PNC TJ1632 93-001

分置

社内資料

本資料は 01年 10月 4日付けで登録区分、
変更する。

[技術情報室]

オンライン
臨界安全監視システムの
基礎研究(II)

(動力炉・核燃料開発事業団　委託研究成果報告書)

1993年3月

摂南大学 工学部 電気工学科

山田　　澄

本資料についての問い合わせは下記に願います。
〒319-11　茨城県那珂郡東海村大字松村4－33
動力炉・核燃料開発事業団
東海事業所
技術開発推進部・技術管理課

社内資料
PNC TJ1632 93-001
1993 年 3月

オンライン臨界安全監視システムの基礎研究（Ⅱ）

山田　澄*

要　旨

一点炉近似動特性方程式に従う未臨界原子炉の中性子検出器信号揺らぎをA／D変換して得られる時系列を自己回帰移動平均モデル同定することにより、オンライン未臨界度推定が可能である。本研究は安価で即応性の良い臨界安全システムの開発にこの未臨界度推定手法を応用するための基礎研究を行うことを目的とし手いる．本研究が目的としている臨界安全監視モニターでは，信号のDC成分が刻々と変化する場合にも適用できるアルゴリズムを開発する必要があり，ハイパスフィルタにより信号の低周波成分を除去する方法を採用した．この方法は単に信号に含まれるDC成分を除去するだけでなく未臨界度推定に不必要な低周波成分を除去する効果もある．本年度は原子炉から得られた定常時系列データを用いて，ハイパスフィルタの特性が未臨界度推定の精度・推定時間に与える影響を時系列データのサンプリング周波数と関連させて検討し，以下の結論を得た．

ハイパスフィルタのカットオフ周波数のわずかな違いが未臨界度の推定値の精度に及ぼす影響は極めて大きいものであることが確認され，①ハイパスフィルタのカットオフ周波数が低すぎて十分DC成分を除去できない場合，未臨界度は実際よりも浅く推定され，②カットオフ周波数が適正値より高い場合，未臨界度は実際よりも深い方に偏る．これはハイパスフィルタの影響が強く，同定されるシステムがハイパスフィルタとの複合形になっていることが原因である．一般的には，ハイパスフィルタのカットオフ周波数はサンプリング周波数に比べてなるべく低くく，またフィルタの次数も低い方が望ましいが，最適カットオフ周波数としては，フィルタのDC成分低減率の変化が最も大きい周波数を選択することが望ましい．

※　本報告書は、山田澄が動力炉・核燃料開発事業団の委託により実施した研究の成果である。

契約番号・・・・・・・・：０４０Ｄ０１１６
事業団担当部課室：東海事業所、安全管理部・安全技術課　　野尻一郎

＊　摂南大学 工学部 電気工学科

OFFICIAL USE ONLY
PNC TJ1632 93-001
MARCH, 1993

Basic Study on On-line Criticality Surveillance System (II)

Sumasu Yamada*

Abstract

In the framework of the point reactor kinetics approximation, on-line subcriticality monitoring can be done by Auto-Regressive Moving Average (ARMA) Model identification of the time series sampled from the signal fluctuation of a neutron detector in a subcritical reactor.

This research focuses the attention on the basic analysis of applicability of the recursive parameter estimation algorithm to developing an inexpensive on-line criticality surveillance system of quick response. Since Criticality Surveillance System demands an algorithm applicable for non-stationary time series with the time-varing DC component, the high pass filter has been adopted to remove the DC component, and it turns out that it is effective for removing not only the DC component but also the low frequency component of the time series. The effects of the high pass filter on the accuracy and response-time of subcriticality estimation have been examined through analysis of the real data from Kinki University Research Reactor, and we obtained the following results.

The estimated subcriticality is fairly sensitive to the cut-off frequency of the high pass filter. If the cut-off frequency is too low, the subcriticality is underestimated. On the contrary, if the cut-off frequency is too high, the subcriticality is overestimated. This is because the model identified from the data includes the property of the high pass filter. Generally, the cut-off frequency of the high pass filter should be selected as low as possible compared with the sampling frequency and the model order of the filter should be as small as possible. However, the numerical analysis of the real time series data indicates that the optimum cut-off frequency should be determined as the frequency where the derivative of the DC component reduction rate of the high pass filter takes a maximum value.

Work performed by Sumasu Yamada under contract with Power Reactor and Nuclear Fuel Development Corporation.
PNC Liaison: Safety Technology Development Section Health and Safety Division, Tokai Work, Ichiro Nojiri.

* Department of Electrical Engineering, Faculty of Engineering, Setsunan University

目次

表・図目次

1　緒言

ＴＭＩ事故，チェルノブイリ事故を他山の石と考えていたわが国においても，平成３年２月９日，関西電力美浜原子力発電所２号機でＥＣＣＳが作動する蒸気発生器伝熱管破断事故が起き，原子力発電への不安をますます増大させる結果となった．

今後，原子力発電をはじめ核燃料再処理施設における安全性の確保及び防護システムはより高度なものを要求される時代に入りつつある．

この様な状況下において，我国では本年１９９３年(平成５年)後半には，高速増殖炉原型炉「もんじゅ」が初臨界を迎え，また，５年後の１９９８年には商業用核燃料再処理工場が稼働する予定となっており，本格的な核燃料サイクル時代を迎える．しかし核燃料サイクルの中心となる再処理技術は，民間レベルでその技術を運用しているのはイギリスとフランスのみで経験的に未知数な面も多く，安全管理の面でも多くの問題点が残されている．特に，処理中の核燃料系の反応度が増大することによって臨界超過の状態に達した場合の臨界事故は最も危険なものであり，作業員の生命の危険はもとより再処理工場システムそのものを復旧不可能な状態に陥れる可能性もある．従って核燃料再処理行程において核分裂性物質の非定常的な流れや蓄積を伴う過程においては，体系の反応度を管理しその未臨界状態を保証する何らかの臨界安全監視システムが不可欠である．

従来から，未臨界度推定に関しては様々な研究がなされてきている。その中には燃料体系にパルス中性子を打ち込み，体系内の中性子密度の時間的減衰を計測するパルス中性子法など厳密な未臨界度測定法があるが，再処理工場などの臨界安全監視システムに要求される機能はこれらとは別の次元にあると考えられる．もちろん

精度が高いということに越したことはないが，そのために大がかりな設備が必要になったのでは再処理のコストにも影響を及ぼし，現在のような軽水炉用燃料が安価な時代では核燃料サイクル自体が商業的には成立し得なくなる恐れがある．また，上述した美浜原子力発電所の事故の例では，事故時に処理するデータ量が一時的にプラント計算機の許容負荷量を超えたとも報告されていることを考慮すると[1]，計算に要するデータ量は極力少ないのことが望ましい．そこで，再処理工場における臨界安全監視システムには少なくとも以下のような条件が要求されるものと思われる．

