Proceedings of the 2nd U.S.-Japan Joint
Seminar on Statistical Time Series Analysis
Honolulu, Hawaii, January 25-29, 1993

A Monte Carlo Filtering and Smoothing Method for Non-Gaussian Nonlinear State Space Models

Genshiro Kitagawa
The Institute of Statistical Mathematics
4-6-7 Minami-Azabu, Minato-ku, Tokyo 106 JAPAN

January 9, 1993

Abstract

A new algorithm of filtering and smoothing for non-Gaussian nonlinear state space models is given. The algorithm is based on a Monte Carlo method and approximate each conditional probability density function by many of its realizations. The significant merit of the present algorithm is that it can applied to any nonlinear non-Gaussian higher dimensional state space models if the dimension of the system noise and the observational noise are low.

1. Introduction

A Monte Carlo method for non-Gaussian nonlinear filtering and smoothing is shown here. In this method, each distribution is expressed by many of its realization, and the movement of each particle is simulated by using the assumed model.

In recent years, the use of non-Gaussian or nonlinear models becomes very popular in time series analysis. For example, West, Harrison and Migon (1986) considered generalized dynamic linear model. Kitagawa (1987) directly generalized the state space model to the case where either of the system noise and the observational noise are non-Gaussian. In the method, recursive formulas for filtering and smoothing are derived and many nonstandard problems in time series such as the abrupt changes of parameters of the model, outliers and skewed distributions can be properly handled with this model. The method can be easily generalized to discrete distribution models (Kitagawa, 1987) and nonlinear models (Kitagawa, 1991).

On the other hand, the difficulty with this numerical method is that it requires intensive use of the computer both in memory and CPU time. To mitigate this difficulty,

Gaussian-sum approximation was used in applying for the non-Gaussian seasonal adjustment where typically 13 dimensional state space model was required. Considerable work has been done on the refinement of the numerical algorithm based on computationally more efficient integration algorithm or a Monte Carlo method.

In this paper, we will present a direct Monte Carlo method for filtering and smoothing. The algorithm is based on the approximation of each conditional probability density function by many of its realizations. The difference of the present algorithm from other Monte Carlo method is that here we use the Monte Carlo method for the entire filtering and smoothing not only for the numerical integration. The significant merit of the proposed algorithm is that it can applied to any nonlinear non-Gaussian higher dimensional state space models if the dimension of the system noise and the observational noise are low.

2. Non-Gaussian Nonlinear State Space Model and Filtering

Assume that the time series y_n is obtained by the following non-Gaussian nonlinear state space model

$$x_n = f(x_{n-1}, v_n) \tag{1}$$

$$y_n = h(x_n, w_n), \tag{2}$$

where x_n is a k-dimensional state vector, v_n and w_n are ℓ-dimensional and 1-dimensional white noise sequences with densities $q(v)$ and $r(w)$, respectively. f and h are possibly nonlinear functions $\mathrm{R}^k \times \mathrm{R}^\ell \to \mathrm{R}^k$ and $\mathrm{R}^k \times \mathrm{R} \to \mathrm{R}$, respectively. It is also assumed that $y = h(x, w)$ has an implicit function such that $w = g(x, y)$. The initial state vector x_0 is distributed according to the density $p(x_0)$.

This type of state space model contains a broad class of models. Some examples follows.

1. If $f(x, v) = Fx + Gv$ and $h(x, w) = Hx + w$, then $g(y, x) = y - Hx$ and we obtain an ordinary state space model.

2. If $f(x, v) = f_1(x) + v$ and $h(x, w) = h_1(x) + w$, then $g(y, x) = y - h_1(x)$ and we obtain a nonlinear state space model.

3. If $f(x,v) = x + v$ and $h(x,w) = e^x w$, then $g(y,x) = ye^{-x}$ and we obtain a model for time series with time-varying variance.

The problem of the state estimation is to evaluate the conditional density $p(x_n|Y_t)$, where Y_t is the σ-algebra generated from $\{y_1, \cdots, y_t\}$. In the following development of the filtering and the smoothing algorithms, each conditional density is expressed by using its realizations.

2.1 One Step Ahead Prediction

Assume that $\{s_1, ..., s_m\}$ and $\{v_1, ..., v_m\}$ are independent realizations of $p(x_{n-1}|Y_{n-1})$ and $q(v)$, respectively. Namely,

$$\begin{aligned} \{s_1, ..., s_m\} &\sim p(x_{n-1}|Y_{n-1}) \\ \{v_1, ..., v_m\} &\sim q(v). \end{aligned} \tag{3}$$

Here, if t_i is defined by

$$t_i = f(s_i, v_i), \tag{4}$$

then, obviously, $t_1, \cdots, t_m$ are distributed as $p(x_n|Y_{n-1})$.

2.2 Filtering

Given the observation y_n and the realizations of the predictor, $\{t_1, \cdots, t_m\} \sim p(x_n|Y_{n-1})$, compute $\alpha_j = p(y_n|t_j) = r(g(y_n, t_j))$ for $j = 1, ..., m$. Next, obtain $s_1, \cdots, s_m$ by the resampling of $t_1, ..., t_m$ with the probabilities proportional to $\alpha_1, ..., \alpha_m$, respectively. Namely, define s_j by

$$s_j = \begin{cases} t_i & \text{with probability } \alpha_1/(\alpha_1 + \cdots + \alpha_m) \\ \vdots & \\ t_m & \text{with probability } \alpha_m/(\alpha_1 + \cdots + \alpha_m) \end{cases} \tag{5}$$

Then $\{s_1, \cdots, s_m\}$ is distributed as $p(x_n|Y_n)$.

This can be verified as follows. When m independent realizations $t_1, \cdots, t_m$ from $p(x_n|Y_{n-1})$ are given, the distribution function of $p(x_n|Y_{n-1})$ is approximated by the empirical distribution function

$$P_n(x) = \frac{1}{m} \sum_{i=1}^{m} I(x, t_i), \tag{6}$$

where $I(x, t_i)$ is the function defined by $I(x, a) = 0$ if $x < a$ and $I(x, a) = 1$ otherwise. This means that $p(x_n|Y_{n-1})$ is approximated by the probability function

$$Pr(X_n = t_i|Y_{n-1}) = \frac{1}{m} \qquad \text{for } i = 1, \cdots, m. \tag{7}$$

Then, given the observation, y_n, the posterior probability is obtained by

$$\begin{aligned} Pr(X_n = t_i|Y_n) &= Pr(X_n = t_i|Y_{n-1}, y_n) \\ &= \frac{p(X_n = t_i, y_n|Y_{n-1})}{p(y_n|Y_{n-1})} \\ &= \frac{p(y_n|X_n = t_i)Pr(X_n = t_i|Y_{n-1})}{\sum_{j=1}^m p(y_n|X_n = t_j)Pr(X_n = t_j|Y_{n-1})} \\ &= \frac{\alpha_i \cdot \frac{1}{m}}{\sum_{j=1}^m \alpha_j \cdot \frac{1}{m}} = \frac{\alpha_i}{\sum_{j=1}^m \alpha_j}. \end{aligned} \tag{8}$$

Therefore m independent realizations $\{s_1, ..., s_m\}$ of $p(x_n|Y_n)$ is obtained by the resampling of $\{t_1, ..., t_m\}$ with

$$Pr(s_i = t_j|Y_n) = \frac{\alpha_j}{\alpha_1 + ... + \alpha_m} \qquad \text{for } i = 1, ..., m. \tag{9}$$

This means that the posterior distribution function

$$\frac{1}{\sum_{j=1}^m \alpha_j} \sum_{i=1}^m \alpha_i I(x, t_i) \tag{10}$$

is approximated by

$$\frac{1}{m} \sum_{i=1}^m I(x, s_i). \tag{11}$$

2.3 Likelihood of the Model

Given the observations $y_1, ..., y_N$, the likelihood of the parameter θ of the model is obtained by

$$L(\theta) = p(y_1, ..., y_N|\theta) = \prod_{n=1}^N p(y_n|y_1, ..., y_{n-1}, \theta) = \prod_{n=1}^N p(y_n|Y_{n-1}). \tag{12}$$

