UNIVERSITY OF MINNESOTA

CENTER FOR TRANSPORTATION STUDIES

RESEARCH REPORT

Real-Time Prediction of Freeway Occupancy for Congestion Control

Vladimir Cherkassky
Sangkug Yi

CTS 97-12

Technical Report Documentation Page

1. Report No.	2.	3. Recipients Accession No.
CTS 97-12		

4. Title and Subtitle	5. Report Date
Real-Time Prediction of Freeway Occupancy for Congestion Control	September 1997
	6.

7. Author(s)	8. Performing Organization Report No.
Vladimir Cherkassky, Sangkug Yi	

9. Performing Organization Name and Address	10. Project/Task/Work Unit No.
Department of Electrical and Computer Engineering University of Minnesota 200 Union Street SE Minneapolis, MN 55455	11. Contract (C) or Grant (G) No. CTS Project #1994002

12. Sponsoring Organization Name and Address	13. Type of Report and Period Covered
	Final Report
	14. Sponsoring Agency Code

15. Supplementary Notes
http://www.its.umn.edu/Publications/ResearchReports/

16. Abstract (Limit: 250 words)

Accurate traffic prediction is critical for effective control of on-ramp traffic (ramp metering). During congestion, traffic shock waves propagate back and forth between the detectors, and traffic becomes inherently non-stationary and difficult to predict. Recently, several adaptive non-linear time series prediction methods have been developed in statistics and in artificial neural networks. We applied these methods to develop real-time prediction of freeway occupancy during congestion periods, from current and time-lagged observations of occupancy at several (neighboring) detector stations. This study used the following function estimation methodologies for real-time occupancy prediction: two statistical techniques, multivariate adaptive regression splines (MARS) and projection pursuit regression; two neural network methods, multi-layer perceptrons (MLP) and constrained topological mapping (CTM). All these methods were applied to freeway occupancy data collected on I-35W during morning rush hours. Data collected on one day was used for training (model estimation), whereas the data collected on a different day was used for testing, i.e., estimating the quality of prediction (generalization). Results for this study indicate that the proposed methodology provides 10-15% more accurate prediction of traffic during congestion periods than the approach currently used by Minnesota DOT.

17. Document Analysis/Descriptors	18. Availability Statement
Neural networks, Ramp metering, Function estimation, Occupancy prediction, Traffic congestion, Traffic forecasting	No restrictions. Document available from: National Technical Information Services, Alexandria, Virginia 22312

19. Security Class (this report)	20. Security Class (this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	37	

Real-Time Prediction of Freeway Occupancy for Congestion Control

Final Report

Prepared by:

Vladimir Cherkasskey
Sangkug Yi

Department of Electrical and Computer Engineering
University of Minnesota

September 1997

Published by:

Center for Transportation Studies
University of Minnesota
200 Transportation and Safety Building
511 Washington Ave. SE
Minneapolis, Minnesota 55455

CONTENTS

APPENDICES

1. Background and motivation

The goal of this project was to apply nonlinear adaptive modeling methods recently developed in Artificial Neural Networks and Statistics, for accurate short-term prediction of freeway occupancy during congested periods.
The data used for this study was collected from the three adjacent detector stations as shown in Fig.1. These stations will be refered to as the Up-Stream, Center, and Down-Stream station, respectively. Loop detectors at each station continuously measure and record highway occupancy, defined as the fraction of time the detector was on during a fixed time interval (i.e. 30 seconds). The measurements from several (i.e. three) loop detectors are then averaged to produce a single measurement of the freeway occupancy at a given time interval. Accurate knowledge of the freeway occupancy is essential for effective control of the ramp metering rates. Minnesota DoT uses a procedure known as occupancy control [Mn DoT, 1994], to choose appropriate setting of the on-ramp metering rate, based on the freeway occupancy measurements at the Central station. The main deficiency of this approach is the assumption that the measured freeway occupancy does not change significantly between adjacent 30 sec time intervals. Whereas this is generally true during non-congested periods when the traffic is smooth (i.e. changes slowly), during congestion the traffic becomes inherently non-stationary and the current measurement of the freeway occupancy becomes a poor predictor of the occupancy during the next time interval (see Appendix 1). Hence, the goal is to obtain better prediction of the freeway occupancy at the Central station, for improved control of ramp metering during congestion.

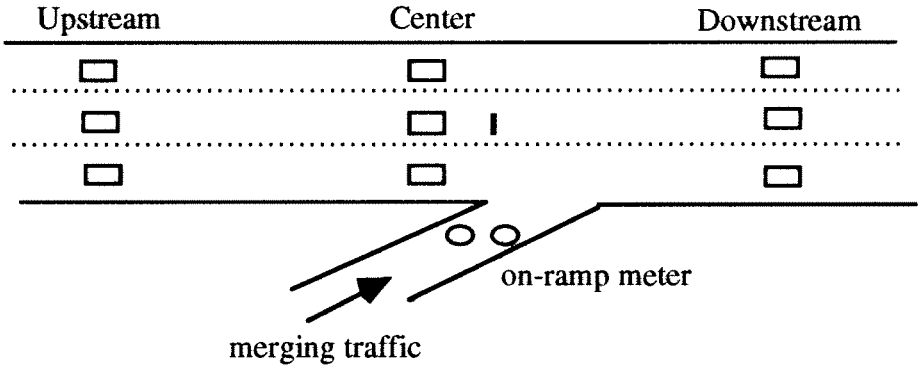


Fig.1 Detector stations set-up

Since past measurements of freeway occupancy at the Up-Stream, Central and Down-Stream stations are available at a given time interval (index) t, they may be (potentially) used to predict the freeway occupancy at the Central station at the next time interval, t+1.Thus the main objective of this study is to develop a model:

$$C(t+1) = \mathbf{f}\,[C(t-), U(t-), D(t-)] \qquad (1)$$

$$t = 1, 2, 3, ...$$

where C(t+1) is estimated current-station occupancy at time t+1;
C(t-) denotes present measurement C(t) along with all past (time-lagged) measurements C(t-1), C(t-2) .... which can be potentially used to predict C(t+1);
similarly, U(t-) and D(t-) denote all available measurements at the Upstream and Downstream station that can be used for prediction.

There are two basic approaches to estimating an unknown function **f** in (1):

- **first-principles modeling**, which involves development of models describing the underlying physical phenomenon. This can be done, for example, using a system of differential equations to describe traffic flow and congestion. Such approach may be too costly and difficult to implement, as it requires detailed modeling of a complex phenomenon (traffic congestion). Moreover, different models need to be constructed for different parts (sections) of a freeway. Usually, after the (first-principles) model is developed, its parameters need to be adjusted(estimated) using real-life data.

- **empirical data modeling**, where an unknown dependency F is estimated solely on the basis of available historical data, i.e. past measurements of freeway occupancy at the three stations. This is an approach adopted in this project.

The rest of this paper is organised as follows:
- section 2 provides background on adaptive methods for function estimation from samples, and describes representative methods used in this study;
- section 3 describes application of adaptive methods to the prediction of freeway occupancy;
- section 4 provides summary and discussion of future research directions.

2. Adaptive methods for function estimation and time series prediction

In the last decade, neural networks have given rise to high expectations for model-free statistical estimation from a finite number of samples(examples). However, there is increasing awareness that Artificial Neural Networks(ANNs) represent inherently statistical techniques subject to well-known statistical limitations [Cherkassky et al, 1994].

The goal of Predictive Learning[Friedman, 1994] is to estimate/learn an unknown functional mapping between the input (explanatory, predictor) variables and the output (response) variables, from a training set of known (input, output) samples. The mapping is typically implemented as a computational procedure (in software). Once the mapping is obtained/ infered from the training data, it can be used for predicting the output values given only the values of the input variables. Inputs and outputs can be continuous and/or categorical variables. When outputs are continuous variables, the problem is known as *regression or function estimation*; when outputs are categorical (class labels), the problem is known as *classification.* Here we consider only regression problems with a single (scalar) output, i.e. we seek to estimate a function **f** of N-1 predictor variables (denoted by vector X) from a given set of n training data points, or measurements, $Z_i = (X_i, y_i)$ (i = 1,...n) in N-dimensional sample space:

$$y = \mathbf{f}(X) + \text{error} \qquad (2)$$

where error is unknown (but zero mean) and its distribution may depend on X. The distribution of training data in X is also unknown and can be arbitrary.

Note that time series prediction can be viewed as generic function estimation (2) where an output y corresponds to predicted (future) value of a time series and predictor variables X correspond to current and/or time-lagged observations of a time series. For example, unknown time series dependency of the form:

$$C(t+1) = \mathbf{f}\,[\,C(t), C(t-1), U(t)] \qquad t = 0,1,2,....$$

corresponds to the general formulation (1) where

$$y = C(t+1) \quad \text{and} \quad X = <\,C(t), C(t-1), U(t)\,>$$

Nonparametric methods make no or very few general assumptions about the unknown function **f**(X). Nonparametric regression from finite training data is an ill-posed problem and meaningful predictions are possible only for sufficiently smooth functions. Additional complications arise due to inherent sparseness of high-dimensional training data (known as the *curse of dimensionality*) and the difficulty in distinguishing between signal and error terms in (2).

Successful application of adaptive methods recently developed in the field of ANNs and Statistics usually require sufficient user expertise in understanding of methods' operation. This is needed in order to tune appropriately user-defined parameters of a method. In order to overcome this problem, we developed a software package for non-expert users, that incorporates several powerful methods for function estimation from samples.

Software package XTAL for non-expert users. In order to enable/improve usability of adaptive methods by non-expert users, several statistical and neural network methods for non-parametric regression (developed elsewhere) were integrated into a single package with a uniform user interface(common for all methods). In addition, all methods were modified so that, at most, 1 or 2 parameters need to be user-defined (no limit is imposed on *internal* parameter tuning transparent to a user). Since most adaptive methods in the package originally had a large number of user-tunable parameters (typically half a dozen or so), most of these parameters were either set to carefully chosen default values or internally optimized in the final version included in the package. The package called XTAL was developed at the University of Minnesota in 1994-95. Detailed description of methods included in XTAL and instructions for obtaining XTAL software over Internet are included in Appendix 0.

3. Prediction of freeway occupancy

In this section we describe application of adaptive methods for function estimation to prediction of the freeway occupancy, i.e. estimating function **f** in eqn (1). The goal is to construct (estimate) short-term prediction of freeway occupancy (at Central station) from past measurements at the Central, Upstream and Downstream stations. By short-term prediction, we mean the prediction one time interval (30 sec) into the future. Longer term predictions do not seem feasible due to chaotic nature of traffic during congestion.
As we discussed in section 2, adaptive methods use the training data to construct the predictive model, whereas an independent test data set is used to estimate the predictive (generalization) performance of the model. In this study, freeway occupancy data from one day (say, Day 1) is used to estimate the model (i.e., as training data), whereas the occupancy measurements from another day (say, Day 2) are used as test data, in order to test the predictive power of the model.
There are three distinct components necessary for successful estimation of the predictive model from data:
- *data collection.* This is very important, since our approach is data-driven, and we need large amounts of statistically representative data in order to construct good predictive models. The data sets used in this study are discussed in section 3.1
- *selection of predictor variables.* There are infinitely many choices for a set of predictor variables that can be used in eqn (1) for predicting the occupancy at Central station at time t+1, i.e. current (or time-lagged) observations of freeway occupancy at all three stations. So the first task is to choose a small number of good predictor variables. This is done by correlating a large number of potential predictor variables with the predicted variable, i.e. the occupancy at the central station. The process of variable selction is described in section 3.2
- *nonlinear modeling and fine tuning.* (section 3.3) Here we apply different modeling methods to the available data of freeway occupancy using different combinations of best predictor variables obtained in section 3.1. We also optimise (fine tune) the modeling models as discussed in section 2. Finally, we discuss and comment on the modeling results.

3.1 Description and use of data sets

Freeway occupancy data has been collected for several sections of I-35W (northbound) in Spring 1995. Each section included 3 neighboring stations as shown in Fig.1. Data was collected during morning hour rush hour periods (6 a.m. to 9:30 a.m.) in 30 sec intervals, resulting in 3.5 x 60 x 2 = 420 samples per station per day. Specific sections of the freeway included the following groups of stations:

41, 42 and 43 (or 090N, 086N and 082N, respectively);
55, 56 and 57 (or 50N, 46N and 42N, respectively);
58, 59 and 60 (or 38N, 35N and 31N, respectively).

The data was collected weekly on Tuesday, Wednesday and Thursday and visually examined for the presence of congestion. According to Mn DoT, congested condition corresponds to freeway occupancy above 20%. It was found that most of the observed data does not exhibit congestion according to this definition, with the exception of the section comprising stations 41, 42 and 43. Hence, only this section data was used for subsequent modeling. Even for this section, congestion does not occur consistently (every day the measurements have been taken), and when it does occur, it has widely varying pattern. For this study, the occupancy measurements on 7 (congested) days were used for constructing/evaluating the prediction model. Appendix 1 shows the data sets used in the form of the time series corresponding to the occupancy measurements at the Central station (station 42) from 6 a.m. to 9:30 a.m., in 30 sec intervals (total 420 samples).