① 合理的な精度と追従性

臨界安全監視システムは，未臨界度を測定するためのシステムではなく，あくまで臨界事故を防ぐことを主目的とするシステムである．従って，推定値が真値に極めて近い精度を有していることは必ずしも要求されてはいない．再処理工場では各部位の許容反応度を設定し，その値に安全係数を掛けた値を臨界安全監視システムに組み込み，その値を超えると警告もしくはシステムの停止信号を発すると考えられ，必ずしも高精度の未臨界度推定が必要とは考えられる．それよりもむしろ未臨界度の変化に対するリアルタイムでの追従性が重視されるべきである．

② コンパクトなシステム

再処理工場は複雑な配管が施され，各施設には厳重な遮蔽が設けられる．このことを考慮すると臨界安全監視システムが大がかりな装置となることは工場の機能性を損なうことにもなりかねない．また，作業員の被爆や放射能による汚染を極力防ぐためにも点検・補修の容易なシステムであることも重要な条件であると考えられる．従って，構成パーツの少ないコンパ

クトなシステムであることが要求されるべきである.

③ 安価なシステム

再処理工場での臨界安全監視システムは，その異常検知により工場の停止・作業員の避難を引き起こすことを考慮すると誤検知は避けなければならない．従って，臨界監視が必要な場所１つ１つに最低３基の測定系を配置し，原子力発電所などで用いられている「2 out of 3」方式によって反応度異常信号の発信を行うことが望ましい．しかし，この方式を臨界事故が起こり得る場所全てに配置するのであれば１基当たりのコストの高低が再処理価格に影響を及ぼすことから，１つのコンポーネントが安価なシステムであることも重要な条件として要求されるべきである.

そこで，上記の３点を考慮し，我々はオンラインパラメータ推定アルゴリズムに基づく未臨界度推定手法の研究を行ってきた．この手法は，原子炉は一点炉近似動特性方程式に従うものとし，未臨界原子炉の中性子検出器出力信号の揺らぎは離散時間系において自己回帰移動平均モデル(ＡＲＭＡモデル)で表現できることに着目し[2]，モデル伝達関数の極をオンラインで計算することにより未臨界度を推定するものである.

未臨界度推定精度は必ずしも高くはないが，この手法には以下のようなメリットがある.

① この手法により未臨界度の推定を行うために必要となる機器は，γ線補償型中性子検出器およびその電流増幅器，信号処理用のアンプ及びアナログフィルタ，Ａ／Ｄコンバータとデータ処理用のコンピュータであり，システムを構成する要素に特殊なものを一切必要としない。したがって，全て

市販品で比較的簡単にシステムを構成することができる．この手法では逐次計算を行うため観測データを蓄積して一括処理する必要がないため，大型汎用計算機など容量の大きな計算機を必要とせず，エンジニアリングワークステーションあるいは高性能パーソナル計算機の利用が可能である．最近その技術の進展が著しいＤＳＰ(Digital Signal Processor)を用いればワン・ボード化も可能である．なお、当然ではあるが適当な強度を持つ中性子源が必要であることは言うまでもない．

② 逐次パラメータ推定アルゴリズムを用いることによりデータの集積を待つこと無く，時々刻々変化する未臨界度をオンラインで追跡することが可能である．

③ 近年，原子力発電プラントではシステム診断に炉雑音解析手法を用いているが，そのデータ管理は膨大な磁気メディアを駆使することによって行われているのが現状である[3]．本手法は，ＡＲＭＡモデルパラメータからシステムの周波数応答を簡単に再生できるのでシステムの異常診断にも応用が可能であり，また，データの保存は１ポイントにつき４つのパラメータのみで済むため，測定対象の履歴を管理する上でも生データやＦＦＴ法などに比べてそのデータ量は大幅に削減・圧縮が可能である．

本手法を未臨界原子炉のシミュレーション時系列データに用いた場合では，比較的精度よくかつ短時間で未臨界度推定が可能との解析結果が得られている[4] [5]．しかし，この手法を実際の原子炉から取得したデータに応用するに当たっては次のような問題点が残されている．

① 観測信号の前処理

研究用零出力炉である近畿大学原子炉から実際に取得された時系列データ

にたいしシミュレーションデータに用いたアルゴリズムをそのまま適用した場合，未臨界度推定値の精度及び変化する未臨界度の追従性がシミュレーション時系列データに対する結果に比べて良くない．この様に実データによる未臨界度推定がシミュレーション時系列データによる推定と比較して劣る理由として未知の外乱の問題が考えられる。すなわち，シミュレーション時系列データは，ほぼ完全な白色時系列を Shaping Filter で加工して生成されており，その時系列は理論に従った性質の良いデータであるのに対し，実データは未知の外的温度の変動，測定系に対する機械的振動等の外乱が加わる上，検出器，増幅器，フィルター，Ａ／Ｄコンバータ等を通過し，各種の処理を施されるため，信号の質(quality)そのものがシミュレーションデータとは本質的に異なっているものと考えられる．特に本手法の様なパラメトリックなスペクトル解析では，ＦＦＴ法などのスペクトル解析とは異なり，限られたパラメータでスペクトル全体を表現するため，未知の雑音によるわずかな信号の歪がパラメータの推定値全体に大きく影響を及ぼす．したがって、観測信号から未臨界度の情報を含む周波数領域のみを最適に抽出する信号処理技術の開発が必要となる。

② 変動するパラメータの追従性

本手法の様な逐次計算アルゴリズムのみならず、いずれの手法にも共通の問題であるが，未臨界度推定法を実用化する上で最も重要なことは，未臨界度が刻々変化する非定常状態にける未臨界度の変化の追従性である。

そこで，本年度（平成４年度）は，上記①の問題の中でもこれまでの予備研究で重要な問題と考えられる時系列データ中に含まれる低周波成分の未臨界度推定に及ぼす影響を検討することが研究課題である．信号処理としてはアナログ信号の状態での前処理と時系列データに変換した後の処理法とがあるが，本研究では特に時系

列データから余分な低周波成分を除去するためのハイパスフィルタと未臨界度推定精度の関係を近畿大学原子炉から得られた時系列データを用いて明らかにし，上記①の問題に対する指針を得る．具体的には，

① 時系列データ中に含まれる低周波成分の適切な除去法と，時系列データのサンプリング周波数が推定値に及ぼす効果．

② アナログ信号の量子化にともなう，量子化誤差によるデータの質の低下が推定値に及ぼす影響．

について検討する．

本論文の構成は以下の通りである．

第２章では，昨年度の報告書で既にその概要を報告しているが、本報告書が一冊の資料として自己整合性を保つよう，本研究で用いているオンラインＡＲＭＡ同定アルゴリズムを改めて説明する．本研究での中心的アルゴリズムであるＲＰＥ(Recursive Prediction Error)法については，特に実際のプログラム上で計算過程に現れる各種の計算パラメータを詳しく説明する．