Therefore, since $p(y_n|Y_{n-1}) = \frac{1}{m}\sum_{j=1}^m \alpha_j$, the log-likelihood is given by

$$\begin{aligned} \ell(\theta) &= \sum_{n=1}^N \log p(y_n|Y_{n-1}) \\ &= \sum_{n=1}^N \log \left(\sum_{j=1}^m \alpha_j \right) - N \log m \\ &= \sum_{n=1}^N \log \left(\sum_{j=1}^m p(y_n|t_j) \right) - N \log m. \end{aligned} \tag{13}$$

2.4 An Algorithm for Filtering

Summarizing the preceding subsections, we obtain the following algorithm for filtering:

1. *Determine k, the number of realizations used for the approximation of each distribution.*

2. *Generate a k-dimensional random number $s_j^{(0)} \sim p_0(x)$ for $j = 1, ..., m$.*

3. *Repeat the following steps for $n = 1, ..., N$*

   (a) *Generate a ℓ-dimensional random number $v_j^{(n)} \sim q(v)$ for $j = 1, ..., m$*

   (b) *Compute $t_j^{(n)} = f(s_j^{(n-1)}, v_j^{(n)})$ for $j = 1, ..., m$.*

   (c) *Compute $\alpha_j^{(n)} = r(g(y_n, t_j^{(n)}))$ for $j = 1, ..., m$.*

   (d) *Generate $s_j^{(n)} \sim (\sum_{j=1}^m \alpha_j^{(n)})^{-1} \sum_{j=1}^m \alpha_j^{(n)} I(x, t_j^{(n)})$ for $j = 1, ..., m$ by the resampling of $t_1^{(n)}, ..., t_m^{(n)}$.*

2.5 An Illustrative Example

Consider a nonlinear one dimensional state space model

$$x_n = x_{n-1} + v_n \tag{14}$$

$$y_n = x_n + w_n, \tag{15}$$

where v_n and w_n are white noises distributed as the Cauchy distribution, $C(0, 0.01)$ and the normal distribution, $N(0, 1)$, respectively. Here, to be specific, we assume that $p(x_{n-1}|Y_{n-1})$ is $N(0, 1)$ and consider the predictive and the filter distributions, $p(x_n|Y_{n-1})$ and $p(x_n|Y_n)$. The panels (a) and (c) in Figure 1 show the density functions of $p(x_{n-1}|Y_{n-1})$ and $q(v_n)$ and their 100 realizations. In (b) and (d), the bold curves show the empirical distribution functions defined by 100 realizations shown in (a) and (c), respectively. The fine curves show the true distribution functions. The panel (e) shows the 100 realizations of $p(x_n|Y_{n-1})$ directly computed by (14). On the other hand, the curve in (e) shows the "exact" density function obtained by numerical convolution of the densities. The panel (f) shows the corresponding distribution function and the empirical distribution function, respectively. (g) shows the "exact" filtered density $p(x_n|Y_n)$. 100 realizations in (g) are the same as the one in (e). However, the probability of each realization is not $1/m$ but is α_j, and the corresponding distribution function becomes the one shown in (h). The panel (i) shows 100 realizations obtained by the resampling of the data shown in (g) according to the probability given by (5). It can be seen that the empirical

distribution function shown in (j) resembles to the one in (i) and is close to the "exact" one shown by fine curve.

3. Smoothing

3.1 Smoothing by Storing the State Vector

In this section, we assume that each density function is approximated by an empirical distribution function and we use the following notation,

$$p(t_j^{(1)}, \cdots, t_j^{(n)}|Y_\ell) \;=\; Pr(X_1 = t_j^{(1)}, \cdots, X_n = t_j^{(n)}|Y_\ell). \tag{16}$$

Assume that $(s_j^{(1)}, ..., s_j^{(n-1)}) \sim p(s_j^{(n)}, ..., s_j^{(n-1)}|Y_{n-1}) = 1/m$ and $v_j^{(n)} \sim q(v)$, and define $(t_j^{(1)}, ..., t_j^{(n)})$ by

$$t_j^{(i)} = \begin{cases} s_j^{(i)} & \text{for } i = 1, ..., n-1 \\ f(s_j^{(n-1)}, v_j^i) & \text{for } i = n. \end{cases} \tag{17}$$

Then $(t_j^{(1)}, ..., t_j^{(n)})$ can be considered as realizations from $p(t_j^{(1)}, ..., t_j^{(n)}|Y_n)$.

Next, given an observation y_n, $p(t_j^{(1)}, \cdots, t_j^{(n)}|Y_{n-1})$ is updated as follows:

$$p(t_j^{(1)}, \cdots, t_j^{(n)}|Y_n) \;=\; p(t_j^{(1)}, \cdots, t_j^{(n)}|Y_{n-1}, y_n) \tag{18}$$

$$\;=\; \frac{p(y_n|t_j^{(1)}, \cdots, t_j^{(n)}) p(t_j^{(1)}, \cdots, t_j^{(n)}|Y_{n-1})}{p(y_n|Y_{n-1})}. \tag{19}$$

This indicates that we can do smoothing by storing and resampling m sets of realizations $(t_j^{(1)}, \cdots, t_j^{(n)})_{j=1,...,m}$ with the same probability as for the filtering.

Therefore, an algorithm for smoothing can be obtained by replacing the Step 3 (d) of the algorithm for filtering by

(d)′ *Generate* $(s_j^{(1)}, ..., s_j^{(n)}) \sim (\sum_{j=1}^m \alpha_j^{(n)})^{-1} \sum_{j=1}^m \alpha_j^{(n)} I(x, t_j^{(n)})$ *for* $j = 1, ..., m$ *by the resampling of* $\{t_j^{(1)}, ..., t_m^{(n)}\}_{j=1,...,m}$.

In principle, this algorithm realizes the fixed interval smoothing for the nonlinear non-Gaussian state space model. However, in practice, since the number of realizations is finite, the repetition of the resampling (d)′ will gradually decrease the number of different samples and will causes a difficulty.

As an example, we consider the trend estimation problem shown in section 5.1 of Kitagawa (1987). We have 500 observation and estimate the trend by the Gaussian

model

$$\begin{aligned} t_n &= t_{n-1} + v_n \qquad v_n \sim N(0, 1.22 \times 10^{-2}) \\ y_n &= t_n + w_n \qquad w_n \sim N(0, 1.043) \end{aligned} \tag{20}$$

The right panel shows the true smoothed density $p(x_1|Y_{500})$ obtained by the Kalman smoother. It can be seen that after 10 steps, the realizations $\{s_1^{(1)}, ..., s_{1000}^{(1)}\}$ seems to become reasonable representatives of $p(x_1|Y_{500})$. However, it should be noted here that significant overlap of the realizations occurs and the number of actual realizations becomes only 22 after 50 steps. It is further reduced to 16 at n=100, 8 at n=200 and 2 at n=500.

Therefore, it is recommended to stop the smoothing algorithm after repeating the resampling not so many times (say less than 5 per cent of m). The algorithm may be written as follows:

(d)'' *For fixed L, generate* $(s_j^{(n-L)}, ..., s_j^{(n)}) \sim (\sum_{j=1}^m \alpha_j^{(n)})^{-1} \sum_{j=1}^m \alpha_j^{(n)} I(x, t_j^{(n)})$ *for* $j = 1, ..., m$ *by the resampling of* $\{t_j^{(n-L)}, ..., t_m^{(n)}\}_{j=1,...,m}$.

This is equivalent to apply the L-lag fixed lag smoother rather than the fixed interval smoother.

Figure 3.1 (a) and (b) show the filtered and 40-lag smoothed estimate of the trend based on the Gaussian model. On the other hand, Figure 3.2 (a) and (b) show the estimates based on a non-Gaussian model

$$v_n \sim \text{Cauchy}(0, 3.48 \times 10^{-5}), \quad w_n \sim N(0, 1.022) \tag{21}$$

figure 3.3 (a) and (b) show the "exact" estimates obtained by (a) the Kalman smoother and (b) the non-Gaussian smoother shown in Kitagawa (1987). It can be seen that the 40-lag smoothed estimates show good accordance with the ones by the "exact" fixed interval smoother.