Note the different shape of time series during congestion periods during different days, suggesting the complexity of the prediction task.
From this 7 days of data, we (arbitrarily) choose two days, and use one day's data as a training set (to estimate the predictive model according to eqn. 1) and the second day as test data (to evaluate how good the model is).

3.2 Variable selection

The natural choice of variables for predicting the occupancy at the central station C(t+1) are the current occupancy at the central, upstream and downstream stations. This choice of predictor variables corresponds to the assumed model of the form:

$$C(t+1) = F [C(t),U(t),D(t)]$$

where the unknown function F needs to be estimated from data (training set).
Other possible predictor variables include:
time-lagged occupancy measurements, i.e. C(t-1), U(t-2) etc.
the difference of the type C(t) - U(t) etc.
change of the occupancy level at a station, such as C(t) - C(t-1) or U(t) - U(t-1) etc.
The number of possible predictor variables (and their combinations) is huge; however we cannot use all of them, since we only have limited training data. It is necessary to choose a small subset of 2-3 most important predictor variables prior to application of adaptive modeling methods. The proper choice predictor variables was based on correlating the occupancy at the central station with potential predictor variables (i.e. time-lagged measurements of all three stations). Most important predictor variables are likely to have large correlation coefficient, consistently for all data sets (corresponding to congested days).
Following extensive correlation analysis of tens of potential predictor variables, we identified the following 3 variables C(t), U(t) - C(t), D(t) -C(t) as most promising predictors for C(t + 1). Since variable C(t) had the largest correlation coefficient, it was included in all models. Hence, we applied adaptive methods to estimate an unknown dependency in the following models:

$$C(t+1) = F [C(t), C(t) - U(t), D(t) - C(t)]$$
$$C(t+1) = F [C(t), C(t) - U(t)]$$
$$C(t+1) = F [C(t), D(t) - C(t)]$$
$$C(t+1) = F [C(t)]$$

where time index t = 0,1,2,3 .....

3.3 Modeling and occupancy prediction results

We used various modeling methods available in XTAL package (as described in section 2) for developing short-term predictive models from available data (i.e. training and test day pairs as described in section 3.1). For most train-test combinations, the following methods consistently provided superior results:
Projection Pursuit , Multivariate Adaptive Regression Splines (MARS) and Constrained Topological Mapping (CTM).
Representative predictive models developed by applying these methods to the training data are discussed next. Note that the models are obtained using the training data, whereas a model's prediction (generalization) performance is estimated using (independent) test data. In this study, training and test data correspond to two different days.
First consider statistical summary of the observed data used for modeling:

OCCUPANCY DATA STATISTICS at C(t) during congestion

date	**mean**	**st. dev.**
4.11	22.49	4.52
4.12	21.90	5.19
5.3	23.66	5.67

Note that the above statistics was calculated over congested periods only, rather than over the whole observation interval from 6 to 9:30 a.m. (see the corresponding time series in Appendix 1).
Representative prediction results shown below include specification of:
- predictor variable(s) used in modeling;
- training and test dates;
- two indices for prediction error on the test data, i.e. RMS error and MAX error, where RMS (root-mean-squared) error measures *average* prediction accuracy, whereas MAX (maximum) error provides the *worst* (absolute) prediction error.

All modeling results estimate short-term prediction model using one day's training data over the whole observation period (i.e. 6 a.m. to 9:30 a.m.), whereas the model's predictive ability is evaluated during the congested period of another (test) day data.

PREDICTIVE MODEL C(t+1) = F [C(t)]

training	/ test date	RMS	MAX	modeling method
4.11	4.12	3.8	15.2	MARS
4.11	5.3	3.7	9.7	MARS
4.12	4.11	3.6	14.4	PPR
4.12	5.3	3.7	8.8	PPR

PREDICTIVE MODEL C(t+1) = F [C(t), U(t) - C(t)]

training	/ test date	RMS	MAX	modeling method
4.11	4.12	3.8	14.6	MARS
4.11	5.3	3.7	9.5	CTM
4.12	4.11	3.9	15.7	MARS
4.12	5.3	3.9	9.7	CTM

PREDICTIVE MODEL C(t+1) = F [C(t), C(t) - U(t), D(t) - C(t)]

training	/ test date	RMS	MAX	modeling method
4.11	4.12	3.6	15	MARS
4.11	5.3	4.0	10.3	CTM
4.12	4.11	3.9	15.2	MARS
4.12	5.3	3.9	9.8	CTM

PREDICTIVE MODEL C(t+1) = C(t)

DATE	RMS	MAX	vs	best nonlinear prediction RMS	MAX
4.11	4.0	14		3.6	14.4
4.12	4.2	15		3.6	9.7
5.3	4.2	13		3.7	9.5

As one can see from the above comparison table, adaptive methods for nonlinear prediction provide 10% - 15% improvement in RMS error over trivial prediction model (future occupancy equals present observation) currently used by Mn DoT.
Several predicted time series, along with true (measured) occupancy data are shown in Appendix 2. Visual inspection of these time series suggests that even best nonlinear predictive models are rather inaccurate. This is not surprising, since the statistical properties of the occupancy time series (measured on the same portion of a freeway) vary significantly from day to day - see examples in Appendix 1. Hence, the properties of the test data and the training data sets corresponding to different days are quite different. For example, compare predictive models developed for 4.11.95 (using 4.12.95 as training data) vs the model for 4.12.95 (using 4.11.95 as training data), - the corresponding model plots are shown in Appendix 3. These models are indeed different, even though there is strong (qualitative) similarity in the shape of both models. Examination of nonlinear models using a single predictor variable C(t) shown in Appendix 3 indicates that incorporating such (nonlinear) model into the procedure for controling on-ramp metering rate durinng congestion currently used by the Traffic Management Center can reduce RMS prediction error by 10-15%.

4. CONCLUSION and FUTURE DIRECTIONS

We applied adaptive nonlinear statistical and neural network methods for short-term prediction of freeway occupancy from the current and time-lagged observations at several neighboring stations. Based on the modeling results, we have observed only limited success (i.e. 10 - 15 % improvement in RMS error) in comparison with method currently used by Mn DoT for occupancy prediction. The reasons for this rather modest improvement in prediction accuracy became clear during our discussions with MN DoT Traffic Management Center (TMC), as explained next. According to TMC, traffic congestion is typically caused by one-at-a-time events (i.e. accidents, poor weather conditions etc.). These events affect the statistics of the (measured) occupancy time series in rather unique ways, which makes prediction (generalization) on the basis of past (training) data difficult. In addition, prediction accuracy is likely to be improved if other available measurements (such as volume) are included in the prediction model. Unfortunately, since we experienced unexpected delay in obtaining (occupancy) data from MN DoT database, we did not have time to include the volume data in the prediction model.
There seems to be three promising directions for continuation of this research:
(a) developing more accurate occupancy prediction models by including the volume as another predictor variable;
(b) incorporating nonlinear model based on a single predictor variable C(t) (as shown in Appendix 3) into the procedure for controling on-ramp metering rate durinng congestion currently used by the Traffic Management Center. Results of our study suggest that nonlinear modeling of C(t+1) as a function of C(t) would reduce RMS predicion error by 10-15% in comparison with a trivial prediction C(t+1) = C(t) currently used by Mn DoT. However, more empirical evidence is needed to provide confidence in the improved prediction of the nonlinear model;

(c) developing an intelligent system for characterization of occupancy time series during congestion at the critical parts of a freeway. The objective here is to identify similar "modes" or "types" of congestion. For example, congestion may be caused by an accident, or weather condition (heavy snow), or unusual traffic pattern (popular athletic event). Assuming that different causes produce different patterns of congestion, the goal would be to find a compact representation (model) of congestion time series from the past (historical) data which also includes the cause of congestion. Then one can use such condensed signatures of congestion for:
- real-time diagnosis of events that cause congestion (i.e. types of accidents);
- efficient storage of historical congestion data. Note that 30-sec occupancy data is currently stored only for one week, after that it is averaged over 5 min intervals for long-term storage. It is obvious that such data (5-min average) is useless for representation/characterization of congestion. However, condensed representations would provide very accurate representation of 30 sec occupancy data at a very low memory cost;
- prediction of congestion for improved congestion control during special events by TMC. For example, it may be possible to predict congestion patterns due to anticipated events, such as downtown convention or (predicted) snowstorm.

REFERENCES

[MN DoT, 1994] Ramp metering by zone - the Minnesota experience, MN DoT

V. Cherkassky, J.H. Friedman and H. Wechsler (Eds), From Statistics to Neural Networks: Theory and Pattern Recognition Applications, NATO ASI Series F, v.136,Springer, 1994.

Friedman, J.H., An Overview of predictive learning and function approximation, in Cherkassky, V., J.H. Friedman and H. Wechsler (Eds), From Statistics to Neural Networks: Theory and Pattern Recognition Applications, NATO ASI Series F, v.136,Springer, 1994.

APPENDIX 0 :

Description of XTAL software package.

Tutorial overview of adaptive methods for function estimation used in XTAL

XTAL Package (version 5.1)
A Multimethod program for function estimation

XTAL was developed by
Vladimir Cherkassky
Don Gehring
Filip Mulier
U of M EE Dept.
Release date: version 5.1 - May 3, 1995

GENERAL INFO: XTAL (stands for Crystal) combines several function estimation modules developed by different authors, under a uniform user interface. Individual methods were coded in FORTRAN (MARS, Projection Pursuit), C (CTM and k-nearest neighbors) and C++(ANN). The package should run on a SUN workstation, but you may have difficulty compiling in other environments. XTAL package contains user instructions for interfacing with XTAL and explains user-defined parameter settings for each method. Instructions are in the file **xtal_doc.txt**

Individual modules were developed by
J. H. Friedman (Stanford U.): MARS, PP -use FORT77 compiler
F. Mulier (U of M): CTM - CC compiler
T. Masters: ANN - use G++ compiler
D. Gehring (U of M): k-NN - use CC compiler

***** This package is for research and educational use only. *****

Instructions for transfering XTAL files from ee machine

1. ftp to oz.ee.umn.edu and give 'anonymous' as username
2. files are found in /users/cherkass
3. be sure to use binary transfer mode for ftp
4. ee machine users transfer the following files:
   readme - describes how to transfer files
   xtal_doc.txt - documentation on XTAL
   xtal_ee_bin.tar - tar file containing run-time code for ee machine

   general users transfer the following files:
   readme - describes how to transfer files
   xtal_v41.ascii_doc - documentation on XTAL
   xtal.tar - tar file full source code for XTAL and run-time for Sun

untarring the files:

type tar -xf xtal_ee_bin.tar OR tar -xf xtal.tar

depending on the file you transferred.

Tutorial overview of adaptive methods for function estimation used in XTAL

Here we consider only regression problems with a single (scalar) output, i.e. we seek to estimate a function **f** of N-1 predictor variables (denoted by vector X) from a given set of n training data points, or measurements, $Z_i = (X_i, y_i)$ (i = 1,...n) in N-dimensional sample space:

$$y = \mathbf{f}(X) + \text{error} \qquad (1)$$

where error is unknown (but zero mean) and its distribution may depend on X. The distribution of training data in X is also unknown and can be arbitrary.

It is important to provide a common taxonomy of statistical and neural network methods for function estimation (from samples) in order to understand their properties.

Here we follow the representation-scheme taxonomy where the function is estimated as a linear combination of basis functions, or basis function expansion[Friedman, 1994]:

$$\hat{f}(\mathbf{x}) = \sum_{j=0}^{M} a_j B_j(\mathbf{x}, \mathbf{p}_j) \qquad (2)$$

where
- $\mathbf{x}$ is a vector of input variables
- a_j are expansion coefficients (to be determined from data)
- $B(\mathbf{x}, \mathbf{p})$ are basis functions
- $\mathbf{p}_j$ are parameters of each basis function
- usually $B(\mathbf{x}, \mathbf{p}_0) = 1$
- M is the regularization parameter of a method.

This taxonomy is also known as *dictionary* methods [Friedman, 1994], since the methods differ according to the set of the basis functions (or dictionary) they use. We can further distinguish between non-adaptive (parametric) and adaptive methods, as follows:

- *non-adaptive methods* use *pre-set* basis functions (and their parameters), so that only coefficients a_j are fit to data. Optimal values for a_j are (usually) found by least squares from n training samples, by minimizing

$$\sum_{i=1}^{n} \left[y_i - \sum_{j=0}^{M} a_j B_j(\mathbf{x}_i, \mathbf{p}_j) \right]^2 \qquad (3)$$

There are two major classes of non-adaptive methods, i.e. global parametric methods (such as linear and polynomial regression), and local parametric methods (such as kernel smoothers, piecewise-linear regression and splines). For a good discussion of non-adaptive methods, see [Friedman,1991]. Note that global parametric methods inevitably introduce bias and local parametric methods are applicable only to low-dimensional problems (due to inherent sparseness of finite samples in high-dimensions known as the Curse of Dimensionality). Hence, *adaptive* methods are the only practical alternative for high-dimensional problems.

- *adaptive methods*, where (in addition to coefficients a_j) basis functions themselves and/or their parameters $\mathbf{p}_j$ are adapted to data. For adaptive methods optimization (3) becomes a difficult (nonlinear) problem. Hence, the optimization strategy used becomes very important. Statistical methods usually adopt greedy optimization strategy (stepwise selection) where each basis function is estimated one at a time. In contrast, neural network methods usually optimize over the whole set of basis functions.