第３章では，その前半部で近畿大学原子炉実験での機器配置など実験の概要と，アナログ信号からデジタル信号に変換する際のサンプリング周波数について詳しく論じる．後半部では，ＡＲＭＡモデルから未臨界度への変換の手法と，推定未臨界度の評価の方法について説明する．

第４章では，定常状態における時系列データに対して，時系列データ中の低周波成分の除去と推定精度について検討し，第３節ではサンプリング周波数が推定時間に及ぼす影響について検討する．第５節ではＡ／Ｄ変換時の量子化ｂｉｔ数と推定精度の関係について考察する．

第５章では，本研究で得た成果をまとめる．

ところで，本研究のような研究では，同じ計算機を用いる中性子輸送計算や熱流動計算などとは異なり，確立されて計算コードがあるわけではなく，専ら各研究課題に即した計算コードの開発が必要となる．したがって，新しいアルゴリズムの導入や改良が容易に行える反面，計算コードが一般的に使用されてプログラムのバグが発見されると言ったことがなく，用いたプログラムによる解析結果の問題点が明らかになりにくいという側面をもつ．３年毎に開かれるＳＭＯＲＮ(Specialist Meeting On Reactor Noise)のベンチマークテストでも明らかになっているが，同一のアルゴリズムを使用しているにもかかわらず研究者間によって結果が大きく異なったりすることがある．このとき，それが単なるサンプリング周波数の設定やモデル次数の選択等の誤りによるものなのか，プログラム上での本質的な誤りによるものなのか判断がつかないことがある[11]．

そこで，本報告にはこの研究で用いた解析基本プログラムを APPENDIX に示す．

プログラムの動作環境は，

① ハードウェア　：　ＰＣ－９８０１シリーズ
実装メモリ６４０ＫＢ以上
浮動小数点数値演算コプロセッサ実装

② ＯＳ　：　ＭＳ－ＤＯＳ

である．なお，ＳＯＮＹデータレコーダより直接ディジタルデータを読み出すにはハードウェアとして，ＧＰ－ＩＢ(ＩＥＥＥ－４８８)インターフェイスボードが必要となる．

2　オンラインＡＲＭＡパラメータ同定アルゴリズム

この章では，本研究で用いるオンラインＡＲＭＡパラメータ同定アルゴリズムについて述べる．

まず第１節に，カルマンフィルタアルゴリズムに等価な逐次推定手法である Recursive Prediction Error Method（ＲＰＥ法）のパラメータ推定手順，及び，本研究で用いている計算プログラム上でのデータ処理の方法などについて解説する．第２節には，ＲＰＥ法の特殊なタイプである PseudoLinear Regression Method（ＰＬＲ法），第３節には，やはりＲＰＥ法の特殊なタイプである Extended Least Squares Method（ＥＬＳ法）について説明する．ＲＰＥ法で問題となる推定値の偏りを補正する方法として残差を白色化するフィルタを用いるが，第４節ではこのフィルタの安定性を保証するための Projection Algorithm を計算プログラム上でどのように実現しているかを述べる．

2．1　Recursive Prediction Error（ＲＰＥ）法[6]

１入力・１出力システムにおいて，入力を$u(t)$，出力を$y(t)$，外乱を$e(t)$としたとき，このシステムは次式に示すようなＡＲＭＡＸ(ｐ，ｑ，ｒ)モデルで記述することができる．

$$y(t)+\sum_{i=1}^{p}a_i y(t-i)=\sum_{k=1}^{r}b_k u(t-k)+e(t)+\sum_{j=1}^{q}c_j e(t-j) \qquad (2.1)$$

なお，$e(t)$ は白色時系列であるものとする．$u(t)$ と$y(t)$ が観測可能で，$e(t)$ が観測不可能な場合、自己回帰（ＡＲ）部パラメータ：a_i，外部入力（ｅＸｏｇｅｎｅｏｕｓ）部パラメータ：b_i，移動平均（ＭＡ）部のパラメータ：c_iを逐次的に求める手法として Recursive Prediction Error（ＲＰＥ）法が良く知られ

ている．このアルゴリズムは揺らぎを発生させる駆動力としての入力$u(t)$が観測不可能な場合，式(2.1)の$u(t)$を零とすることにより得られるＡＲＭＡ(ｐ，ｑ)過程のモデルパラメータ推定アルゴリズムとしても用いることが可能である．従って，未臨界原子炉の中性子検出器出力信号揺らぎの場合，ＡＲＭＡ(ｐ，ｑ)モデルのパラメータ推定としてこのアルゴリズムを用いが，ここではより一般的なＡＲＭＡＸモデルの逐次パラメータ推定アルゴリズムについて説明する．

時間推移オペレータz^{-1}を用いると，式(2.1)は次のように書ける．

$$A(z^{-1})y(t) = B(z^{-1})u(t)+C(z^{-1})e(t) \tag{2.2}$$

$$A(z^{-1}) = 1+a_1z^{-1}+a_2z^{-1}+\cdots+a_pz^{-1}$$

$$B(z^{-1}) = 1+b_1z^{-1}+b_2z^{-1}+\cdots+b_rz^{-1}$$

$$C(z^{-1}) = 1+c_1z^{-1}+c_2z^{-1}+\cdots+c_qz^{-1}$$

ここで，$e(t)$が観測できるとして観測データベクトル

$$\phi^T(t) = [-y(t-1),\ -y(t-2),\ \cdots,\ -y(t-p),\ u(t-1),\ u(t-2),\ \cdots,\ u(t-r),\ e(t-1),\ e(t-2),\ \cdots,\ e(t-q)] \tag{2.3}$$

及び，これから推定しようとする各パラメータベクトル

$$\theta^T = [a_1,\ a_2,\ \cdots,\ a_p,\ b_1,\ b_2,\ \cdots,\ b_r,\ c_1,\ c_2,\ \cdots,\ c_q] \tag{2.4}$$

を定義すると，予測誤差$v(t)$は式(2.1)より，

$$v(t) = y(t)-\theta^T\phi(t) \tag{2.5}$$

なる関係がある。そこで評価関数

$$V_N(\theta) = \frac{1}{N}\sum_{t=1}^{N}[y(t)-\theta^T\phi(t)]^2 \tag{2.6}$$

を最小にするようなパラメータθを逐次的に求めるのが逐次型ＡＲＭＡモデルパラメータ同定アルゴリズムである．このことは$V_N(\theta)$が最小，すなわち予測誤差v(t)の分散を最小とするパラメータを推定することになる．したがって，最適パラメータは

$$\frac{\partial V_N(\theta)}{\partial \theta} = 0 \tag{2.7}$$

となるようにθを選べばよい．この様なθを$\hat{\theta}(N)$とおくと，

$$\hat{\theta}(N) = \left[\sum_{t=1}^{N}\phi(t)\phi^T(t)\right]^{-1} \cdot \sum_{t=1}^{N}\phi(t)y(t) \tag{2.8}$$