Since the distribution is expressed by the realizations, we can obtain various statistics besides the mean value. However, they are not shown here.

3.2 Smoothing by Two-filter Formula

Another way of smoothing is to use the two-filter formula used in Kitagawa (1990). Define $Y^n = \{y_n, \cdots, y_N\}$, then we have $Y_N = Y_{n-1} \cup Y^n$. Therefore

$$p(x_n|Y_N) = p(x_n|Y_{n-1}, Y^n)$$

$$\begin{aligned} &\propto p(x_n, Y^n|Y_{n-1}) \\ &= p(Y^n|x_n)p(x_n|Y_{n-1}). \end{aligned} \tag{22}$$

Here $p(Y^n|x_n)$ can be evaluated by the following backward filtering.

$$\begin{aligned} p(Y^N|x_N) &= p(y_n|x_n) \\ p(Y^{n+1}|x_n) &= \int p(Y^{n+1}|x_{n+1})p(x_{n+1}|x_n)dx_{n+1} \\ p(Y^n|x_n) &= p(y_n|x_n)p(Y^{n+1}|x_n). \end{aligned} \tag{23}$$

We assume that the function $z = f(x, v)$ in equation (1) has an implicit function such that $v = \gamma(z, x)$.

1. *Do smoothing shown in section 2.4 and store the realizations of the predictor,* $(t_j^{(n)}, j = 1, ..., m)$ *for n=1,...,N.*

2. *Do backward filtering and the smoothing by the following algorithm*

   (a) *Compute* $\beta_j^{(n)} = r(g(y_N, t_j^{(n)}))$ *for j=1,.....,m.*

   (b) *Repeat the following steps for n=N−1,...,1*

      i. *Compute* $\delta_j^{(n)} = \beta_j^{(n+1)} q(\gamma(t_j^{(n+1)}, tj^{(n)}))$ *for j=1,...,m*

      ii. *Compute* $\beta_j^{(n)} = r(g(y_n, t_j^{(n)}))\delta_j^{(n)}$ *for j=1,...,m*

      iii. $(\sum_{j=1}^m \beta_j^{(n)})^{-1} \sum_{j=1}^m \beta_j^{(n)} I(x, t_j^{(n)})$ *is the distribution function approximating the smoother* $p(x_n|Y_N)$

      iv. *If necessary, generate* $s_1^{(n)}, ..., s_m^{(n)}$ *by the resampling of* $t_1^{(n)}, ..., t_m^{(n)}$ *with the relative weights* $\beta_1^{(n)}, ..., \beta_m^{(n)}$.

4. Examples

4.1 Trend Estimation (k=2)

For the estimation of the mean value function of the nonstationary time series, we use a second order trend model

$$\begin{aligned} T_n &= 2T_{n-1} - T_{n-2} + v_n \\ y_n &= T_n + w_n. \end{aligned} \tag{24}$$

Here v_n and w_n are white noise sequences that are not necessarily Gaussian. As the distribution of v_n or w_n, we might use, for example, the following models:

$$\begin{array}{ll} \text{Gaussian} & p(x) = (2\pi\sigma^2)^{-\frac{1}{2}} \exp\{-x^2/2\sigma^2\} \\ \text{Cauchy} & p(x) = \tau\pi^{-1}(\tau^2 + x^2)^{-1} \end{array}$$

Figure 4.1 shows an artificially generated data with three bents at n=150, 250 and 400. Figure 4.2 shows the estimate of the trend T_n based on a Gaussian model

$$v_n \sim N(0, 1.0 \times 10^{-4}), \quad w_n \sim N(0, 0.36)$$

The estimate $\hat{T}_n$ is obtained as the mean of 4000 realizations. The abrupt changes of the slope are not so clear. On the other hand, Figure 4.3 shows the results by a non-Gaussian model

$$v_n \sim \text{Cauchy}(0, xxx), \quad w_n \sim N(0, 0.36)$$

With this non-Gaussian system noise model, the sudden changes of the slope are clearly detected.

4.2 Seasonal Adjustment

The standard model for seasonal adjustment is

$$y_n = T_n + S_n + w_n \tag{25}$$

where T_n is the trend component given above and S_n is the seasonal component defined by

$$S_n = -(S_{n-1} + \cdots + S_{n-p+1}) + u_n. \tag{26}$$

Figure 5.1 shows the quarterly inventory of private companies in Japan. In the middle of the series, the energy crisis occurred and the trend of the series suddenly changed. However, the estimate by a Gaussian model shown in Figure 5.2 does not detect the abrupt changes of the trend. On the other hand, the one by a non-Gaussian model clearly detects the abrupt changes of the trend and the seasonal pattern twice in 1973 and in 1974.

4.3 Nonlinear Model

We consider the data artificially generated by the following model which was originally used by Andrade Netto et. al.(1978) and reconsidered in Kitagawa (1991):

$$\begin{aligned} x_n &= \frac{1}{2}x_{n-1} + \frac{25x_{n-1}}{1+x_{n-1}^2} + 8\cos(1.2n) + v_n \\ y_n &= \frac{x_n^2}{20} + w_n. \end{aligned} \tag{27}$$

The problem is to estimate the true signal x_n from the sequence of observations $\{y_n\}$ assuming that the model (29) is known. Our filter and smoother were applied to this problem. For comparison, the well-known extended Kalman smoother and a nonlinear smoother based on numerical computation were also applied.

Figure 6.1 shows the data y_n and the signal x_n. Figure 6.2 shows the estimate of x_n by the Monte Carlo method. It can be seen that fairly good estimate of the signal was obtained by this method. Figure 6.3 shows the results by the "exact" nonlinear smoother (Kitagawa 1991) and the extended Kalman smoother. It can be seen that the Monte carlo method provides close approximation to the "exact" smoother and that it is by far better than the extended Kalman smoother.

Reference

Harrison and Stevens (1976), "Bayesian forecasting", (with discussion), *Journal of the Royal Statistical Society*, Ser. B, 34, 1-41

Kitagawa,G. (1987) "Non-Gaussian state-space modeling of nonstationary time series" (with discussion). *J. Amer. Staist. Assoc.* 82, 1032-1063.

Kitagawa,G. (1989) "Non-Gaussian seasonal adjustment". *Computers & Mathematics with Applications*, Vol.18, No.617, 503-514.

Kitagawa,G. (1990) "The two-filter formula for smoothing and an implementation of the Gaussian-sum smoother", Research Memorandum, No.388, The Institute of Statistical Mathematics.

Kitagawa,G. (1991) "A Nonlinear smoothing method for time series analysis". *Statistica Sinica*, Vol. 1, No. 2, 371-388.

West,M., Harrison,P.J. and Migon,H.S. (1985). Dynamic generalized linear models and Bayesian forecasting (with discussion). *J. Amer. Statist. Assoc.* 80, 73-97.