Adaptive methods can be further classified as:

- *Global methods*, which use basis functions *globally* defined in the domain of **x**. Most popular choice is univariate basis functions of projections (linear combinations) of the input variables, as in Artificial Neural Networks and Projection Pursuit (see below). This choice is very attractive since it automatically achieves dimensionality reduction;
- *Local methods,* that use *local* basis functions (in **x**-space). Such methods either use local basis functions explicitly (such as Radial Basis Function networks with locally-tuned units) or do it implicitly via adaptive distance metric in **x**-space (as in adaptive-metric nearest neighbors, adaptive kernel smoothers, partitioning methods such as CART etc.). Such methods effectively perform data-adaptive local feature selection.

Description of representative methods

Based on the taxonomy of methods presented above, a number of regression methods has been selected for comparison and included in the XTAL package. These methods were chosen in order to represent a member of each of the major classes of methods. Each method has usable (public-domain) software implementation developed by its original author(s). These methods, along with appropriate setting of their parameters used in XTAL, are described next.

Nearest neighbors

A simple version of k-nearest neighbors regression was implemented in the XTAL package for benchmark purposes. Nearest neighbors is a *locally parametric method* where the response value for a given input is an average of the k closest training samples (in **x**-space) to this input. The value of k controls the amount of smoothing performed and is set by the user in the XTAL package.

Projection pursuit

Projection Pursuit [Friedman and Stuetzle, 1981] is a *global adaptive method* which exhibits good performance in high-dimensional problems and is invariant to linear coordinate transformations. The model generated by this method is the sum of univariate functions g_i of linear combinations of the elements of **x**

$$\hat{f}(\mathbf{x}) = \sum_{j=1}^{M} g_j\left(\mathbf{p}_j^T \mathbf{x}\right) \qquad (4)$$

The parameters $\mathbf{p}_j$ and the functions g_j are adaptively optimized based on the data. For each of the M projections, the algorithm determines the best $\mathbf{p}_j$ using a gradient descent technique to search for the projection which minimizes the unexplained variance. Each g_j is a smoothed version of the projected data with smoothing parameters chosen according to a fit criteria such as cross-validation, Since the model is additive, the search for function projections is done iteratively using the so-called backfitting algorithm. This is a greedy optimization technique where each additive term is estimated one at a time. The model is decomposed based on unexplained variance:

$$y - \sum_{\substack{j=1 \\ j \neq k}}^{M} g_j\left(\mathbf{p}_j^T \mathbf{x}\right) = g_k\left(\mathbf{p}_k^T \mathbf{x}\right) \qquad (5)$$

The $\mathbf{p}_k$ is optimally chosen using gradient descent while holding the $\mathbf{p}_j$, $j \neq k$ fixed. In each iteration another term is pulled out of the summation and an optimal $\mathbf{p}_k$ is found. This procedure is repeated until the average residual does not vary significantly. In this way, each function projection is chosen to best fit the largest unexplained variance of the data.

The original implementation of projection pursuit, called SMART (Smooth Multiple Additive Regression Technique)[Friedman, 1984], employs a heuristic search strategy for selecting the number of projections in order to avoid poor solutions due to multiple local minima. The SMART user must select the largest number of projections (M_L) to use in the search as well as the final number of projections (M_F). The strategy is to start with M_L projections and remove projections based on their relative importance until the model has M_F projections. The model with M_F projections is then returned as the regression solution. To improve ease of use in the XTAL package, M_F is set by the user, but M_L is always taken to be $M_F + 5$. Also, the SMART package allows the user to control the thoroughness of optimization. For the XTAL implementation, this was set to the highest level.

Artificial Neural Network (ANN)

Multilayer perceptrons with a single hidden layer and a linear output unit compute a linear combination of basis functions (2), where the basis functions are fixed (sigmoid) univariate functions of linear combinations of input variables. This is a global adaptive method in our taxonomy. Various training (learning) procedures for such networks differ primarily in the optimisation strategy used to estimate parameters (weights) of a network. XTAL package uses a version of multilayer feedforward network with a single hidden layer described in [Masters, 1993]. This version employs conjugate gradient descent for estimating model parameters (weights) and performs a very thorough (internal) optimization via simulated annealing to escape from local minima (10 annealing cycles). The original implementation from [Masters, 1993] was used with minor modifications. The method's implementation in XTAL has a single user-defined parameter - the number of hidden units. This is the complexity parameter of the method. There is close similarity between Projection Pursuit Regression and ANN in terms of representation, as both methods use nonlinear univariate basis functions of linear combinations (projections) of input variables. However, the two methods use very different optimization procedures: PPR uses greedy (stepwise) optimization to estimate additive terms in (2) one at a time, whereas ANN training estimates all the basis functions simultaneously.

2.2.4 Multivariate Adaptive Regression Splines

MARS [Friedman 1991] is a *global adaptive method* in our taxonomy. This method combines the idea of recursive partitioning regression (CART) [Breiman et al., 1984] with function representation based on tensor-product splines. The method of recursive partitioning consists of adaptively splitting the sample space into disjoint regions and modeling each region with a constant value. The regions are chosen based on a greedy optimization procedure where in each step, the algorithm selects the split which causes the largest decrease in mean squared error. A basis function for each region can be described by

$$B_j(\mathbf{x}) = I\left[\mathbf{x} \in R_j\right] \qquad (5)$$

where I is the indicator function. In this case I has the value one if the vector $\mathbf{x}$ is in region R_j and zero otherwise. The model can then be described by the following expansion on these basis functions

$$\hat{f}(\mathbf{x}) = \sum_{j=1}^{M} a_j B_j(\mathbf{x}). \qquad (6)$$

The MARS method is based on similar principles of recursive partitioning and greedy optimization, but uses continuous basis functions rather than ones based on the indicator function. The basis functions of the MARS algorithm can each be described in terms of a two-sided truncated power basis function (truncated spline) of the form

$$b_q^{\pm}(x-t) = \left[\pm(x-t)\right]_+^q \qquad (7)$$

where t is the location of the knot, q is the order(of splines) and the + subscript denotes the positive part of the argument. The basic building block of the MARS model is a pair of these basis functions which can be adjusted using coefficients to give a local approximation to data (Figure 1).

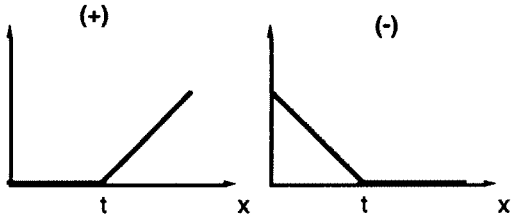


Figure 1 Pair of one-dimensional basis functions used by the MARS method.

For multivariate problems, products of the univariate basis functions are used. The basis functions for MARS can be described by

$$B_j^{(q)}(x) = \prod_{k=1}^{K_j} \left[s_{j,k} \cdot \left(\mathbf{x}_{v(j,k)} - t_{j,k} \right) \right]_+^q . \qquad (8)$$

This is a product of one-dimensional splines each with a directional term ($s_{j,k} = \pm 1$). The variable K_j defines the number of splits required to define the region j, v indicates the particular variable of $\mathbf{x}$ used in the splitting and $t_{j,k}$ is the split point.

The MARS model can be interpreted as a tree where each node in the tree consists of a basis function and uses a tree-based algorithm for constructing the model. Like other recursive partitioning methods, nodes are split according to a goodness of fit measure. MARS differs from other partitioning methods in that all nodes (not just the leaves) of the tree are candidates for splitting. Figure 2 shows an example of a MARS tree.

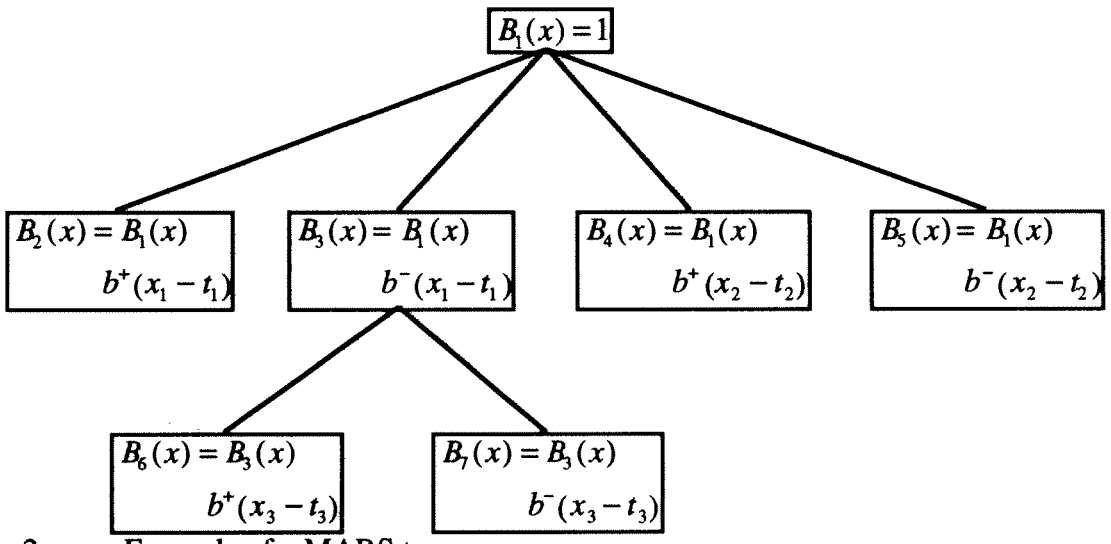


Figure 2 Example of a MARS tree.

The function described is

$$\hat{f}(\mathbf{x})=\sum_{j=1}^{7} a_j B_j(\mathbf{x}) \qquad (9)$$

The depth of the tree indicates the interaction level. On each path down variables are split at most once. The algorithm for constructing the tree is a forward stepwise and backwards stepwise strategy. In the forward stepwise procedure, a search is performed over every node in the tree to find a node, which when split improves the fit according to the fit criteria. This search is done over all candidate variables, split points $t_{j,k}$, and basis coefficients. For example, in Figure 2 the root node $B_1(\mathbf{x})$ is split first on variable x_1, and the two daughter nodes $B_2(\mathbf{x})$ and $B_3(\mathbf{x})$ are created. Then, the root node is split again on variable x_2 creating the nodes $B_4(\mathbf{x})$ and $B_5(\mathbf{x})$. Finally node $B_3(\mathbf{x})$ is split on variable x_3. In the backwards stepwise procedure, leaves are removed which cause either an improved fit or slight degradation in fit as long as model complexity decreases.

The measure of fit used by the MARS algorithm is the generalized cross validation (GCV) estimate [Craven and Wahba, 1979]. This GCV measure provides an estimate of the future prediction accuracy and is determined by measuring the mean squared error on the training set and penalizing this measurement to account for the increase of variance due to model complexity. The user can select the amount of penalization imposed (in terms of degrees of freedom) for each additional split used by the algorithm. Theoretical and empirical studies seem to indicate that adaptive knot location adds between 2 and 4 additional model parameters for each split [Friedman, 1991]. The user also selects the maximum number of basis functions and the interaction degree for the MARS algorithm. In the XTAL implementation, the user selects the maximum number of basis functions and the degrees of freedom (recoded as an integer from 0-9). The interaction degree is defaulted to allow all interactions.

The MARS method is well suited for high as well as low dimensional problems with a small number of low order interactions. An interaction occurs when the effect of one variable depends on the level of one or more other variables and the order of the interaction indicates the number of interacting variables. Like other recursive partitioning methods, MARS is not robust in the case of outliers in the training data. It also has the disadvantage of being sensitive to coordinate rotations. For this reason, the performance of the MARS algorithm is dependent on the coordinate system used to represent the data. This occurs because MARS partitions the space

into axis-oriented subregions. The method does have some advantages in terms of speed of execution, interpretation, and relatively automatic smoothing parameter selection.

Constrained Topological Mapping (CTM)

CTM[Cherkassky and Lari-Najafi, 1991] is an example of *local adaptive method.* In terms of representation of the regression estimate, CTM is similar to CART, where the input ($\mathbf{x}$) space is partitioned into disjoint (unequal) regions, each having a constant response (output) value. However, unlike CART's greedy tree partitioning, CTM uses (non-recursive) partitioning strategy originating from the neural network model of Self-Organizing Maps [Kohonen, 1989]. In CTM model, a (high-dimensional) regression surface is estimated (represented) using a fixed number of "units" (or local prototypes) arranged into a (low-dimensional) topological map. Each unit has ($\mathbf{x}, y$) coordinates associated with it, and the goal of training (self-organization) is to position the map units in order to achieve faithful approximation of the unknown function. The CTM model uses a suitable modification (for regression problems) of the original Self-Organizing Maps (SOM) algorithm [Kohonen, 1989] in order to faithfully approximate the unknown regression surface from the training samples. The main modification is that the best-matching unit step of SOM algorithm is performed in the space of predictor variables ($\mathbf{x}$-space), rather than in the full ($\mathbf{x}$, y) sample space[Cherkassky and Lari-Najafi, 1991]. The effectiveness of SOM / CTM methods in modeling high-dimensional distributions / functions is due to the use of low-dimensional maps. The use of topological maps effectively results in performing kernel smoothing in the (low-dimensional) map space, which constitutes a new approach to dimensionality reduction and dealing with the Curse of Dimensionality [Mulier and Cherkassky, 1995].