で与えられることになる．これを逐次計算方式に書き換えるため，式(2.8)においてN→tに書き換え$\phi(t)$の(p+r+q)×(p+r+q)共分散行列$R(t)$を

$$R(t) = \sum_{k=1}^{t}\phi(k)\phi^T(k) \tag{2.9}$$

と定義し，その逆行列の平均値を，

$$P = \overline{R^{-1}}(t) = \frac{1}{t}R^{-1}(t) \tag{2.10}$$

と表すと，以下に示す逐次計算アルゴリズムを得る．

なお、確率時系列データの相関行列$\overline{R}(t)$は使用するデータ数が増加するに連れて一定の値に収束する．従ってその逆行列であるＰマトリックスも収束に向かう．従ってマトリックス内の各要素も非常に小さな値になる．後で述べるとおりＰマトリックスは推定パラメータ$\hat{\theta}$の修正量をコントロールするゲインファクターであり，Ｐマトリックスの収束は推定パラメータの収束を意味する．Ｐマトリックスが収束すると新規の情報に対する感度が鈍くなる．従って，パラメータの更新と言う観点

からは，Pマトリックスが収束している状態でパラメータ変動が発生するような時系列データにおいては，パラメータ変化の追跡が大幅に遅れることが起こる可能性がある．

以下に逐次パラメータ推定アルゴリズムを示す．説明の過程で用いられる各シンボルを Table 2.1 にまとめて示す．

Table 2.1 Notations used in the RPE method.

Symbol	Discription	Formation
$\hat{\theta}(t)$	Estimation parameter vector of $\theta(t)$	(p+r+q) Row-vector
$L(t)$	Gain vector	(p+r+q) Row-vector
$P(t)$	P-Matrix	(p+r+q)*(p+r+q) Matrix
$\phi(t)$	Time series vector	(p+r+q) Row-vector
$\phi(t)$	Revised time series vector	(p+r+q) Row-vector
e_p	Prediction Error	A variable
$\lambda(t)$	Forgetting Factor	A variable
K_t	Contraction Factor	A variable
$\tilde{y}(t)$	Revised value of $y(t)$	A variable
$\tilde{u}(t)$	Revised value of $u(t)$	A variable
$\tilde{v}(t)$	Revised value of $v(t)$	A variable

先程は説明を簡単にするため$e(t)$は観測できるとしてが，実際には観測出来ないので$e(t)$を推定しながらパラメータ推定を行っていかなければならない。そのため，$e(t)$の推定値として式誤差（残差とも呼ぶ）$v(t)$を次のように定義する．

$$v(t) = y(t) - \hat{\theta}^T(t)\phi(t) \tag{2.11}$$

ただし，ここで$\phi(t)$は(2.3)式で$e(t)$を$v(t)$に置き換えたものである．
以下に Recursive Prediction Error Method（ＲＰＥ法）のアルゴリズムを示す。

観測される時系列データ$y(t)$，$u(t)$を新たに取得する毎に以下の［ＳＴＥＰ］に従って各種の計算と推定パラメータの更新を行う．

［ＳＴＥＰ１］　$y(t)$の推定値$\hat{y}(t)$の計算．

$$\hat{y}(t) = \hat{\theta}^{T}(t-1)\phi(t) \tag{2.12}$$

［ＳＴＥＰ２］　Prediction Error の計算．

$$e_p = y(t)-\hat{y}(t) \tag{2.13}$$

［ＳＴＥＰ３］　$\hat{\theta}(t)$の更新．

$$\hat{\theta}(t) = \hat{\theta}(t-1)+L(t-1)\cdot e_p \tag{2.14}$$

［ＳＴＥＰ４］　式誤差(残差)$v(t)$の計算．

$$v(t) = y(t)-\hat{\theta}^{T}(t)\phi(t) \tag{2.15}$$

［ＳＴＥＰ５］　式誤差の非白色性による推定値の偏りを補正．
$1/\hat{C}(z^{-1})$による式誤差の白色化

$$\tilde{y}(t) = y(t)+\sum_{i=1}^{p}(K_i\cdot \hat{c}_i(t)\cdot(-\tilde{y}(t-i))) \tag{2.16}$$

$$\tilde{u}(t) = u(t)-\sum_{i=1}^{r}(K_i\cdot \hat{c}_i(t)\cdot \tilde{u}(t-i)) \tag{2.17}$$

$$\tilde{v}(t) = v(t) - \sum_{i=1}^{q}\left(K_i \cdot \hat{c}_i(t) \cdot \tilde{v}(t-i)\right) \tag{2.18}$$

［ＳＴＥＰ６］　$\phi(t)$ 及び $\phi(t)$ の更新.

$$\begin{aligned}\phi^T(t+1) = [&-y(t),\ -y(t-1),\ \cdots,\ -y(t-p+1),\\ &u(t),\ u(t-1),\ \cdots,\ u(t-r+1),\\ &v(t),\ v(t-1),\ \cdots,\ v(t-q+1)]\end{aligned} \tag{2.19}$$

$$\begin{aligned}\phi^T(t+1) = [&-\tilde{y}(t),\ -\tilde{y}(t-1),\ \cdots,\ -\tilde{y}(t-p+1),\\ &\tilde{u}(t),\ \tilde{u}(t-1),\ \cdots,\ \tilde{u}(t-r+1),\\ &\tilde{v}(t),\ \tilde{v}(t-1),\ \cdots,\ \tilde{v}(t-q+1)]\end{aligned} \tag{2.20}$$

以降 $\phi(t)$ はこの更新されたものである.

［ＳＴＥＰ７］　マトリックス $P(t)$ の更新.

$$P(t) = \frac{P(t-1)}{\lambda(t)} - \frac{P(t-1)\phi(t)\phi^T(t)P(t-1)}{\lambda(t)+\phi^T(t)P(t-1)\phi(t)} \cdot \frac{1}{\lambda(t)} \tag{2.21}$$

［ＳＴＥＰ８］　ゲインベクトル $L(t)$ の更新.

$$L(t) = P(t)\phi(t) \tag{2.22}$$

［ＳＴＥＰ９］　周辺処理を行い［ＳＴＥＰ１］へもどる.

上記のアルゴリズムを実際の計算機プログラムとして実現するに当たり，アルゴリズム上，以下に示すような二つの補助パラメータを導入している.

①　忘却係数.

