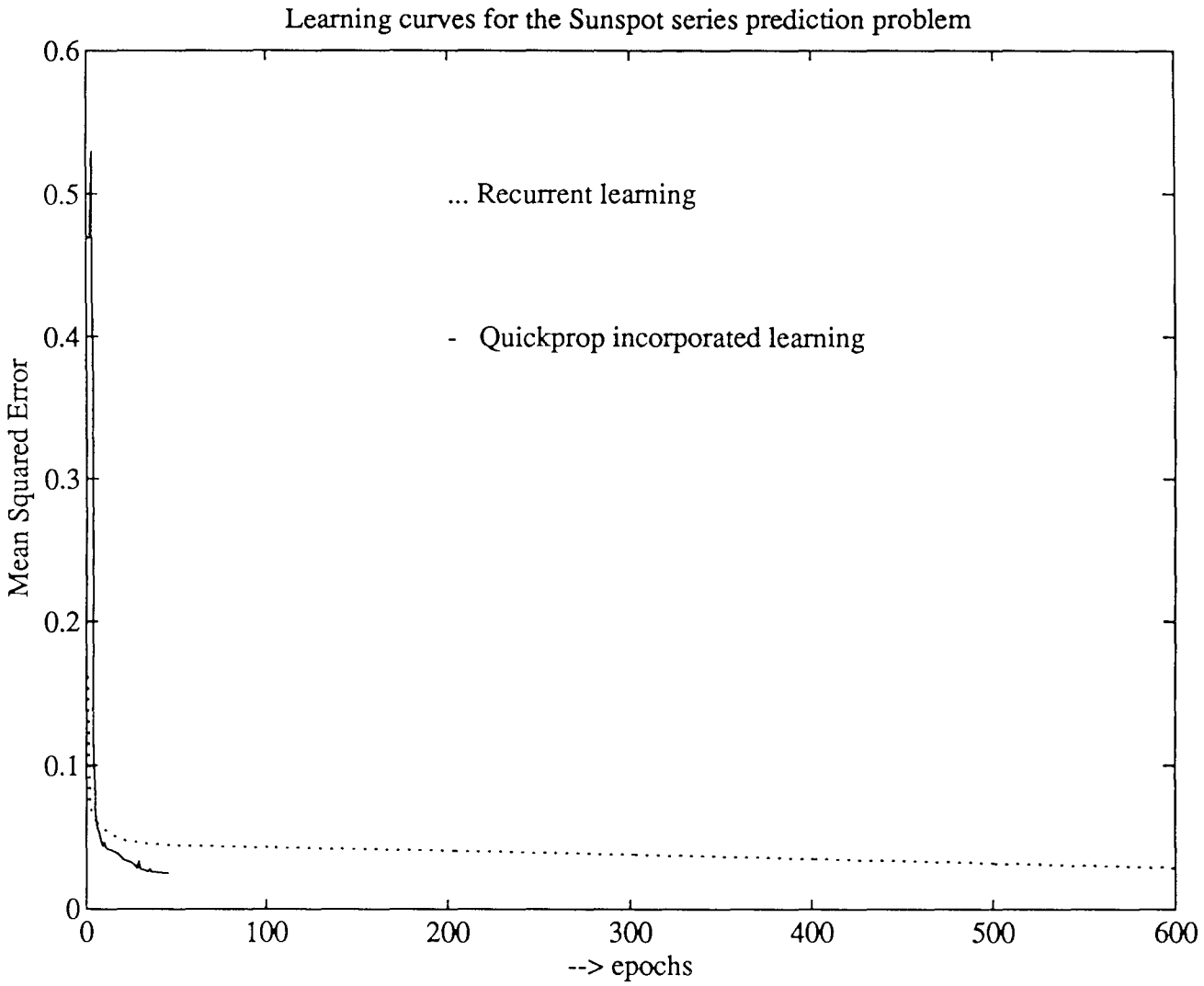
The parameters for the Exp. AR series were

N= 5; Γ1 = 0.4; λ1 = 0.8; γ2 = 0.4; Γ2 = -0.3; λ2 = -0.7; Γ3 = 0.0; λ3 = 0.1
Γ4 = 0.0; λ4 = 0.2; Γ5 = 0.0; λ5 = 0.8