The trained CTM provides piecewise-constant interpolation between the units The constant-response regions are defined in terms of the Voronoi regions of the units in the predictor space. Prediction based on this model is essentially a table lookup. For a given input $\mathbf{x}$, the nearest unit is found in the space of the predictor variables and the piecewise constant estimate for that unit is given as response.

Improvements based on statistical considerations

Empirical results show that the original CTM algorithm provides accurate regression estimates. However, it lacks some key features found in other statistical methods:

(1) Piecewise-linear vs. piecewise-constant approximation: The original CTM algorithm uses a piecewise constant regression surface, which is not an accurate representation scheme for smooth functions. Better accuracy could be achieved using, for example, a piecewise-linear fit.

(2) Control of model complexity: Up to this point, there has been little understanding of how model complexity in the CTM algorithm is adjusted. Interpreting the unit update equations as a kernel regression estimate[Mulier and Cherkassky, 1995] gives some insight by interpreting the neighborhood width as a kernel span in the topological map space. The neighborhood decrease schedule then plays a key role in the control of complexity.

(3) Global variable selection : Global variable selection is a popular statistical technique commonly used (in linear regression) to reduce the number of predictor variables by discarding low-important variables. However, the original CTM algorithm provides no information about variable importance, since it gives all variables equal strength in the clustering step. Since the CTM algorithm performs self-organization (clustering) based on Euclidean distance in the space of the predictor variables, the method is sensitive to predictor scaling. Hence, variable selection can be implemented in CTM *indirectly* via adaptive scaling of predictor variables during training. This scaling makes the method adaptive, since the quality of the fit in the response variable affects the positioning of map units in the predictor space. This feature is important for high dimensional problems where training samples are sparse, since local parametric methods require dense samples and global parametric methods introduce bias. Hence, adaptive methods are the only practical alternative.

(4) Batch vs flow-through implementation. The original CTM (as most neural network methods) is a flow-through algorithm, where samples are processed one at a time. Even though flow-through methods may be desirable in some applications (i.e. control), they are generally inferior to batch methods (that use all available training samples) commonly used in statistics for estimation, both in terms of computational speed and estimation accuracy. In particular, the results of modeling using flow-through methods may depend on the (heuristic) choice of the learning rate schedule[Mulier and Cherkassky, 1995]. Hence, the batch version of CTM has been developed in [Cherkassky and Mulier, 1994].

These deficiencies in the original CTM algorithm can be overcome using statistically-motivated improvements, as detailed next. These improvements have been incorporated in the latest version of CTM included in XTAL package.

Local linear regression estimates

The original CTM algorithm, can be modified to provide piecewise linear approximation for the regression surface. Using locally weighted multiple linear regression, the neighborhood function would be used to weight the observations and zero and first order terms can be estimated for each unit. The regression estimate for each unit is *global*, but local samples are given more weight according to the neighborhood function.

Using cross-validation to select final neighborhood width

The complexity of a model produced by the SOM or CTM method is determined by the final neighborhood width used in the training algorithm [Mulier and Cherkassky, 1995]. In other words, the final neighborhood width is a smoothing (regularization) parameter of the CTM method. In order to estimate the correct amount of smoothing from the data, one commonly uses cross-validation. In this procedure, a series of regression estimates are determined based on portions of the training data, and the sum of squares error is measured using the remaining validation samples. For each regression estimate, a different subset of validation samples are chosen from the original training set, so that *each training sample* is used *exactly once* for validation. The final measure of generalization error is the average of the sum of squares error. Because of the computational advantages, *leave-one-out* cross validation was chosen for CTM. In this case, each sample is systematically removed from the training set to be used as the validation set.

Variable selection via adaptive sensitivity scaling

The CTM algorithm effectively applies the Kohonen SOM in the space of the predictor variables to determine the unit locations. The SOM is essentially a clustering technique, which is sensitive to the particular distance scaling used. For CTM, a heuristic scaling technique can be implemented based on the *sensitivity* of the linear fits for each Voronoi region. We would like to adjust the scales of the predictor variables so that those variables that are most important in the regression are given more weight in the distance calculation. The sensitivity of a variable on the regression surface can be determined locally for each Voronoi region. These *local sensitivities can be averaged* over the Voronoi regions in order to judge the *global importance* of a variable on the whole regression estimate. Since new regression estimates are given with each iteration of the CTM algorithm, this scaling can be done *adaptively*, i.e. the scaling/variable importance parameters are used in the distance calculation when clustering the data according to the Voronoi regions. This effectively causes more units to be placed along variable axis which have larger average sensitivity.

Implementation details

In order to make a regression method practical, there are a number of implementation issues that must be addressed. The Batch CTM software package provides some additional features that improve the quality of the results and improve the ease of use. For example, in interpolation (no noise) problems with a small number of samples, model selection based on

cross-validation provides an overly smooth estimate. For these problems it may be advantageous to do model selection based on the mean squared error on the training set. The package allows the user to select either cross-validation or training set error or a mixture of the two to perform model selection. This effectively provides the user control over a complexity penalty. The package is also capable of automatically estimating the number of units of the map based on the error (cross-validation or training set) score. This is done using a heuristic search strategy which first exhaustively tries maps with 1, 2, 3, 4, and 5 units per dimension to find the number of units which provide the lowest error. It then starts with a map with a large number of units (typically 1000), and then decreases this number based on the width of the neighborhood which gives the best error. If the neighborhood width turns out to be large, then it reduces the number of units proportionately. This procedure is iterated 15 times. The final model selected is the one with the lowest cross-validation error encountered in the entire search. This procedure provides good automatic selection of the number of units when the user does not wish to enter a specific number. When used with XTAL, the user supplies the model complexity penalty, an integer from 0-9 (max. smoothing) and the dimensionality of the map.

REFERENCES

Breiman, L., Friedman, J.H., Olshen, R.A., and C.J. Stone , Classification and Regression Trees, Wadsworth, Belmont, CA,1984.

Cherkassky, V. and H. Lari-Najafi, Constrained topological Mapping for Nonparametric Regression Analysis, Neural Networks, Pergamon, v.4, 27-40. 1991.

Cherkassky, V., J.H. Friedman and H. Wechsler (Eds), From Statistics to Neural Networks: Theory and Pattern Recognition Applications, NATO ASI Series F, v.136, Springer, 1994.

Cherkassky, V. and F. Mulier, Application of self-organizing maps to regression problems, JASA, submitted, 1994

Craven, P. and Wahba, G., Smoothing noisy data with spline functions, Numerische Mathematik, 31, 377-403, 1979.

Friedman, J.H. and W. Stuetzle, Projection Pursuit Regression, JASA, v.76, 817-823, 1981.

Friedman, J. H., SMART user's guide, TR no.1, Dept of Statistics, Stanford University, 1984

Friedman, J.H., Multivariate Adaptive Regression Splines, (with discussion), Ann. Statist., 19, 1-141, 1991.

Friedman, J.H., An Overview of predictive learning and function approximation, in Cherkassky, V., J.H. Friedman and H. Wechsler (Eds), From Statistics to Neural Networks: Theory and Pattern Recognition Applications, NATO ASI Series F, v.136,Springer, 1994.

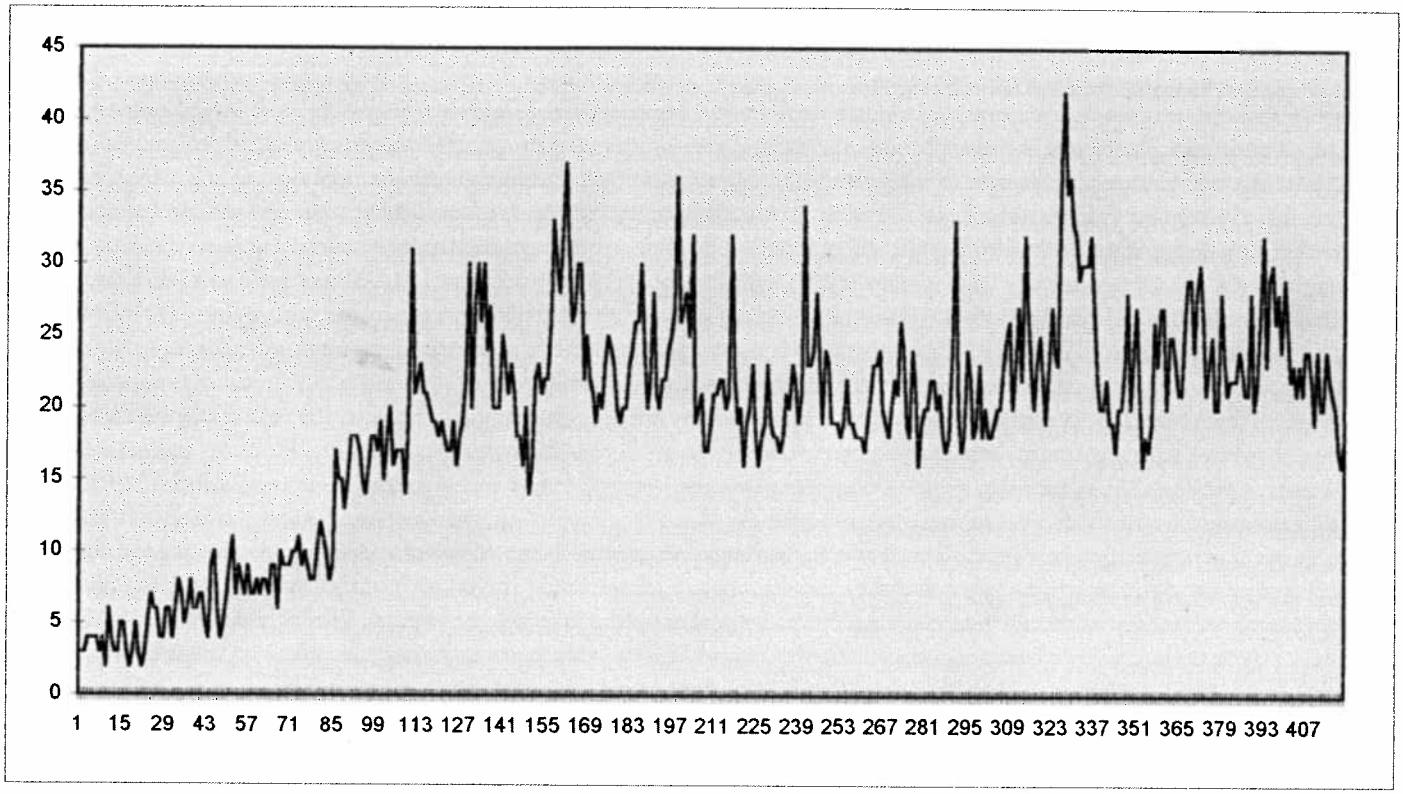
Kohonen, T., Self-organization and Associative Memory, Berlin: Springer-Verlag, 1984.
Masters, T., Practical Neural Network Recipes in C++, Academic Press, 1993.