忘却係数 λ は［ＳＴＥＰ７］におけるＰマトリックス更新時に作用するパラメータで，これはＰマトリックス更新時に一段階前の値が更新時の値に与える影響の度合いを調整するための係数である．Ｐの値はその定義から分かるように，使用するデータ量の増加するのに逆比例して減少する性質を持ち，(2.14)式のパラメータ修

正ゲインL(t)は式(2.22)のようにP(t)に比例している．したがって，(2.21)式から分かるように，0＜λ＜1の時はλ＝1の時に比べて等価的にPマトリックスの値の減少が少なく，推定するパラメータの修正量をλ＝1の場合より大きくするよう作用する．定常状態にある時系列データ(θ＝一定)を同定するときは，同定初期段階においてはPマトリックス自身が収束していないため，λを1より小さい値にしておくことによってARMAパラメータの変化量を大きくし，パラメータの収束を速める．そしてPマトリックスが小さな値に収束した後は，λ＝1としてPマトリックスの安定化，ひいてはARMAパラメータの安定化を図ることが望ましい．

しかし，非定常な時系列データのパラメータ推定を行うような場合，実状に応じてλの値を設定する必要がある．本研究で用いた計算プログラム内では予めλの初期値としてλ_0，更新のためのファクターとしてμ_λを与え$\lambda(t)$は式

$$\lambda(t) = (\lambda_0 - 1)\mu_\lambda{}^{t-1} + 1 \qquad (2.23)$$

のように定義している．

② 縮約係数．

縮約係数K_tは［STEP5］における式誤差の白色化の段階で，白色化の度合いを調整するための係数である．RPE法において推定値に偏りを生じる原因の一つに［STEP4］で計算される式誤差の非白色性の影響によるものがある．そこで，不偏一致推定量を得るために式誤差$v(t)$を仮想的な白色雑音を入力とするMAモデルで表し，そのMAモデル伝達関数の逆数を白色化フィルタとして用い式誤差を白色化する．このとき時系列データ全体もこのフィルタで補正する過程が［STEP5］である．この原理については Appendix-A に示す．しかし，使用データ量が少ないときやシステムが過渡的状態にある時は，使用する白色化フィルタの伝達関数$\hat{C}(z^{-1})$の精度が悪く，この白色化フィルタを用いると還ってパラメータ推定に悪影響を及ぼし，パラメータの収束を損なうこたが生じる．従って，推定パラメータが大きく変動する同定初期段階などでは白色化を行わないようにするなどの処置が必要となってくる．その処置法の一つとして白色化補正を行う段階で縮約係数K_tを導入して，パラメータ同定の進み具合に応じて白色化の度合いを制御する．本研究で用いた計算プログラム内ではK_tの初期値をK_0＝0 とし，読み込みデータ数に依

存してK_tを更新するための補助パラメータとして$\mu_K(0 \leqq \mu_K \leqq 1)$を用いて，

$$K_t = 1 - \mu_K^t \tag{2.24}$$

のように定義している．

このとき，μ_Kの設定値に対し，縮約係数が０．９９９９９９以上になるまでに必要なデータ数を Fig.2.2 に示す．数縮約係数の意味と，式(2.16)，(2.17)，(2.18)での縮約係数の作用のさせ方については Appendix-B に示す．

ベクトル及びマトリックスの初期設定.

実際に上記逐次パラメータ同定アルゴリズムを使用する際，先ず手続きループに入る前に各ベクトル及びマトリックスの初期設定を設定しておかなければならない．特に，パラメータベクトル$\hat{\theta}(0)$に関しては，先験的な知識によりその近似値が既知である場合には，その値を初期値として設定することによりパラメータ収束も速く，より正確なパラメータ同定を行うことが可能である．しかし，本研究では一般的にそのような値が分からないものとして，以下のように初期値設定を行っている．

$$\hat{\theta}^T(0) = [0,\ 0,\ \cdots,\ 0,\ 0,\ \cdots,\ 0,\ 0]$$
$$L^T(0) = [0,\ 0,\ \cdots,\ 0,\ 0,\ \cdots,\ 0,\ 0]$$
$$P(0) = \alpha \times I \quad (\alpha \text{は任意の正数，} I \text{は単位行列})$$

Ｐマトリックスの初期設定に関して，推定パラメータの偏りを少なくするためにαは十分に大きな値が望ましいとされているが，必ずしもそうではないとの報告もある[7]．本研究では，通常α＝１０００を用いているが，必要に応じて変更が可能である．

なお，本研究で用いている計算プログラムでは，計算時間の短縮とメモリ容量の節約を実現するため，基本的にはＰマトリックスの計算にはＵＤＬ分解法と呼ばれるアルゴリズムを用いており，実際の計算プログラムでは［ＳＴＥＰ７］及び［ＳＴＥＰ８］での計算方法が，式（2.21），(2.22）とは異なる．ただし，得られる結

果は理論的に全く同じである.

2．2　PseudoLinear Regression（ＰＬＲ）法[8]

PseudoLinear Regression（ＰＬＲ）法はＲＰＥ法のアルゴリズムにおいて縮約係数K_tが常に零，すなわち，$\phi(t) = \phi(t)$としたものである．このアルゴリズムでは，本章第１節でも述べたように式誤差の非白色性が強い場合，パラメータ推定値の真値への収束はＲＰＥ法に比べて悪くなり，しかも推定値に偏りを生じることがある．しかし，当然，推定されるＭＡ部$\hat{C}(z^{-1})$の精度及び安定性に依存しないため，ＭＡ部が必ずしも安定な伝達関数とならないような非定常状態の時系列データに対しても有効な逐次オンラインパラメータ同定アルゴリズムである.

2．3　Extended Least Squares（ＥＬＳ）法[6]

Extended Least Squares（ＥＬＳ）法はＲＰＥ法のアルゴリズムにおいて縮約係数K_tが常に零であると同時に，忘却係数λも常に１としたものである．従ってＰＬＲ法の特殊なケースであると言え，ＲＰＥ法の中では最もシンプルな逐次オンラインパラメータ同定アルゴリズムである．この方法は他の方法に比べて最も計算量が少なくてすむ反面，パラメータ推定の対象とするシステムの特性に対する依存性が強いなどの欠点をもつことが報告されている[5].