Table Captions

Figure 1 (a) Initial distribution, (b) distribution function of the initial distribution
(c) system noise, (d) distribution function of the system noise
(e) predictor, (f) distribution function of the predictor
(g) filter, (h) distribution function of the filter
(i) resampled filter, (h) distribution function of the resampled filter

Figure 2 Trace of the realizations of the smoother

Figure 3.1 (a) Monte carlo filter for Gaussian model
(b) 40-lag Monte Carlo smoother for Gaussian model
Figure 3.2 (a) Monte carlo filter for non-Gaussian model
(b) 40-lag Monte Carlo smoother for non-Gaussian model
Figure 3.3 (a) "exact" smoother for Gaussian model
(b) "exact" smoother for a non-Gaussian model

Figure 4.1 Artificially generate data
Figure 4.2 Estimated trend by a Gaussian model
Figure 4.3 Estimated trend by a non-Gaussian model

Figure 5.1 Inventory of private companies in Japan
Figure 5.2 Seasonal adjustment by a Gaussian model
Figure 5.3 Seasonal adjustment by a non-Gaussian model

Figure 6.1 Data and signal
Figure 6.2 Monte Carlo smoother
Figure 6.3 "Exact" smoother
Figure 6.4 Extended Kalman smoother

Figure 1

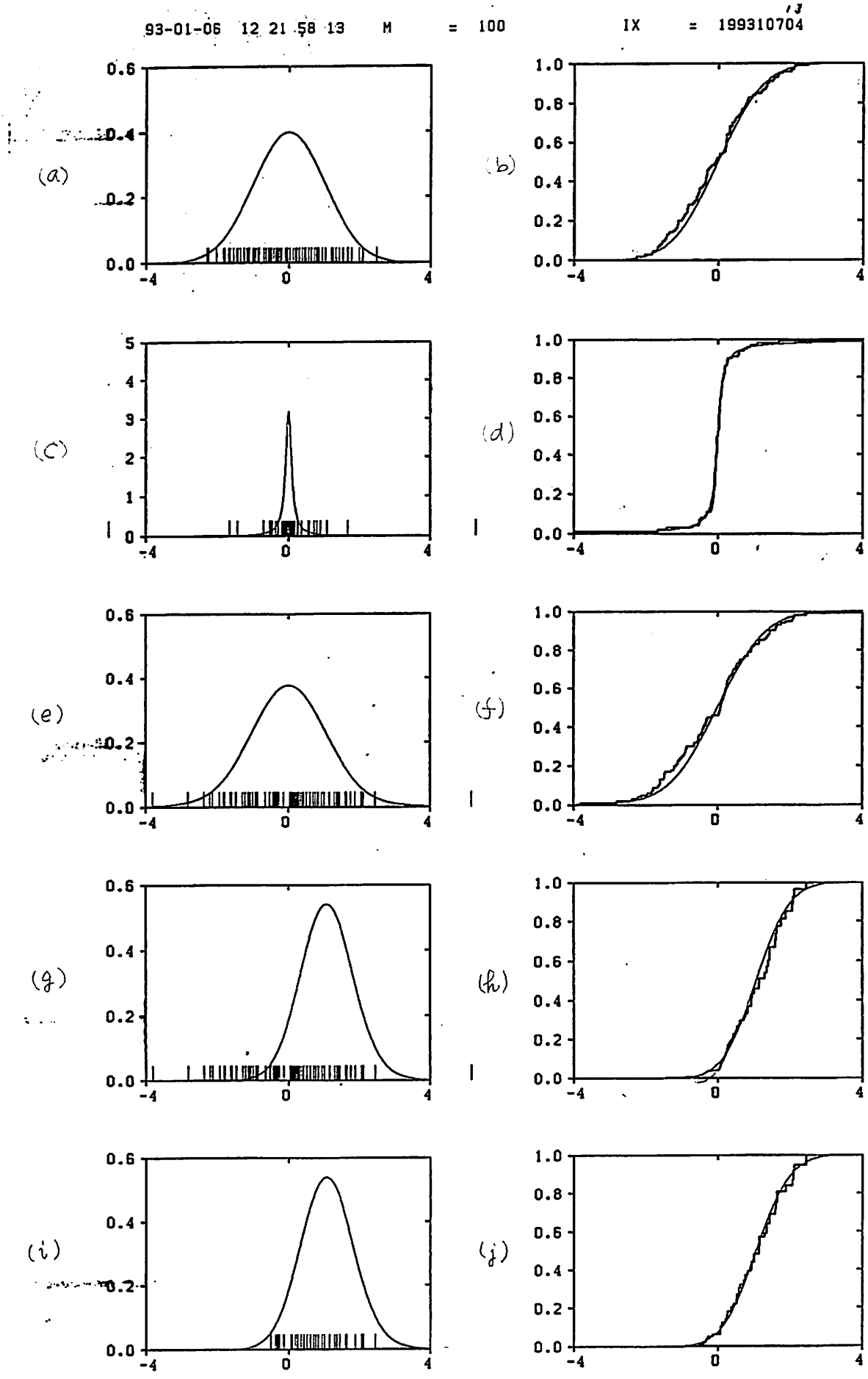


Figure 2

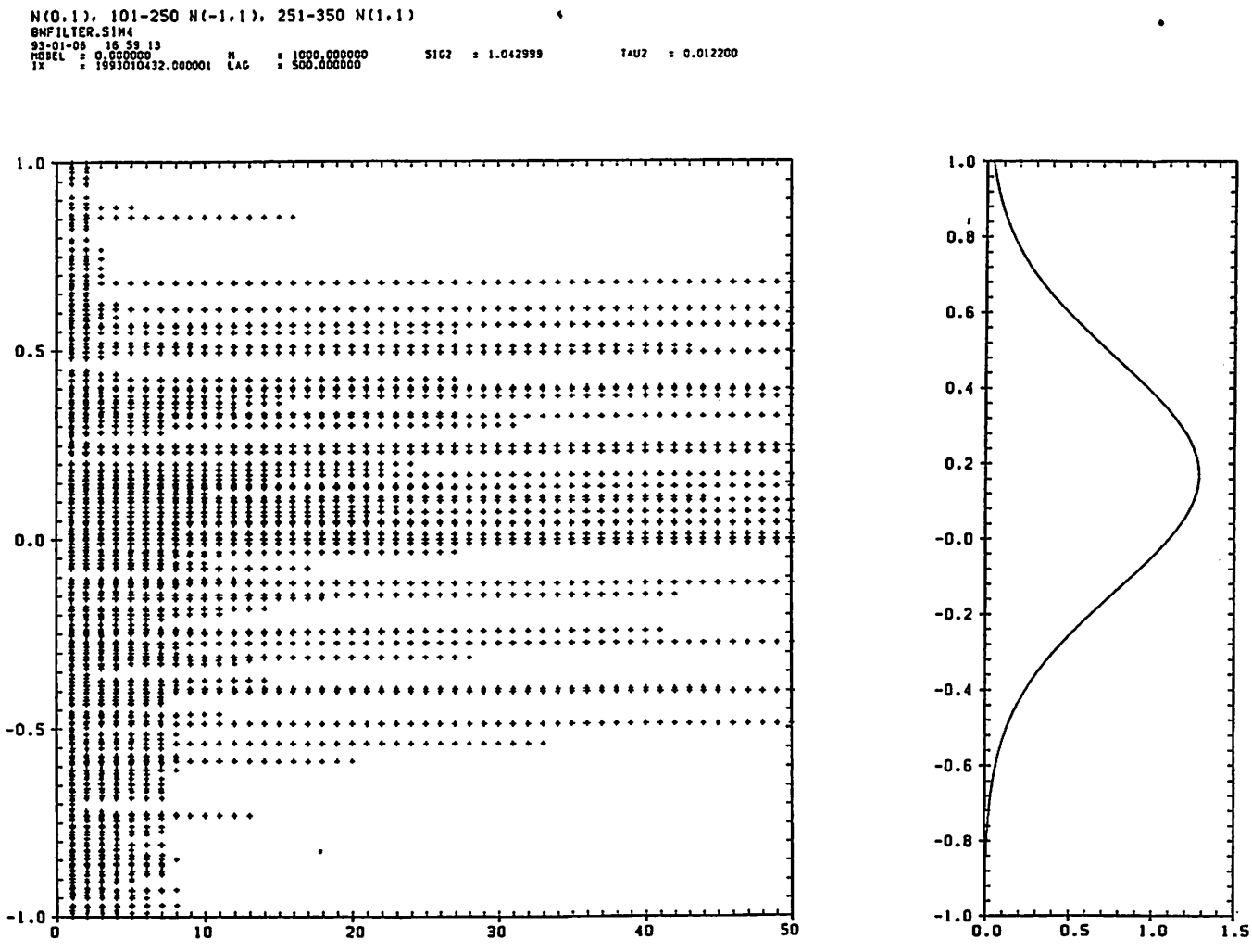
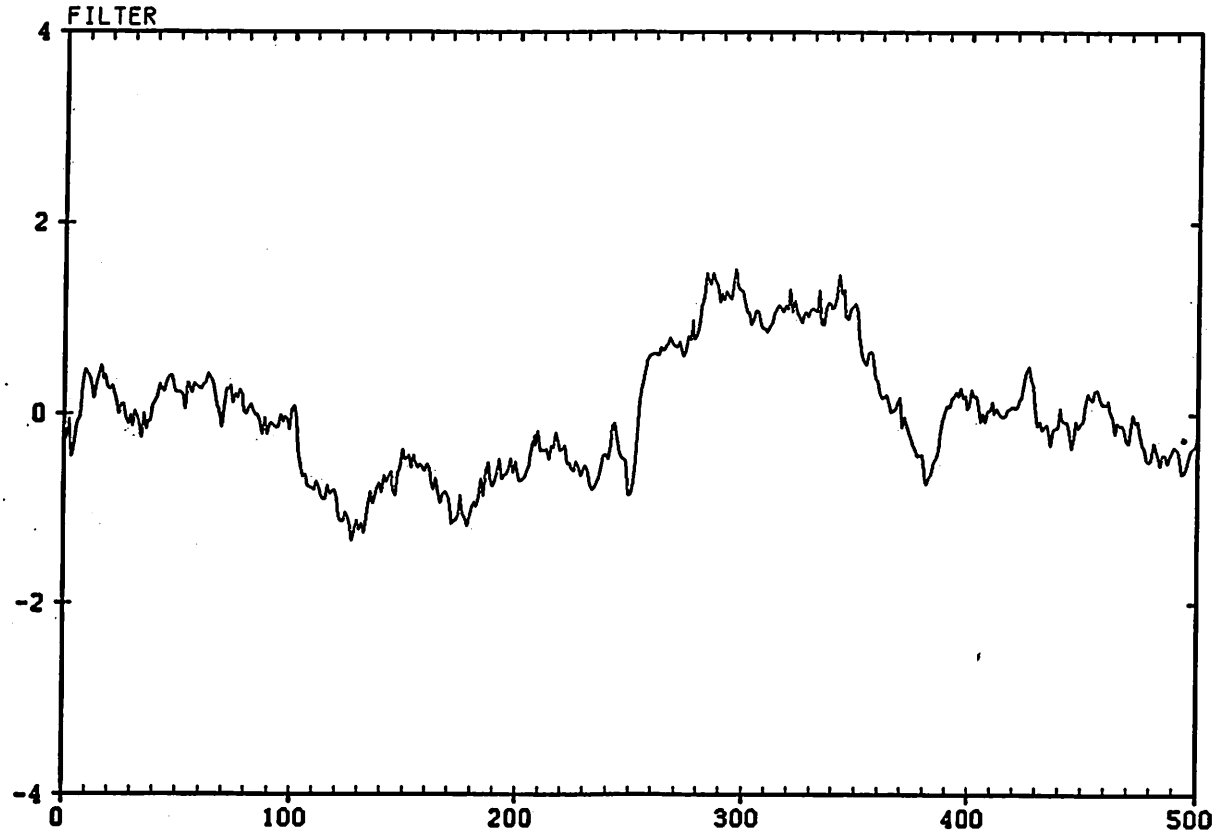
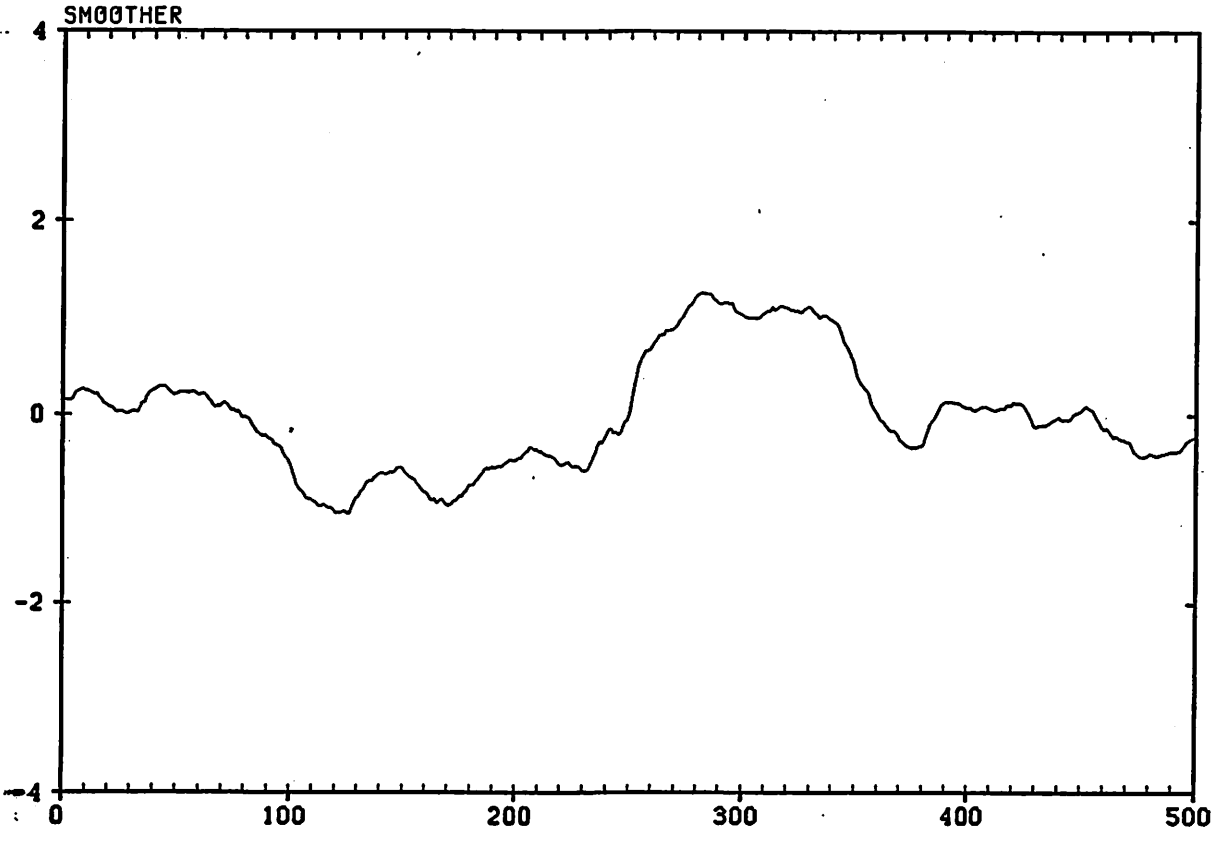

N(0,1), 101-250 N(-1,1), 251-350 N(1,1)
GNFILTER.SIM4
93-01-06 16 59 13
MODEL = 0.000000 M = 1000.000000 SIG2 = 1.042999 TAU2 = 0.012200
IX = 1993010432.000001 LAG = 500.000000
1.0
0.5
0.0
-0.5
-1.0
0
10
20
30
40
50
1.0
0.8
0.6
0.4
0.2
-0.0
-0.2
-0.4
-0.6
-0.8
-1.0
0.0
0.5
1.0
1.5

Figure 3. 1

N(0,1), 101-250 N(-1,1), 251-350 N(1,1)
@NFILTER.SIM4
92-11-30 21 52 22
MODEL = 0.000000 M = 10000.000000 SIG2 = 1.042999 TAU2 = 0.012200
IX = 1992110080.000002 LAG = 20.000000



(a)