Mulier, F. and V. Cherkassky, Self-organization as an iterative smoothing process, Neural Computation, 7, 1141-1153, 1995

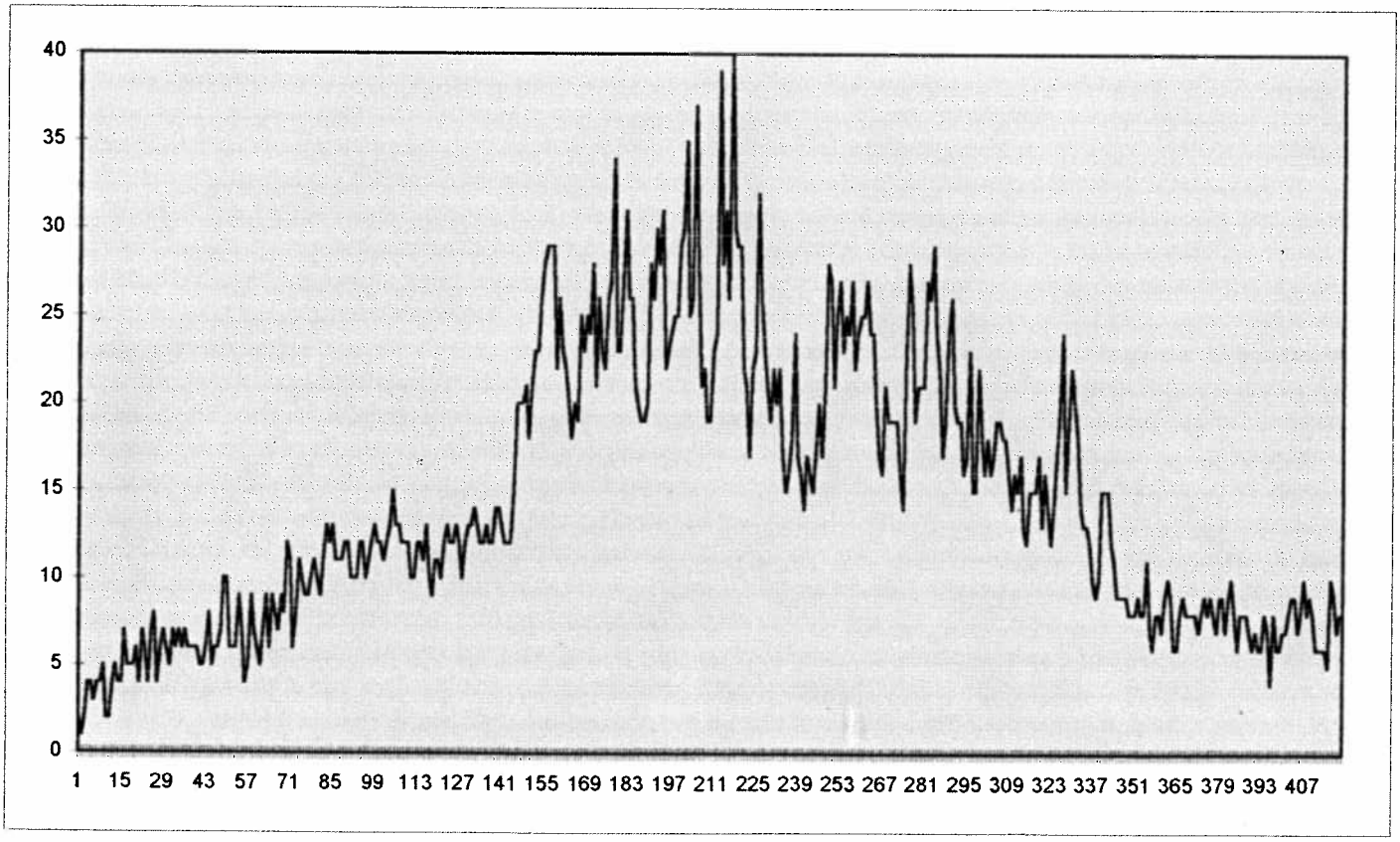
APPENDIX 1

Freeway occupancy data (at central station) used in this study

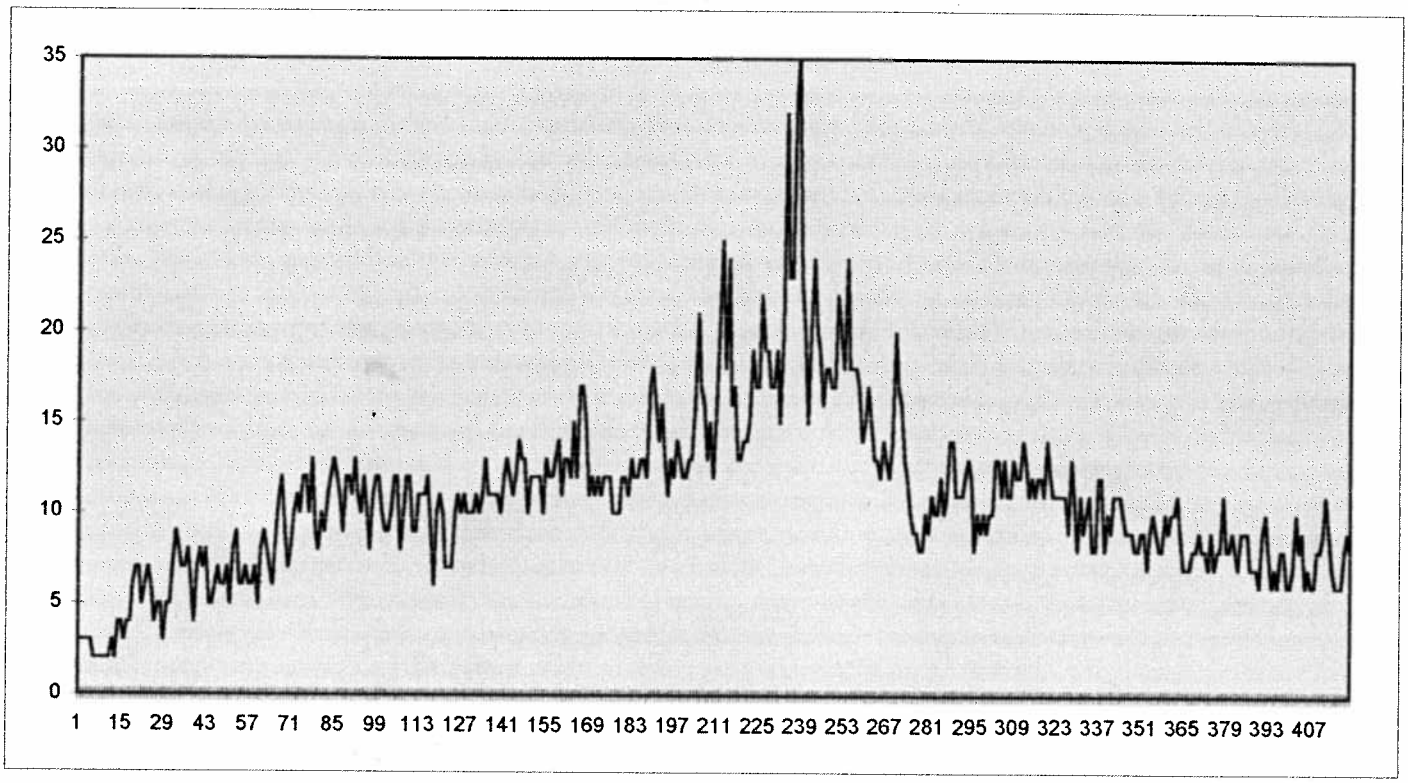
4.11.95.41.43



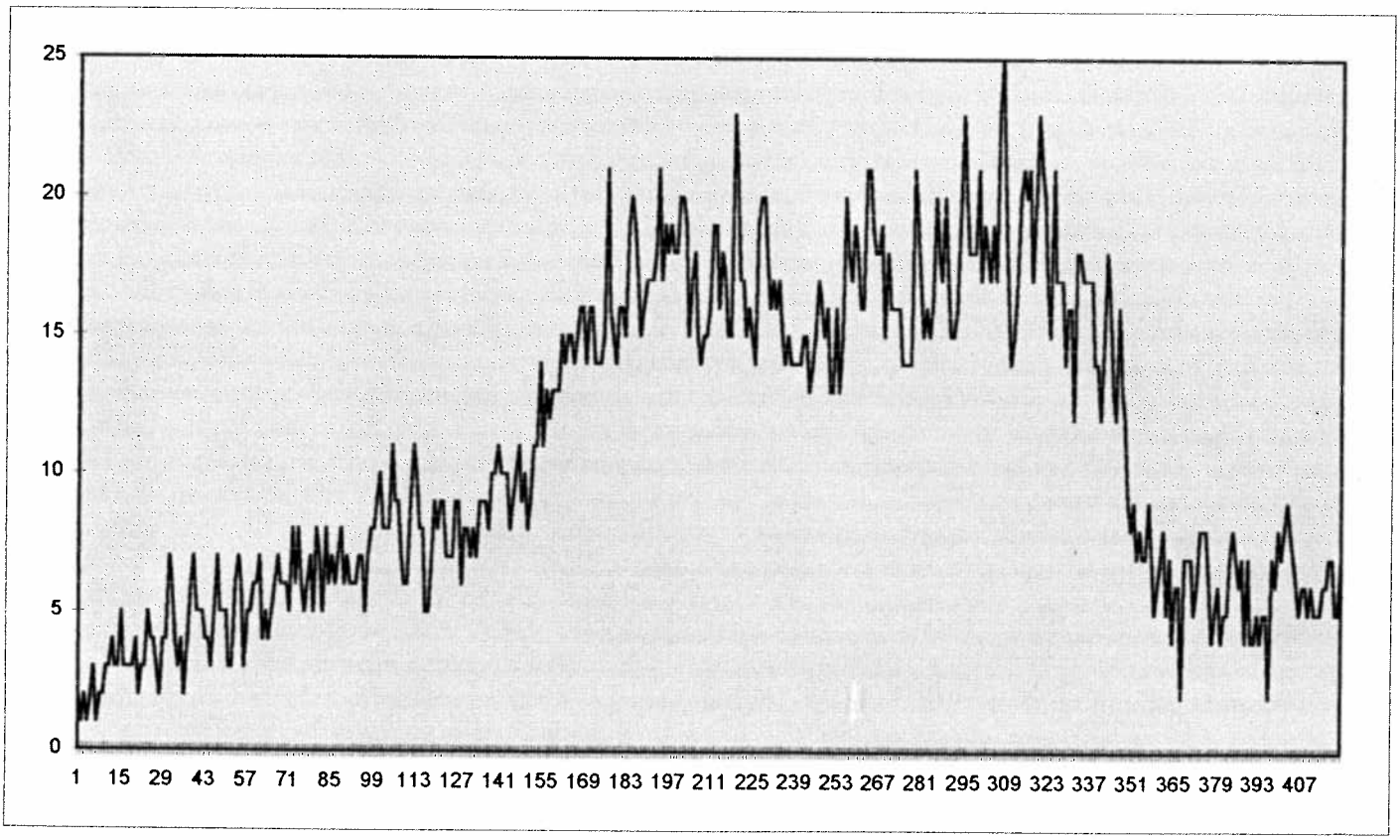
4.12.95.41.43



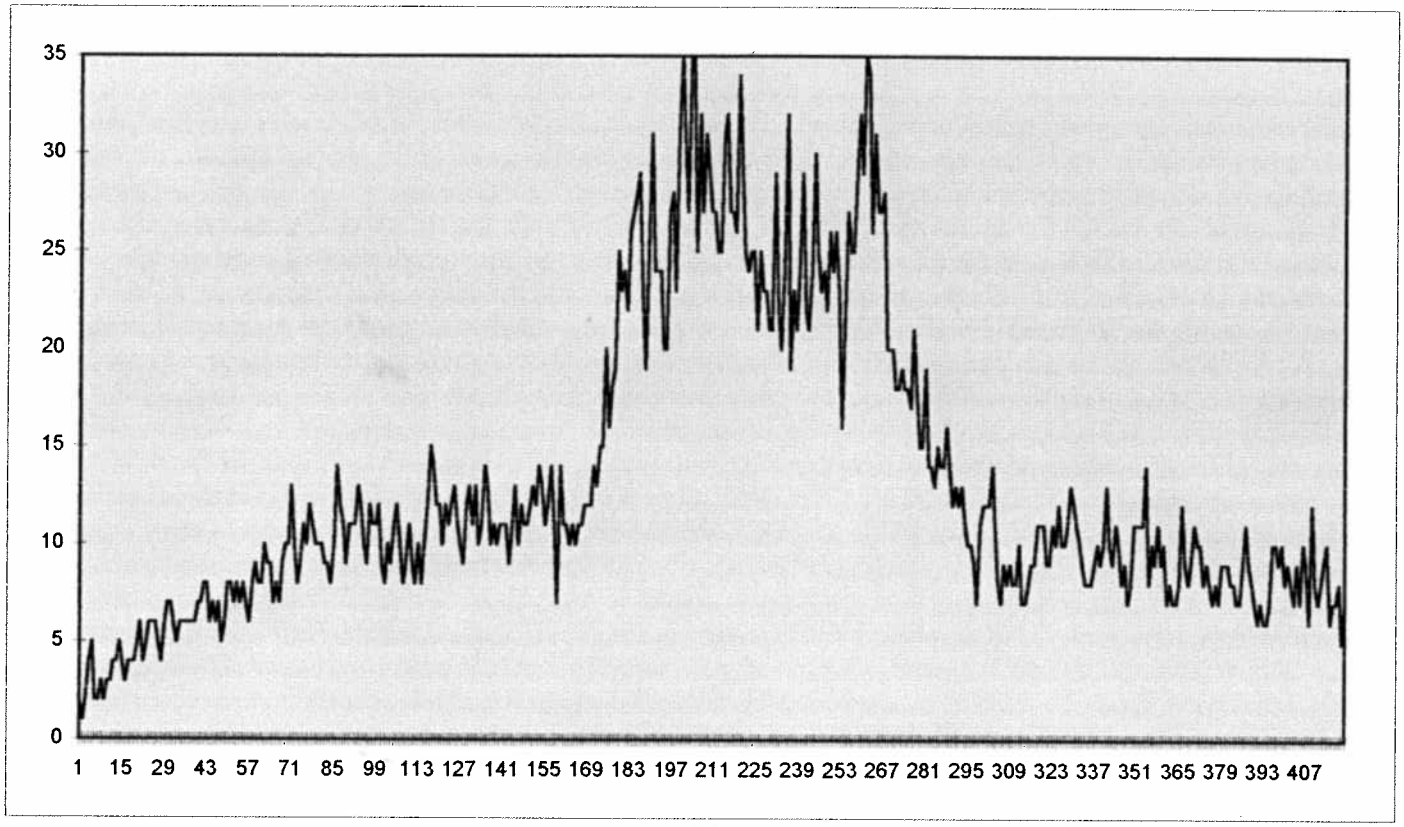