2．4　ＲＰＥ法における Projection Algorithm

ＲＰＥ法のアルゴリズムにおいて，［ＳＴＥＰ５］は不偏一致推定量を得るという意味で重要な過程であるが，白色化フィルタを形成するＭＡ部$\hat{C}(z^{-1})$が発散フィルタとなっている可能性がある．そこで，縮約係数K_tが常に零となるＰＬＲ法やＥＬＳ法を除いて，式誤差$v(t)$の白色化を行う場合，白色化フィルタを形成するＭＡ部伝達関数$\hat{C}(z^{-1})$が安定，すなわち，

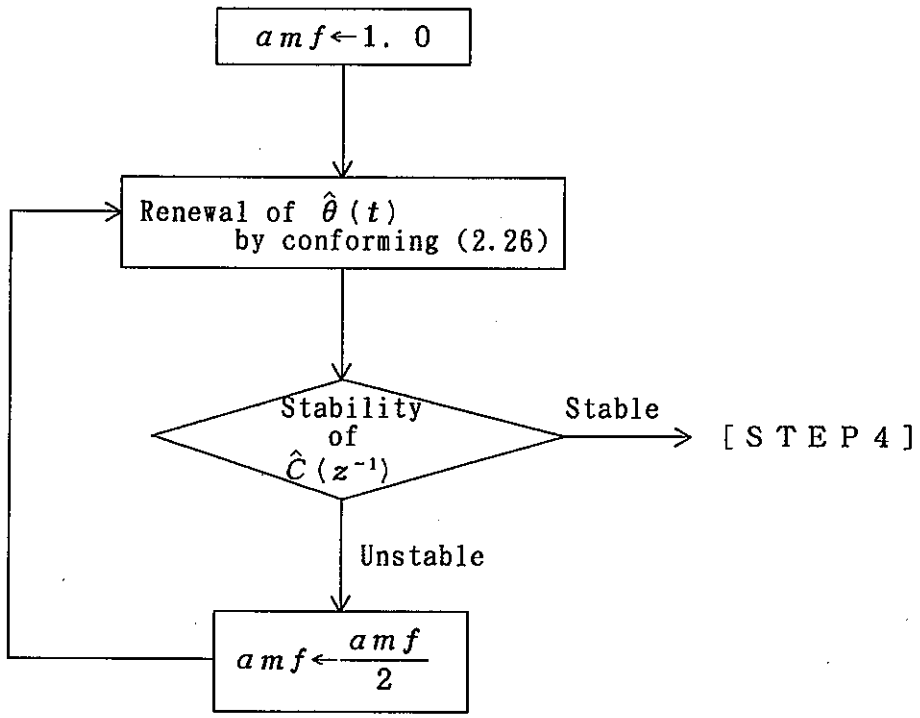
$$\hat{C}(z^{-1}) = 0 \qquad (2.25)$$

の全ての根が単位円の内部に投影(Projection)されることを保証する機能がアルゴリズムの中で必要となる．本研究で用いている計算プログラムでは［ＳＴＥＰ３］を以下のように拡張することによってこれを実現している．

［ＳＴＥＰ３’］

$$\hat{\theta}(t) = \hat{\theta}(t-1)+L(t-1)\cdot e_p \cdot amf \qquad (2.26)$$

ここでamf は調整係数(AjustMent Factor)で，以下に示すフローチャートに従ってその大きさが変更される．



本研究で用いている計算プログラムでは，$\hat{C}(z^{-1})$ の安定性判別に Suhur Cohn の判定条件を用いている[9]．

なお，Projection Algorithm は，式誤差の白色化を行わないＰＬＲ法やＥＬＳ法にも適用できるが，推定値の真値への漸近が極度に遅れる場合があることが本研究の過程で確認されている．

3　近畿大学原子炉での実験及び未臨界度推定手法

実際の未臨界システムとして近畿大学原子炉(UTR-KINKI)を定常運転し，この時の中性子検出器出力揺らぎを観測し，これを本研究の実データとして利用している．ここではその実験の概略と，中性子検出器出力揺らぎ信号の前処理，Ａ／Ｄ変換後のデータ処理法を述べるとともに，ＡＲＭＡモデル同定による未臨界度推定の原理及び解析手法について述べる．

第１節では近畿大学原子炉の概要，第２節では実験に用いた原子炉の炉心体系について，第３節では信号処理系及び測定機材について述べる．第４節には，本研究の解析に用いるマスター時系列データの作成法とサンプリング周波数の選択について，実際のデータ解析結果を用いて説明する．第５節には未臨界度推定の原理を説明し，実験データから得られるＡＲＭＡモデルと理論的に導出されるＡＲＭＡモデルとの相違点について述べる．第６節では同定したＡＲＭＡモデルパラメータを未臨界度に変換する方法について説明する．第７節にはＡＲＭＡモデルから導出される未臨界度の評価基準として用いる原子炉未臨界度の測定法について述べる．

３．１　近畿大学原子炉（ＵＴＲ－ＫＩＮＫＩ）の概要

UTR-KINKIは，９０％濃縮ウランを燃料とした軽水減速黒鉛反射体付き非均質型２分割炉心を持つ熱中性子炉である．最大熱出力は１Ｗで，制御棒は安全棒２本，シム安全棒１本，及び調整棒１本の合計４本で構成されている．原子炉の起動は，全挿入状態にある制御棒の内まず安全棒＃１，安全棒＃２の順で引き抜き，これらの安全棒の引き抜きが完了しないとシム安全棒及び調整棒は駆動することができない構造となっている．また，出力が小さく原子炉を生体遮蔽タンクが覆っている構造

のため運転中に原子炉に近づくことが可能である．したがって，近畿大学原子炉は実験を行う上で次のような利点を有している．

① 小型研究炉であるため，原子炉を占有して実験を行うことが可能である．
② 零出力炉であるためシステム同定にあたって一点炉近似動特性方程式を適用できる．
③ 黒鉛反射対付き２分割炉心であるため，炉心中央部に Dry Access Area があり比較的大型のＣＩＣ及び高圧電源を直接炉の中央部に容易に設置できる．
④ 本研究ではその性質上，電源系統などから進入する外部雑音の低減が重要な課題であるが，この原子炉では検出器の前置増幅器を炉頂に設置可能である上，検出器出力と信号処理系の間の距離を短くできる．したがって，電源系の一点アースの設置，信号ラインの短縮化による雑音低減が可能である．

３．２　炉心体系と中性子検出器の位置

中性子検出器出力揺らぎの観測には直径９ｃｍ，有効長３０ｃｍのＣＩＣ(印加電圧±１８０Ｖ)を用い，黒鉛反射体中央ストリンガー位置に Fig.3.1 に示すように配置された．このとき最も中性子束の大きい中央部の黒鉛ストリンガー（１０ｃｍ×１０ｃｍ×６０ｃｍ）を抜き，更に吸収体となるＣＩＣを配置することになるため原子炉が起動できない場合が生じることがある．そこで，体系の反応度の減少を幾分かでも補償するためＣＩＣの上部に２層にわたって黒鉛ブロックを配置し反応度の減少を補っている．その配置方法を Fig.3.2 に示す．

なお，外部中性子源は Pu-Be １Ｃｉ である．

3．3　信号処理系

炉内に配置したＣＩＣからの出力信号を時系列データに変換する信号処理系の基本ブロック図を Fig.3.3 に示す.