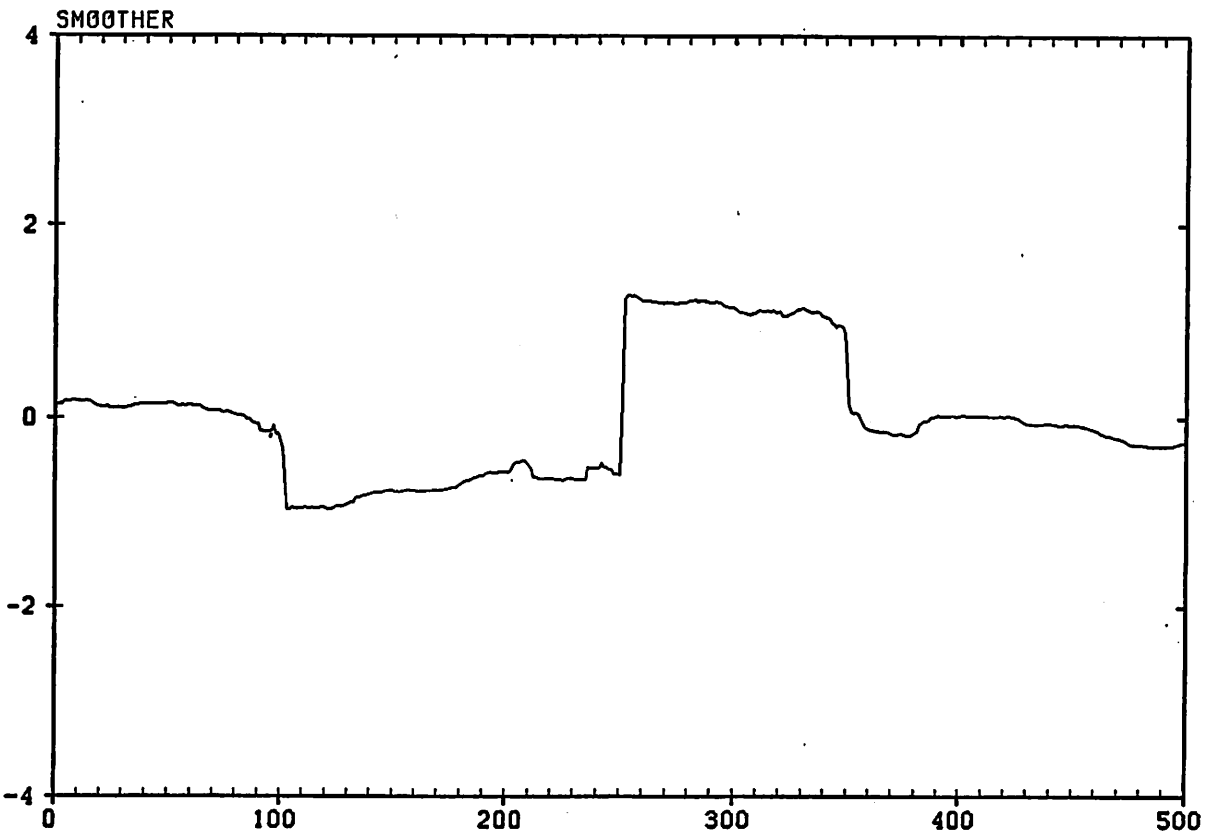


(b)

Figure 3. 2

N(0,1), 101-250 N(-1,1), 251-350 N(1,1)
GNFILTER.SIM4
92-11-30 20 29 52
MODEL = 1.000000 M = 10000.000000 SIG2 = 1.021999 TAU2 = 0.000035
IX = 1992110080.000002 LAG = 50.000000

(a)

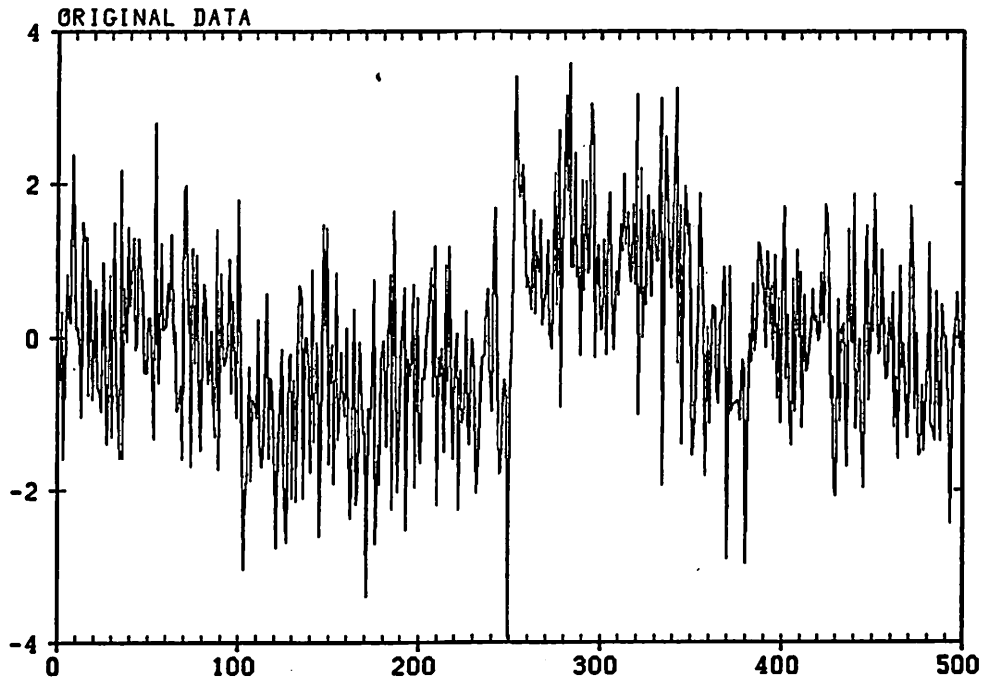


(b)

Figure 3. 3

This is the original data

(a)



(b)