4.26.95.41.43



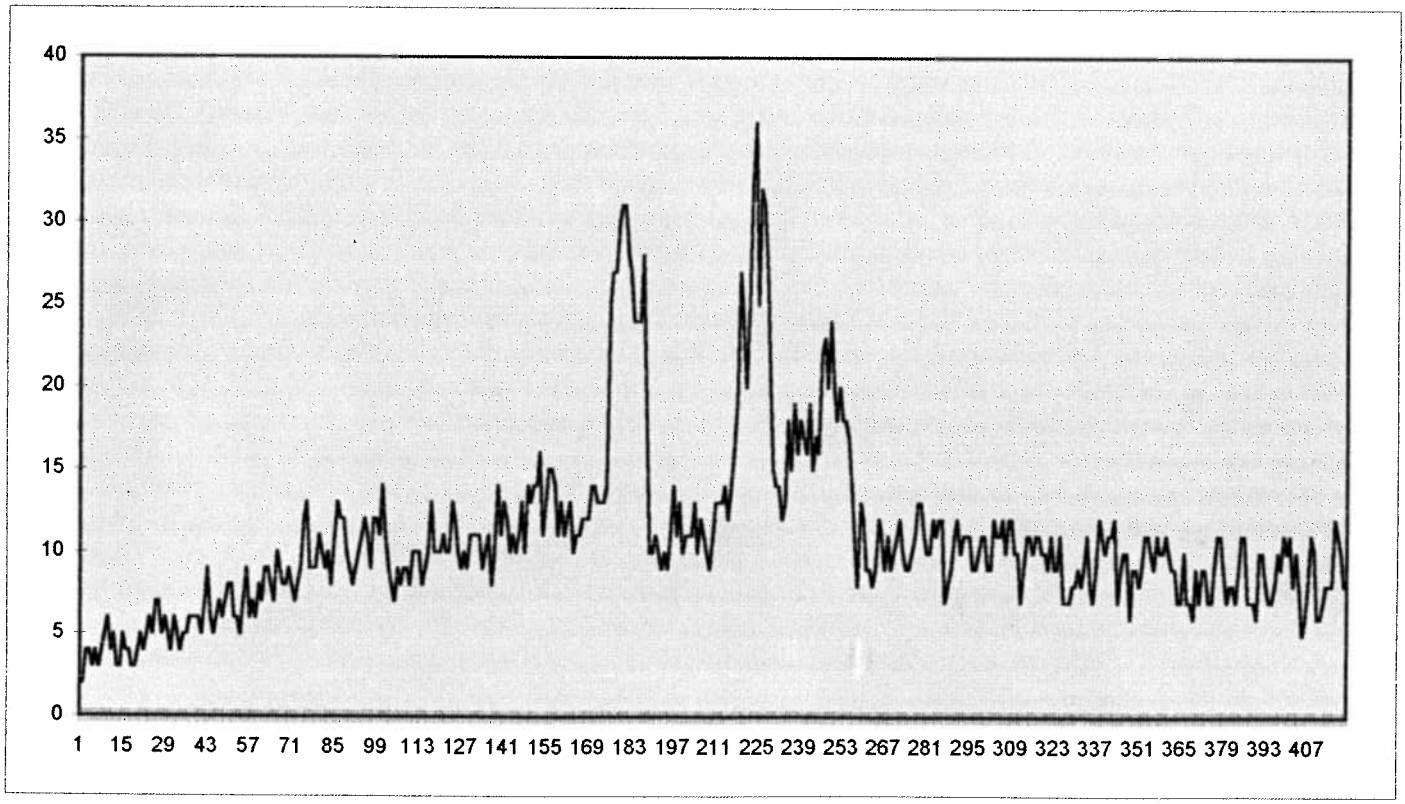
4.26.95.55.57



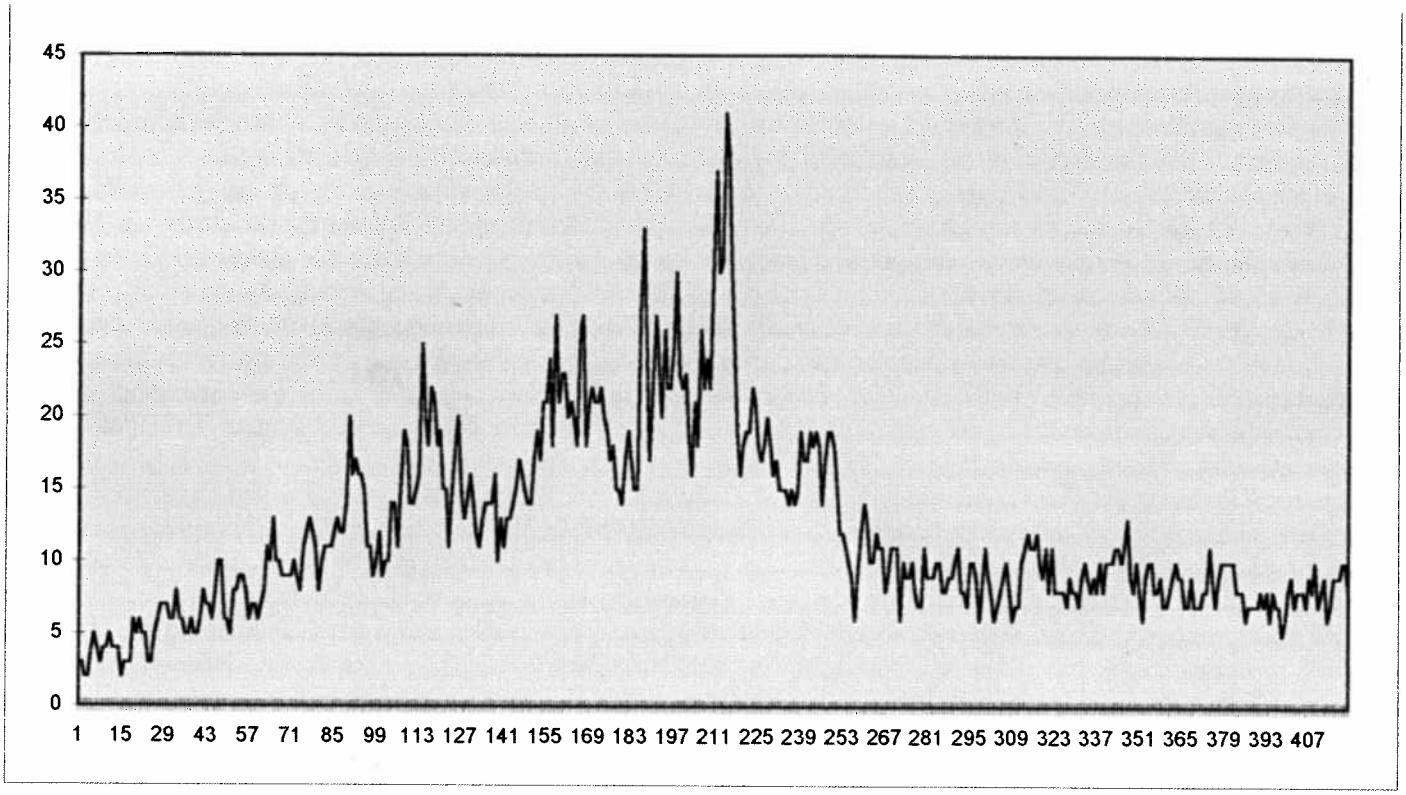
5.3.95.41.43



5.4.95.41.43

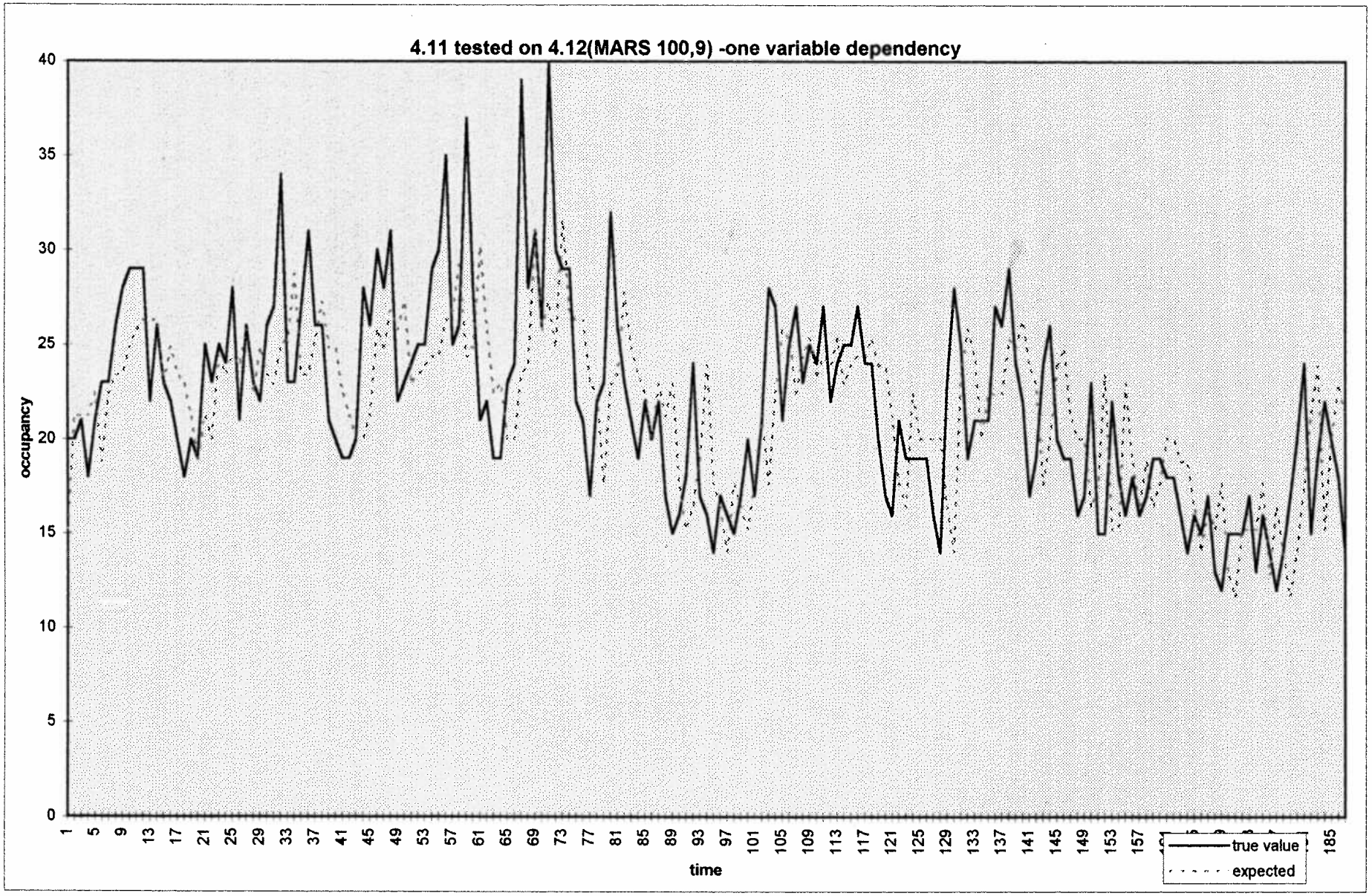


6.6.95.41.43

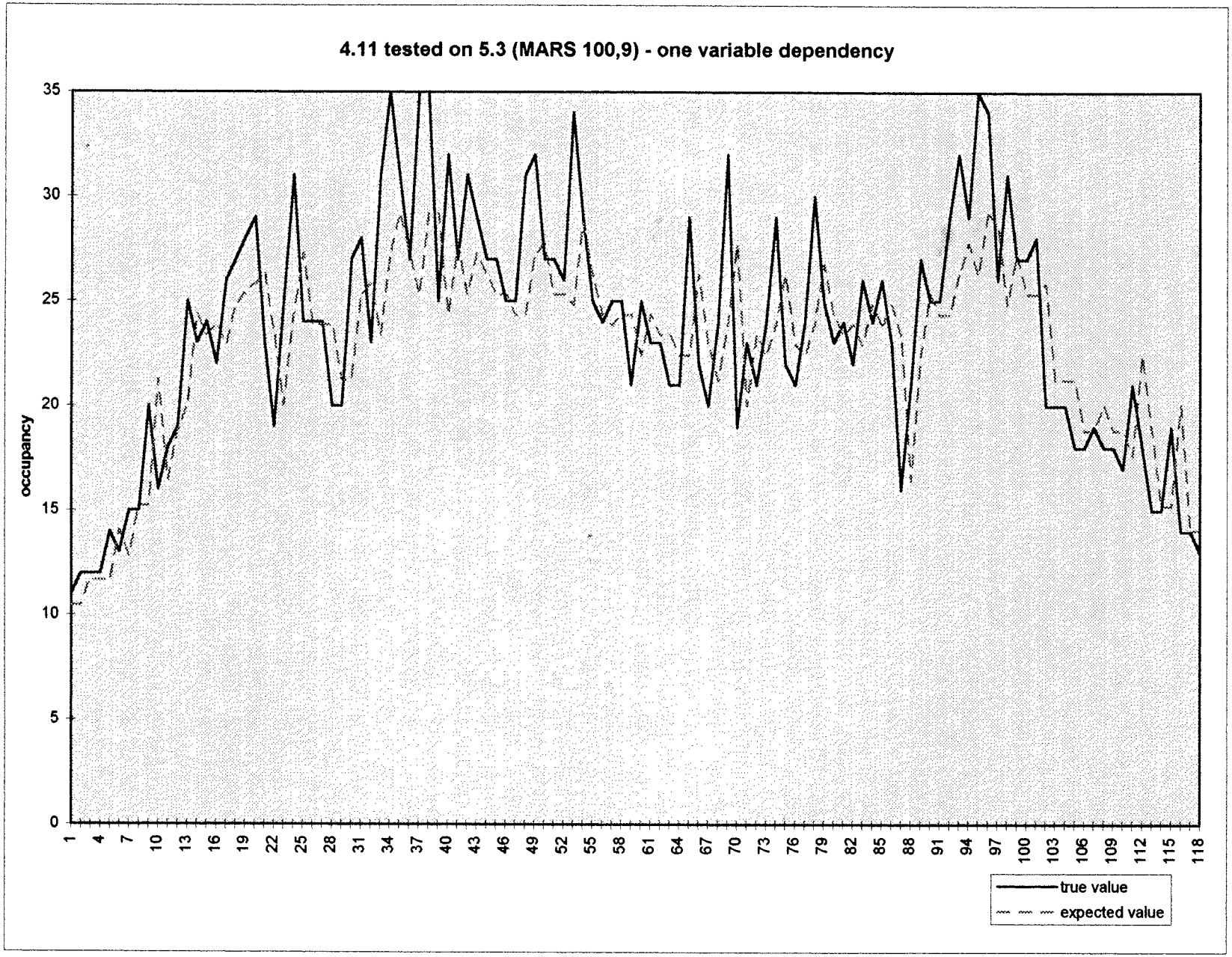

45
40
35
30
25
20
15
10
5
0
1 15 29 43 57 71 85 99 113 127 141 155 169 183 197 211 225 239 253 267 281 295 309 323 337 351 365 379 393 407