ＣＩＣからの電流信号は Current Amp. により電圧に変換され，その電圧は10^7～10^9倍に増幅されて出力電圧となる．図中の Low Pass Filter は信号のサンプリング時の周波数エリアシング防止用アナログフィルタであり，４次のバタワース特性を有する．High Pass Filter は，非定常運転時のアナログ信号をサンプリングする際の平均値（バイアス）除去用アナログフィルタであり，６次のバタワース特性を有する．Gain Amp. はデジタル信号として信号を記録する際にＰＣＭデータレコーダの１６ビット量子化特性を十分に生かすためのゲイン調整用の線形増幅器である．

オシロスコープおよびＦＦＴアナライザは信号をモニターするためのものである．信号は，「ＳＯＮＹ：ＰＣＭデータレコーダＰＣ－１０８Ｍ」を用いて，１６ｂｉｔ直線量子化のデジタル信号として記録される．

本研究では観測信号から電源のハムを除去するなどの特別な処理をする事なく直接ＡＲＭＡモデル同定するため，電源系の雑音である６０Ｈｚのハム及びその高調波を可能な限り除去する必要があり，Fig.3.4 に示すポイントにアースラインを集中し，一点アースを行っている．

3．4　時系列データのサンプリング

3.4.1　時系列データの取得プロセス

中性子検出器出力揺らぎ信号はカットオフ周波数を５００Ｈｚにセットしたアナログのローパスフィルタで前処理され，ＰＣＭデータレコーダによりサンプリング周波数４８ｋＨｚでサンプリングされた後，ＤＡＴ用テープに記録される．この時系列データはサンプリング周波数が高く，このままの時系列データを用いてＡＲＭＡモデル同定を行って未臨界度を推定することはできない．そこで，このマスター時系列データをディジタルフィルタを用いて計算機処理し，未臨界度推定に適したサンプリング周波数でサンプリングし直したデータに変換する必要がある．この変

換処理を行うシステムを Fig.3.5 に示す．このシステムで変換されたデータは Intel 形式の符号付き２バイト整数型データとしてフロッピーディスクに格納され，ＡＲＭＡ（２，２）モデル同定に用いられる．したがって，以下の節では，サンプリング周波数はデータレコーダのＡ／Ｄ変換周波数ではなく再サンプリングによる周波数変換後の周波数を意味している．

3.4.2 オンラインＡＲＭＡモデルパラメータ逐次推定における ローパスフィルタとサンプリング周波数

アナログ信号をデジタル信号に変換する場合，信号が持つ周波数特性を完全に保持するには，その信号に含まれる最大周波数成分の２倍以上のサンプリング周波数でサンプリングする必要がある．そこで５００Ｈｚにカットオフ周波数を有するローパスフィルタで処理されたアナログ信号をＡ／Ｄ変換する場合，少なくとも１０００Ｈｚのサンプリング周波数でサンプリングされる必要がある．また，一般にはローパスフィルタのカットオフ周波数が５００Ｈｚである場合でも，ローパスフィルタの減衰特性は周波数にたいし一定の割合でしか減衰しないことを考慮し，１２００～１５００Ｈｚのサンプリング周波数でサンプリングされるのが通例である．しかし，必ずしもこのような常識をそのまま適用することがＡＲＭＡモデル同定を行うにあたって適当であるとは言い切れないことを以下で説明する．

本研究においては，エリアシング防止用のローパスフィルタとして４次のバタワース特性を持つフィルタを使用したことは既に述べたが，そのカットオフ周波数が５００Ｈｚの場合，バタワースフィルタのゲイン特性は２００Ｈｚ付近から徐々に減少し始め，５００Ｈｚでゲインは約－３ｄＢとなる．従って１ＫＨｚより充分大きなサンプリング周波数を用いた場合，得られた時系列データには５００Ｈｚ領域にローパスフィルタの折れ点周波数付近の特性がそのまま現れることになる．このような時系列データをＡＲＭＡモデル同定すると，未臨界原子炉のシステムとローパスフィルタの複合形のシステムを同定することになり，原子炉本来のシステムモデル次数にフィルタのモデル次数を加えた次数のモデルが必要となる．

したがって，フィルタ特性が顕著に現れているこの様な時系列データに対しては，未臨界原子炉システムの理論的モデル次数(２，２)のＡＲＭＡモデルでは未臨界度推

定が不可能となる．この問題を回避する一つの方法として，ＡＲＭＡモデルの次数を高くして，初めから未臨界原子炉とローパスフィルタの複合形のシステムとしてパラメータ同定を行い，高次ＡＲＭＡモデルのパラメータから必要な情報を取り出すことが考えられる。しかし，このような方法は計算量を増大させる上，本研究の最終目的である非定常時のパラメータ推定の過渡特性を悪くする[10]．

そこで，本研究ではサンプリング周波数をナイキスト周波数よりやや低い８８８．８８８Ｈｚ(Sampling Time = 0.001125 sec)に設定し，エリアシングを意識的に発生させ，ローパスフィルタの減衰特性とエリアシングによる周波数成分の折り返し特性とを互いに相殺させ，スペクトル的にローパスフィルタの特性が時系列データに現れないようにする方法を採用している．

次項では，信号のスペクトルにローパスフィルタの特性が残った場合，及びエリアシングを生じた場合の未臨界度推定について述べる．

3.4.3 実データによる解析

この項では，パラメータ同定を行う時系列データの周波数特性に①エリアシング防止用のローパスフィルタの特性が現れている場合，②サンプリング周波数が低すぎてエリアシングを生じた場合，③3.4.2 項で述べたようにローパスフィルタの特性とエリアシングの影響を互いに相殺させた場合，の三つの場合についてそれぞれの未臨界度推定がどのようになるかを比較・検証する．

未臨界度が０．２８９＄で定常運転された原子炉から得た中性子検出器・出力揺らぎを，上記の三つの場合に対応してそれぞれ１０００Ｈｚ，８００Ｈｚ，８８８．８８８…Ｈｚの周波数でサンプリングし，それぞれの時系列データに対し

① ＥＬＳ法(Projection = OFFF)によるＡＲＭＡ(２，２)モデル同定

② ＲＰＥ法(μ_K = 0.999, Projection = OFF)によるＡＲＭＡモデル(１，１)同定

を行い未臨界度推定を行った．この結果を以下にまとめる．ただし，それぞれの時系列データは低周波成分を除去するために０．０２Ｈｚにカットオフ周波数を有する１次のバタワース特性デジタルフィルタを通してある．