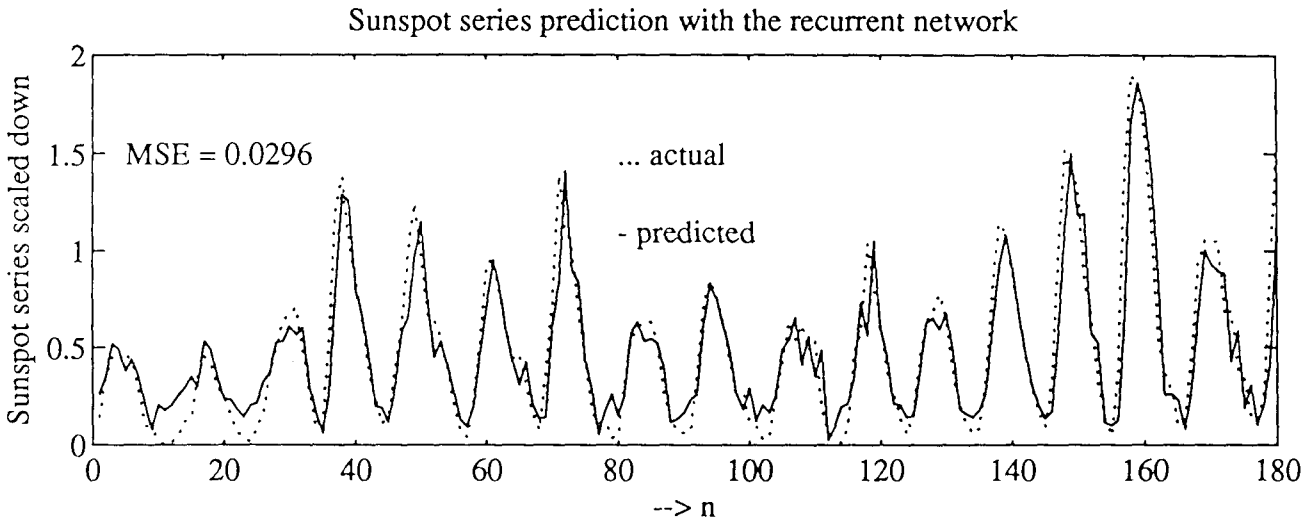
Simulation Results

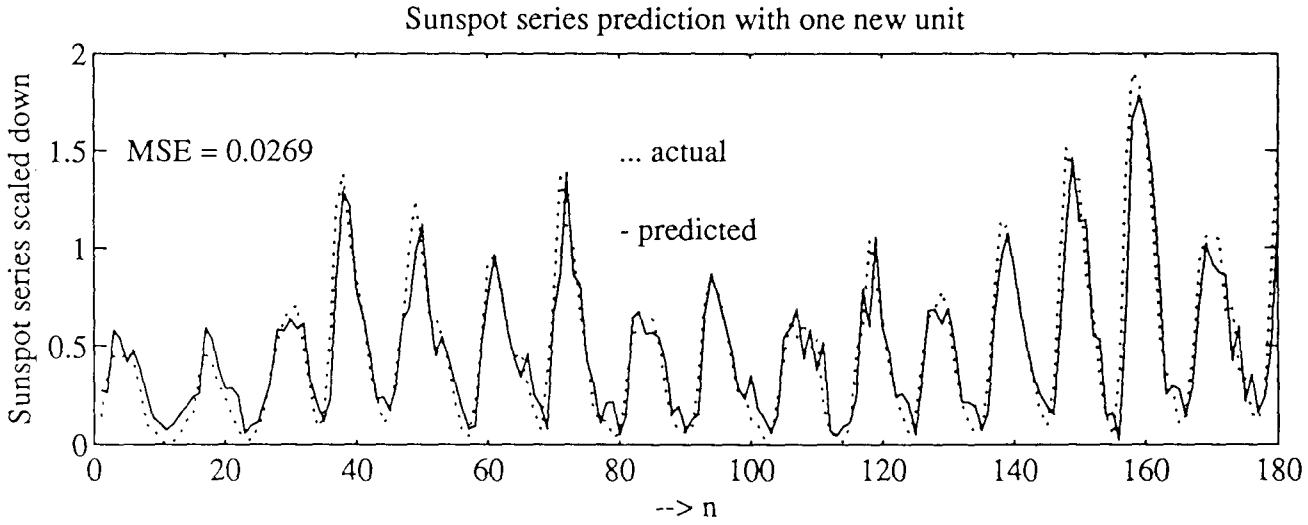


Simulation Results



Simulation Results



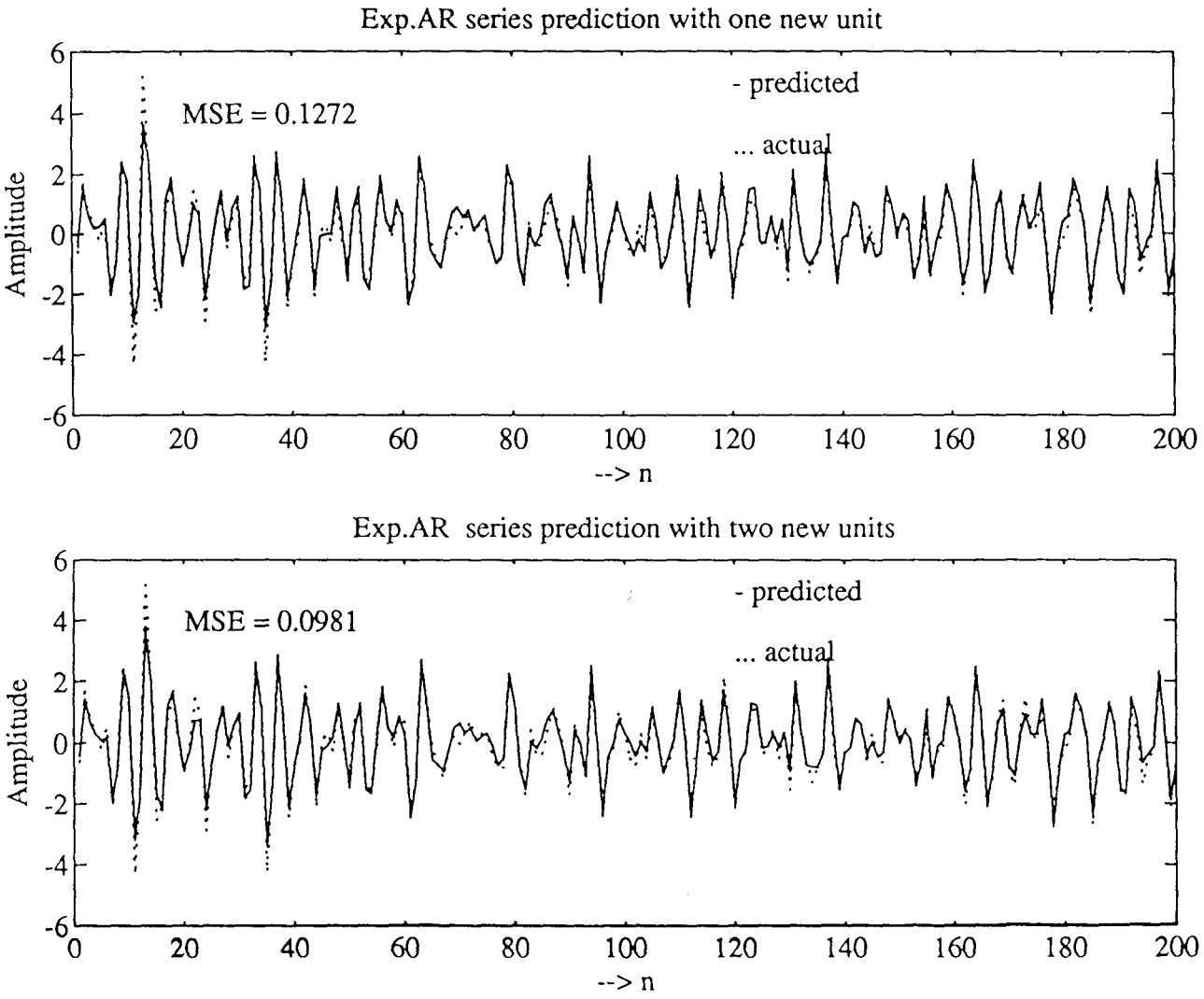


NEURAL NET SIGNAL MODELING

Performance Comparison for the Sunspot Series

Mean Squared Error	**Exp.AR TS1**
SDM – Kalman Filter	0.0364
II Order Volterra Filter	0.0758
RBF Network	0.0362
Feedforward Network	0.0512
Recurrent Network	0.0296
Parallel Combination	0.0312
Proposed Technique	0.0269

Simulation Results



Simulation Results