APPENDIX 2

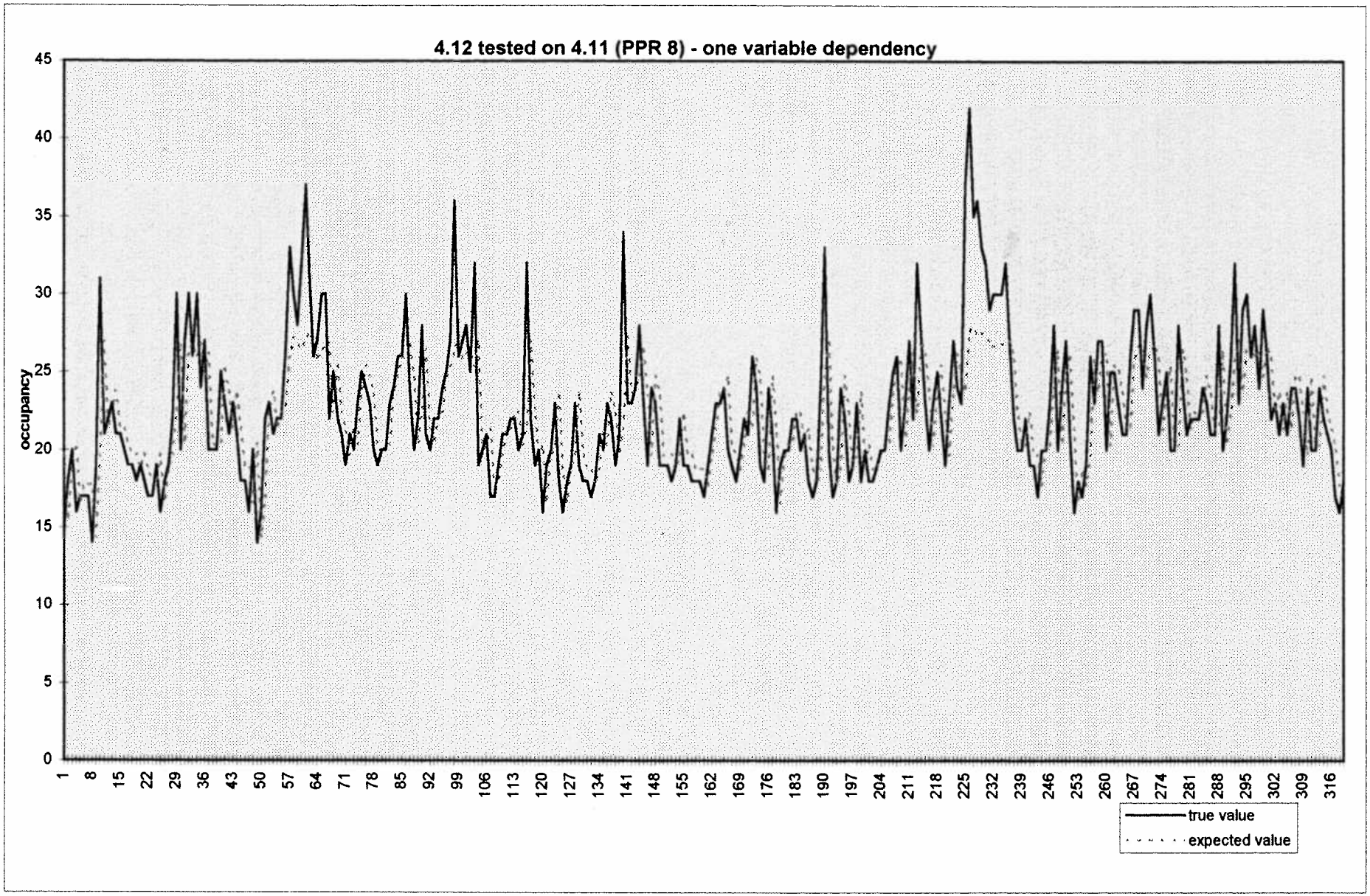
Sample time series generated by prediction models, shown along with true occupancy observations (test data).