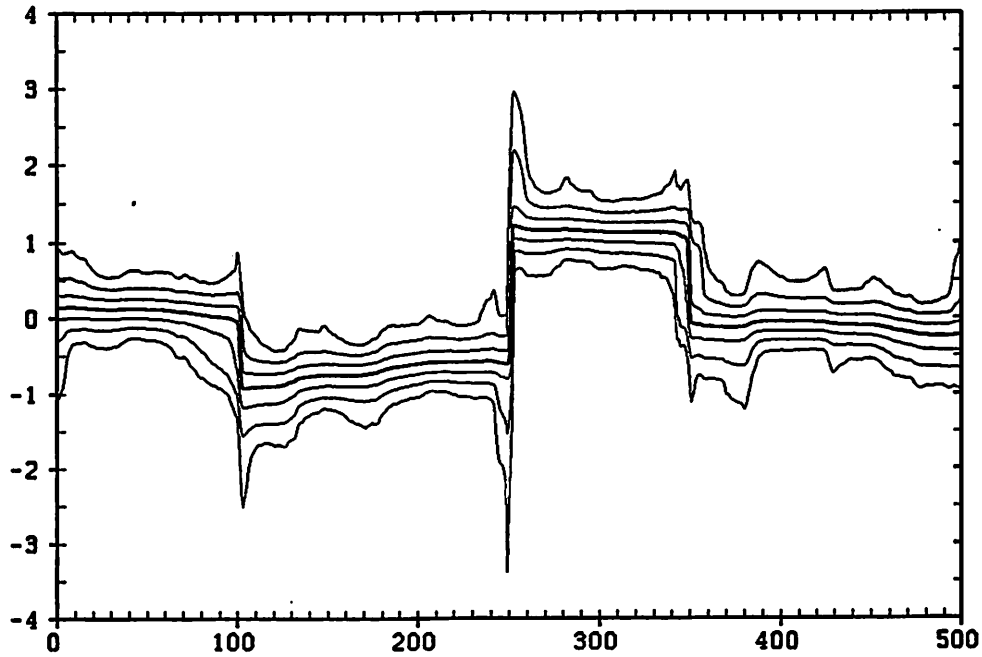


Figure 4. 1

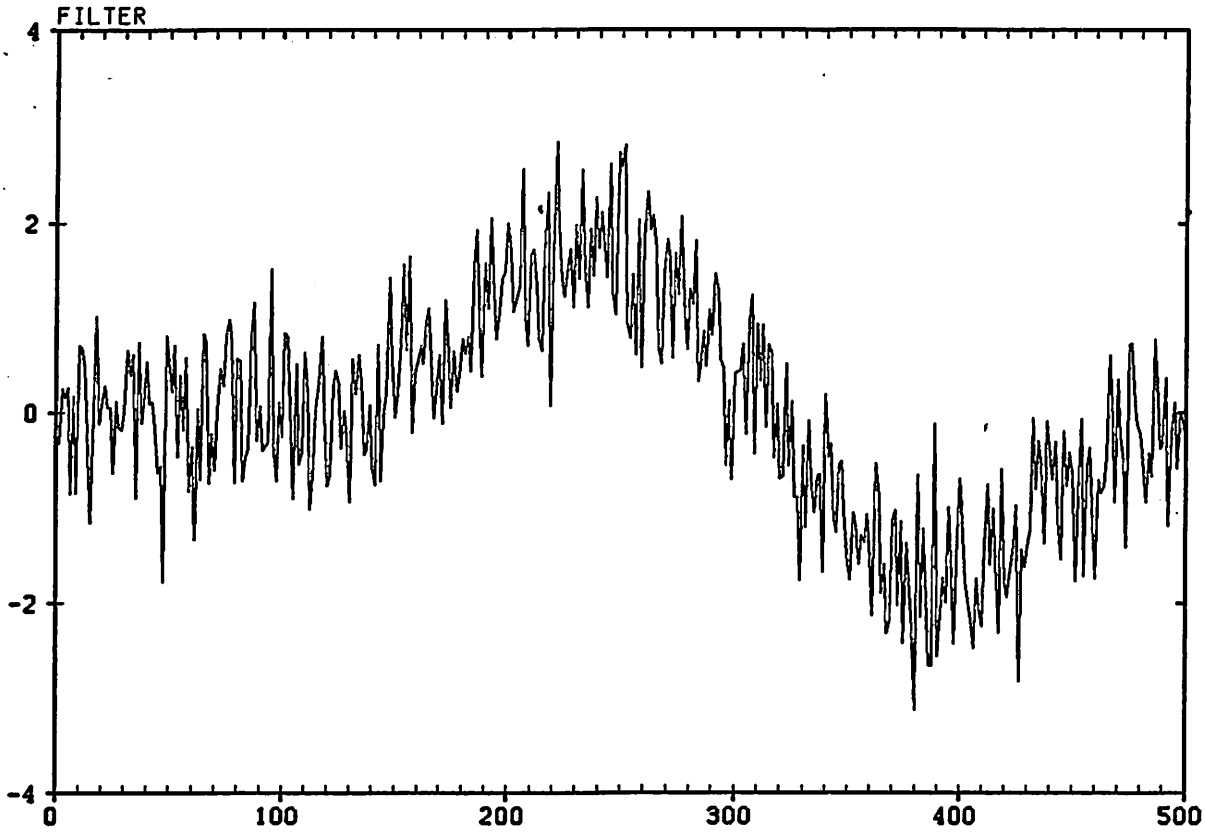


Figure 4. 2

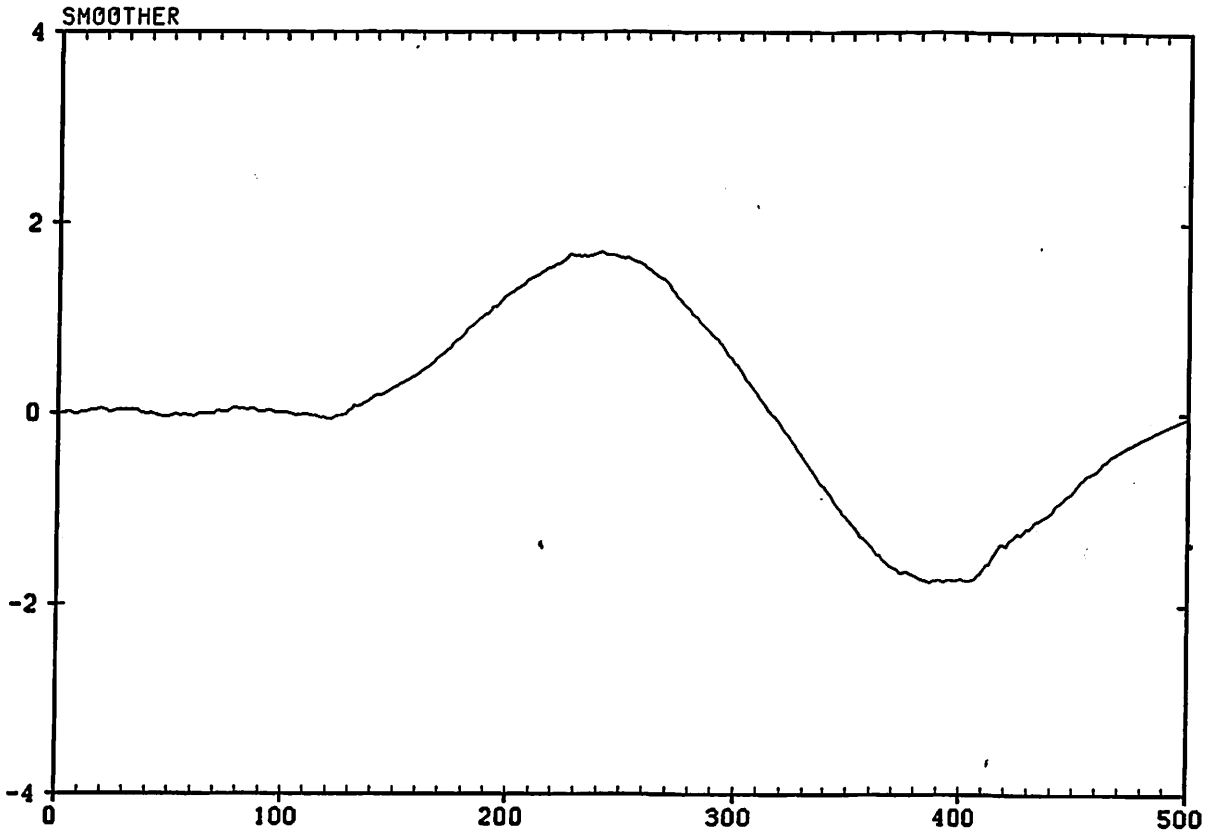


Figure 4. 3

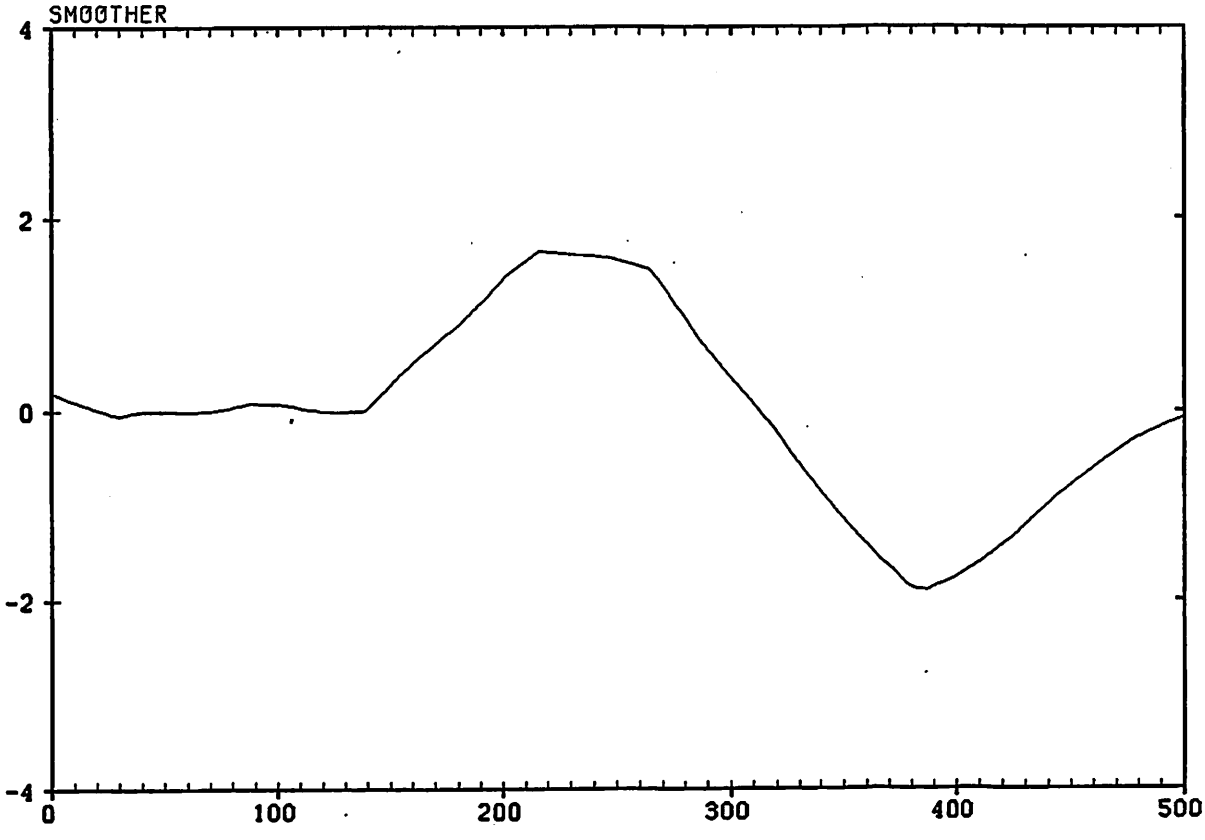