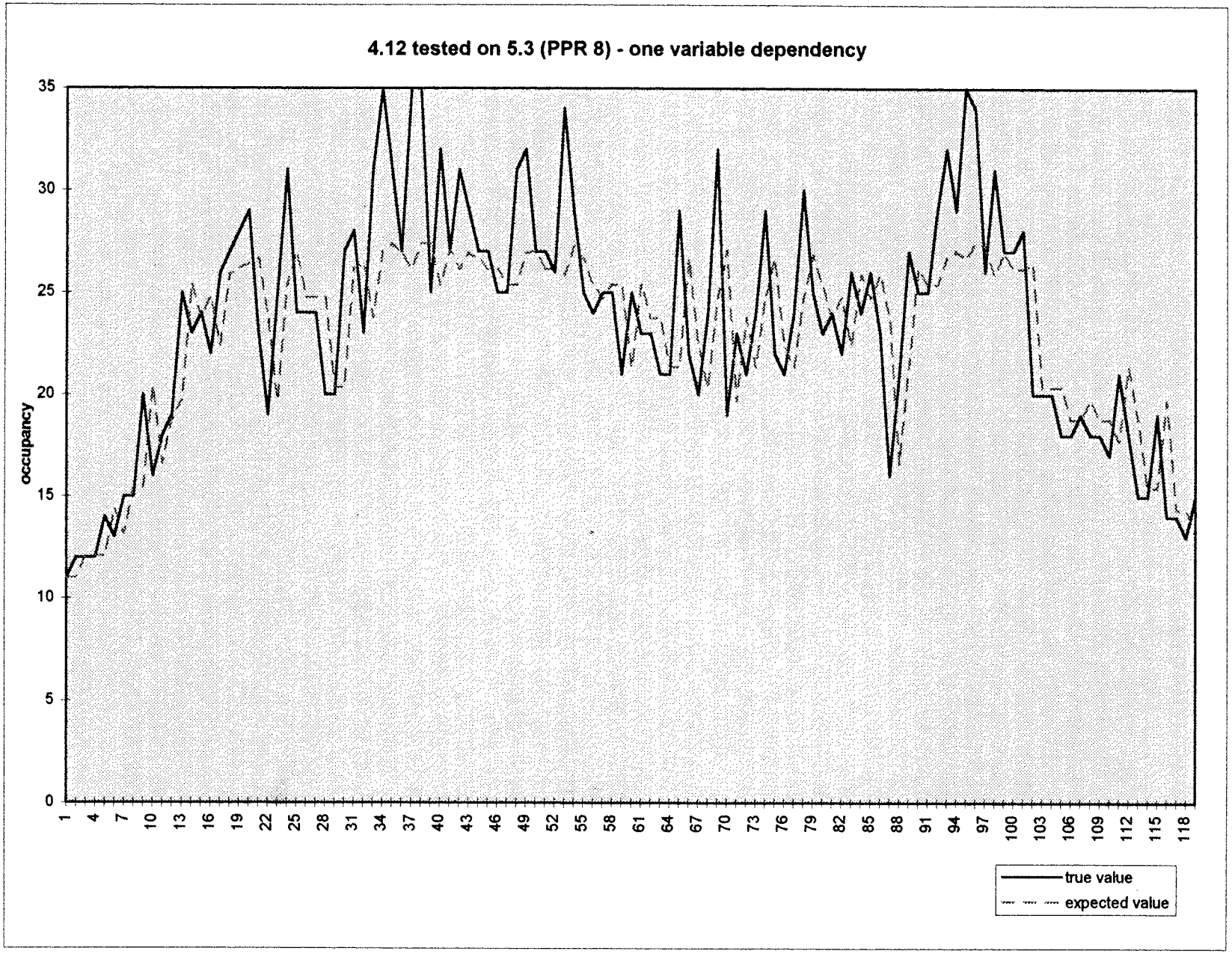
rms	3.8
max	15.17



RMS	3.76
MAX	9.69



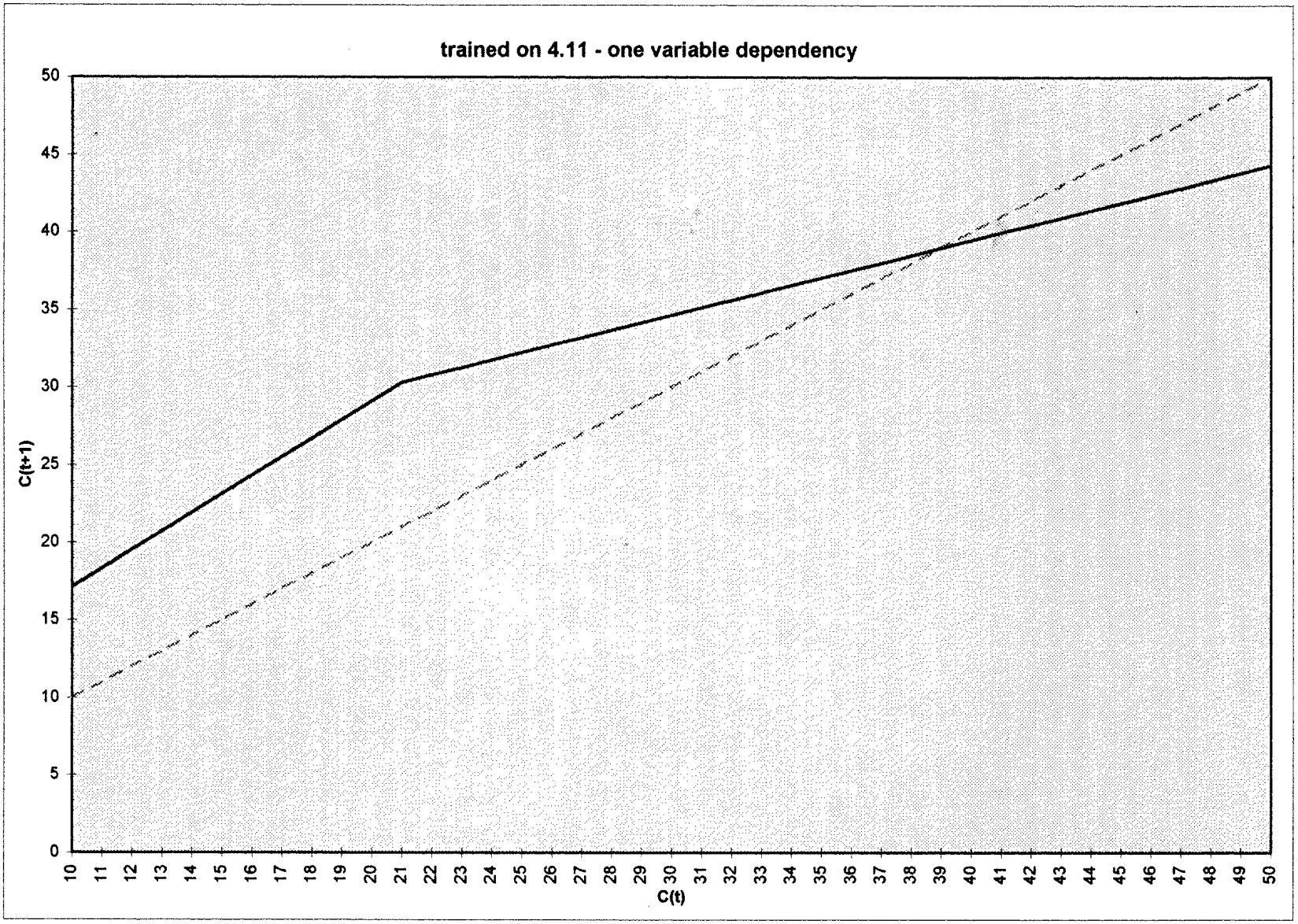
rms	3.65
max	14.4



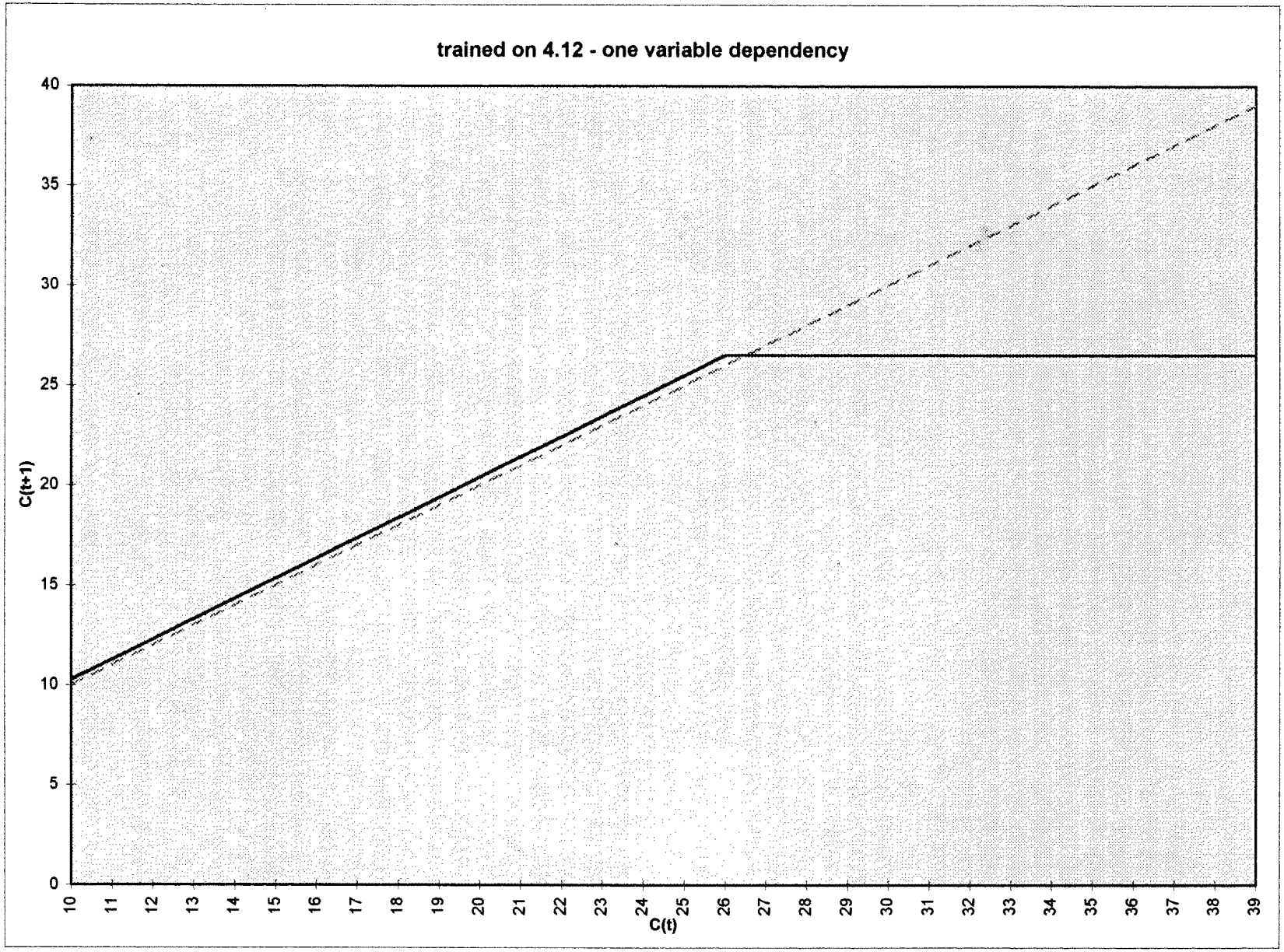
RMS	3.71
MAX	8.82

APPENDIX 3

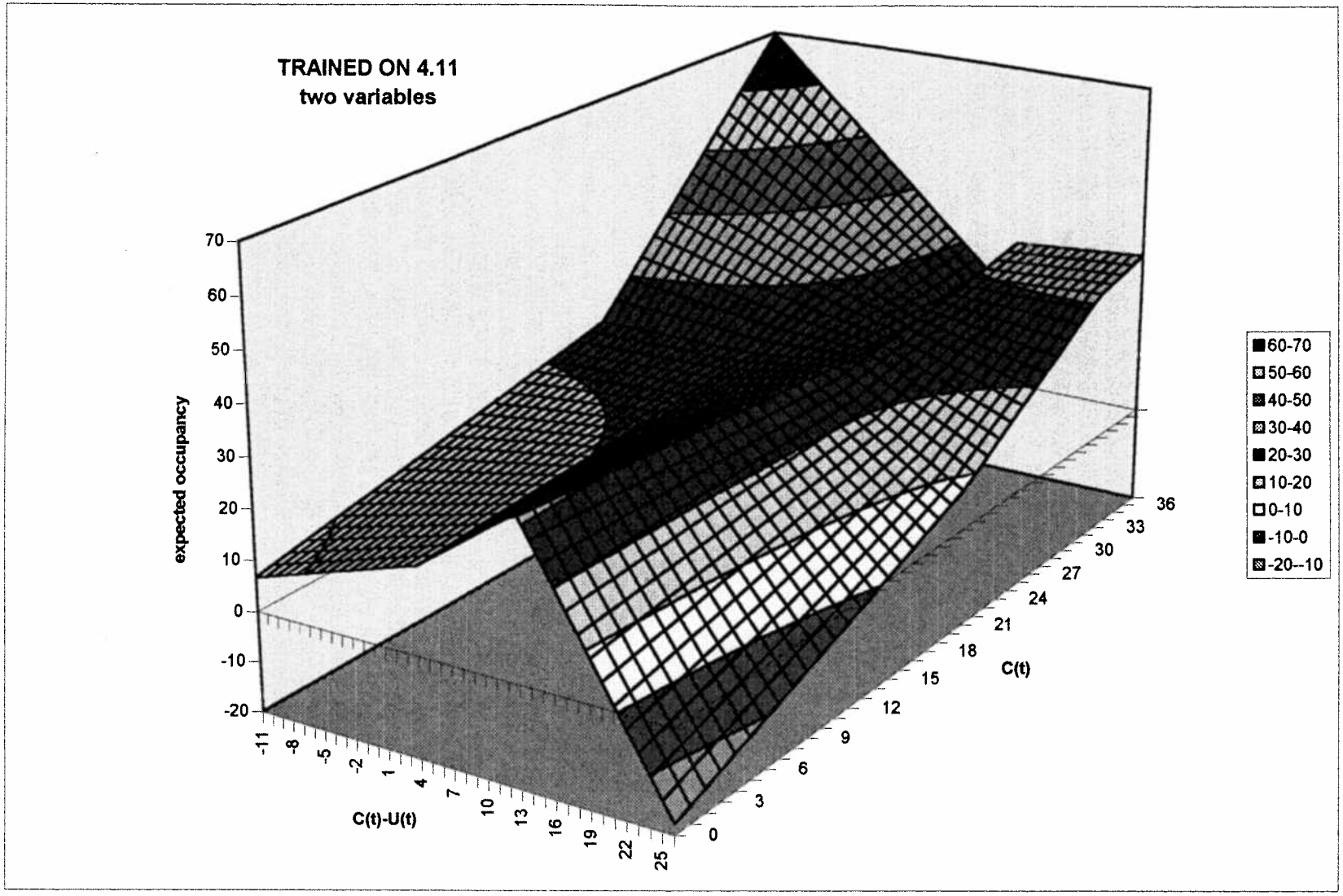
Plots of nonlinear prediction models



$$25 + 0.482(x - 10)^{+} - 0.714(x - 21)^{-}$$

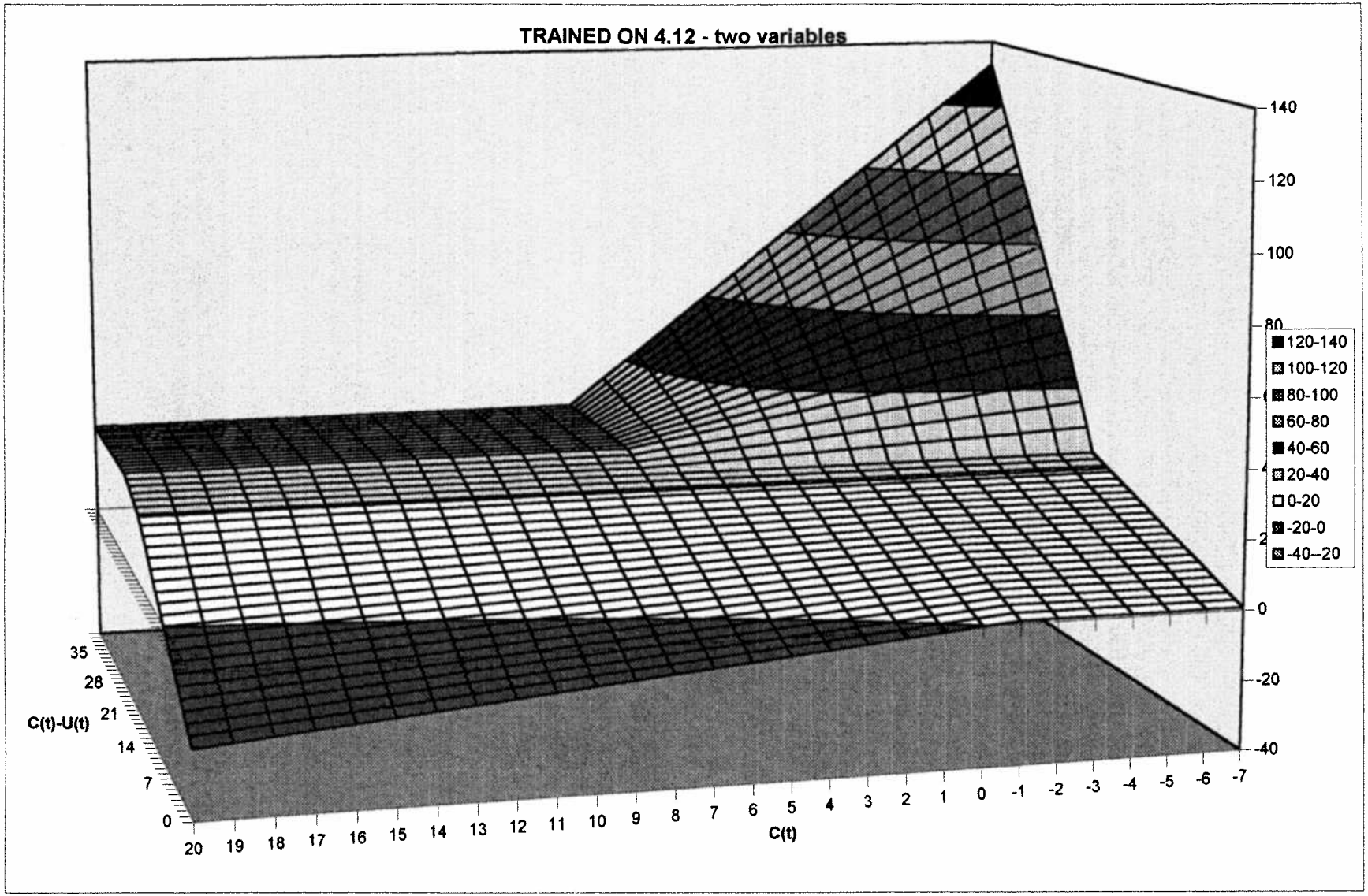

trained on 4.12 - one variable dependency
C(t+1)
C(t)
0
5
10
15
20
25
30
35
40
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39

$$26.5 - 1.015(x - 26)^{-}$$



$$22.94 - 0.0667(x_1 - 21)^{-}(x_2 + 11)^{-} - 0.1428(x_1 - 33)^{-}(x_2 - 13)^{+} + 0.1453(x_1 - 23)^{+}(x_2 - 12)^{-}$$
$$- 0.308(x_1 - 23)^{-}(x_2 - 12)^{-} - 0.554(x_1 - 21)^{+}(x - 14)^{+} + 0.565(x_2 - 13)^{+}(x_1 - 21)^{+} + 0.0284(x_1 - 33)^{-}(x_2 - 7)^{+}$$

$$x_1 = C(t) - U(t); \quad x_2 = C(t)$$



$$26.26 - 0.977(x_1 - 26)^{-} + 0.454(x_1 - 22)^{+}(x_2 - 6)^{-} - 0.0503(x_1 - 20)^{-}(x_2 + 1)^{+}$$

$$x_1 = C(t); \qquad x_2 = C(t) - U(t)$$