Figure 5

To be given later

Figure 6. 1

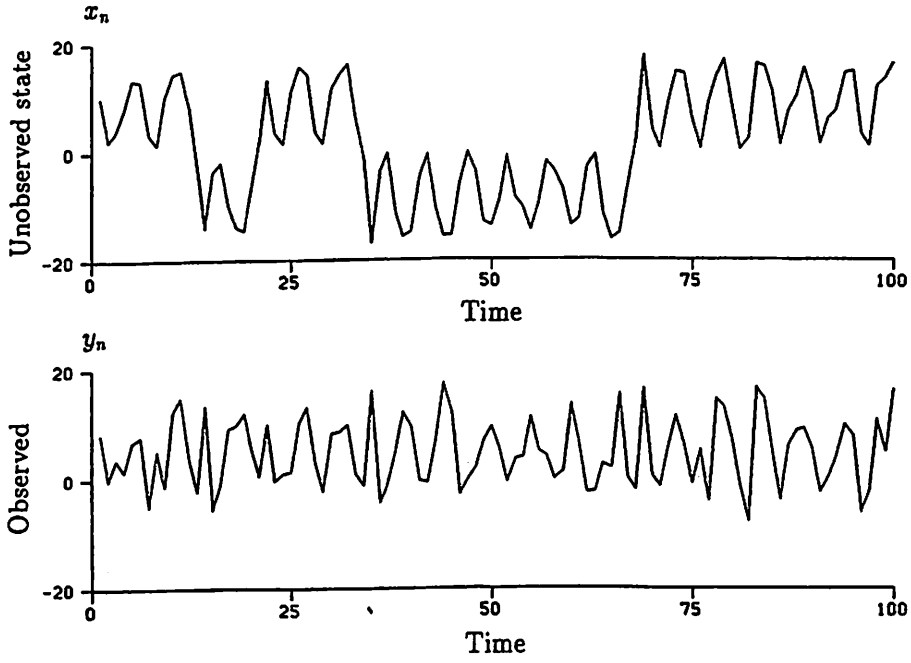


Figure 6. 2

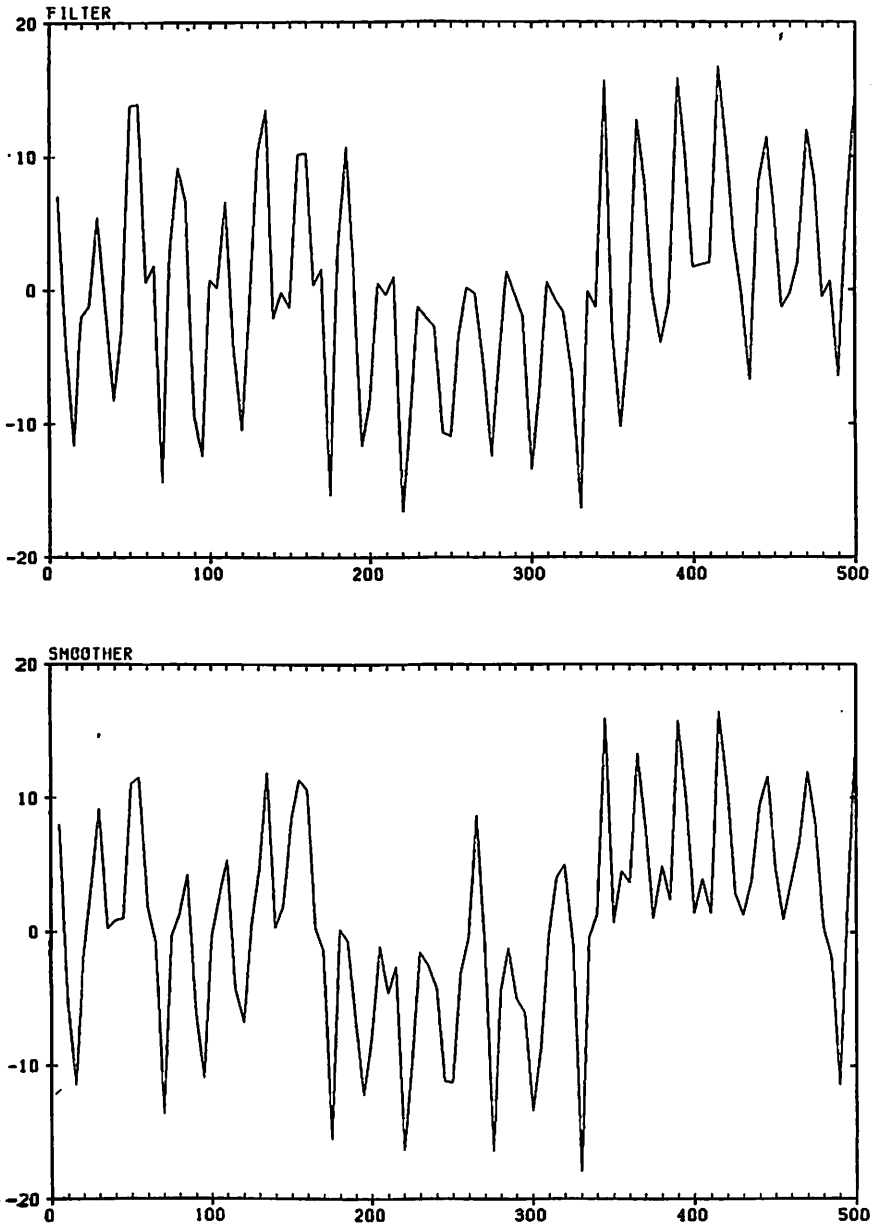


Figure 6. 3

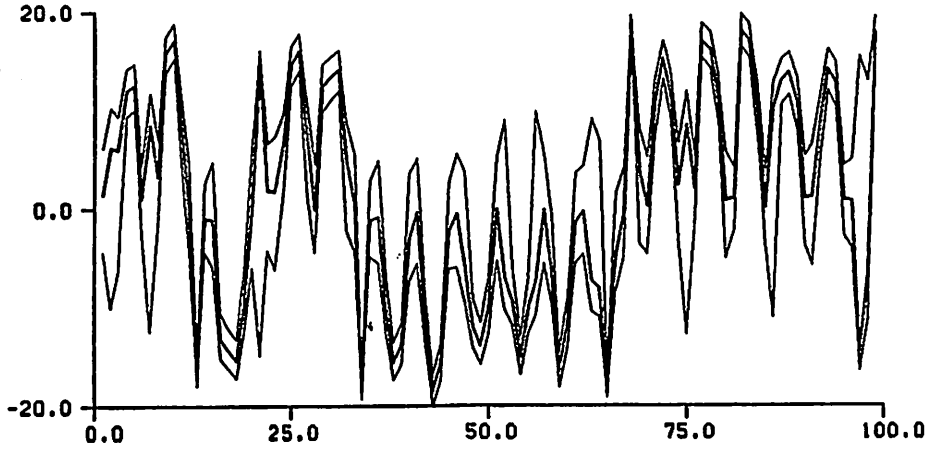


Figure 6. 4