［１］　ＥＬＳ法によるＡＲＭＡ(２，２)モデル同定の結果

ＡＲＭＡ(２，２)モデルは本研究の中心となるモデルであるが，Fig.3.6 に示すようにいずれの場合も未臨界度の推定に問題はないと言える．推定値が安定した状態でにある２４０，０００データを使用して得られたパラメータによるＡＲＭＡ(２，２)モデルのパワースペクトラム密度（ＰＳＤ）と，高速フーリエ変換法(ＦＦＴ)（1024 点によるＰＳＤの６００サンプル平均）によるＰＳＤとを Fig.3.7 に示す．エリアシングを生じている場合（Fig.3.7 (b)）も，ローパスフィルタの特性が現れている場合（Fig.3.7 (c)）も，ＦＦＴ法により求められたＰＳＤとＡＲＭＡ(２，２)モデルによるＰＳＤは非常に良く一致している．

８００Ｈｚサンプリングではエリアシングの影響が最も強く影響する３００～４００Ｈｚ付近の周波数成分の折り返しが，また１０００Ｈｚサンプリングでは３００Ｈｚ付近からのローパスフィルタの折れ点の特性が正確にＡＲＭＡ(２，２)モデル同定されていることになる．これは，本来未臨界原子炉のシステムをＡＲＭＡモデルで表現するには２次のモデルが必要であるにもかかわらず，実際には未臨界度０．２８９＄の理論モデルのＰＳＤ（Fig.3.8）に見られる０．００１Ｈｚ付近の低周波成分の盛り上がりは同定出来ないことに起因している．すなわち，未臨界システムの理論的ＡＲＭＡモデルは（２，２）モデルであるが，採用しているサンプリング周波数で得られるデータには０．０１Ｈｚ以下の情報はほとんど含まれていないため，未臨界システムを表現するモデルとしては高周波側の折れ点で代表される１次システムでよいことになり，残りの１次分をエリアシングによる周波数成分の折り返し，または，ローパスフィルタの周波数特性の同定に使用していると考えることができる．従って，全体のスペクトルに対しその影響の度合いが小さい場合には，多少のエリアシング効果や，ローパスフィルタの特性が時系列データに含まれていてもＡＲＭＡ(２，２)モデルで未臨界度推定が可能であることになる．

しかし，あくまで影響の度合いが小さいことが前提であって，信号のスペクトルにローパスフィルタの折れ点の特性が顕著に現れる場合は，ローパスフィルタの特性を表現するために次数の高いモデルが必要となり，全体で５次か６次のＡＲＭＡモデルとして同定することが必要となる．しかし，この様に複合形のシステムとみなして同定する場合，仮定したモデルの次数が不当に低いと次の［２］で述べるような問題が生じる．

［２］　ＲＰＥ法によるＡＲＭＡ(１，１)同定の結果

［１］で述べたように，実際には未臨界原子炉システムの低周波領域の盛り上がり（Fig. 3.8 参照，０．００１Ｈｚ付近）は同定できないことから，未臨界体系は原理的には１次システムでモデル化し，その折れ点周波数から未臨界度を推定することが可能である．しかし，１０００Ｈｚサンプリングの場合，Fig.3.9 に示すようにローパスフィルタの特性が時系列データに含まれていると正確な未臨界度推定は不可能となる．［１］と同様に推定値が安定している２４０，０００データ使用後の推定パラメータから導かれるＡＲＭＡ（２，２）モデルのＰＳＤをＦＦＴ（1024 点計算，６００サンプル平均）で計算したＰＳＤと比較したものを Fig.3.10 に示す．ローパスフィルタの特性とエリアシングの影響を互いに相殺させた８８８．８８８…Ｈｚサンプリング（Fig.3.10 (a)）の場合では，ＡＲＭＡ(１，１)モデルとＦＦＴによるスペクトルはかなり良い一致を示している．しかし，ローパスフィルタの特性が現れている Fig.3.10 (c) の場合では，ＡＲＭＡ(１，１)モデルではローパスフィルタによるスペクトルの変化を十分モデルの中に表現し切れず，ＡＲＭＡ(１，１)モデルの折れ点周波数は未臨界原子炉システムの本来の折れ点周波数から大きくずれてしまう．しかし，エリアシングが生じている場合でも，Fig.3.10(b) のようにエリアシングによる折り返しの周波数成分が小さい場合は，未臨界度推定に与える影響も小さい．

以上の点を考慮すると，ＡＲＭＡモデル同定によるパラメータ推定を行う場合はサンプリング周波数の選択に対し，一般のデジタル信号処理の常識を超えた配慮が必要であると言える．

3．5　未臨界度推定の原理とモデリング

3.5.1　未臨界度推定の基本原理

定常状態にある未臨界原子炉システムは，

① 遅発中性子一群一点炉近似動特性方程式に従う．

② 中性子密度と先行核密度の揺らぎ及び検出器に混入する雑音がガウス白色色過程である．

という仮定の下で，次式

$$y(t)+\sum_{i=1}^{2}a_i y(t-i) = e(t)+\sum_{j=1}^{2}c_j e(t-j) \qquad (3.1)$$

で定義されるＡＲＭＡ(２，２)モデルで表現することができる．ここで$y(t)$はランダムな時系列，$e(t)$はガウス白色時系列である．

このモデルにおける伝達関数は，

$$H(z^{-1}) = \frac{1+c_1 z^{-1}+c_2 z^{-2}}{1+a_1 z^{-1}+a_2 z^{-2}} \qquad (3.2)$$

である．

一方，未臨界原子炉システムの周波数応答は Fig. 3.8 に見られる高周波側１次遅れを表す１次システムで近似でき，その折れ点角周波数ω_cは炉物理定数として

l　：中性子寿命

γ　：遅発中性子が減速過程での消滅を免れて熱中性子になる確率　／
即発中性子が減速過程での消滅を免れて熱中性子になる確率

β　：遅発中性子の割合

ρ_0　：未臨界度

を与えた場合，

$$\omega_c = \frac{\gamma\beta - \rho_0}{l} \tag{3.3}$$

で表される．ここで，l，γ，β はそのシステムに固有な値であるから，周波数応答の高周波側の折れ点周波数と未臨界度は１対１に対応する．

従って，中性子検出器信号の揺らぎを第２章で述べたＡＲＭＡモデル同定アルゴリズムを用いてパラメータ推定し，モデルのスペクトルの折れ点周波数に対応するシステム極から未臨界度を推定することができる．ただし，実際にはＡＲＭＡモデル同定を行う時系列データには，エリアシングによる周波数成分の折り返しが皆無とは言えず，幾分かの誤差を含む．

3.5.2 実データによるモデルと理論モデル

実際のデータをＡＲＭＡ(２，２)モデル同定して得たモデルのパラメータは理論モデルのパラメータとは全く異なる．これは，用いるデータ量の点かもともと情報量が乏しい上に，実際の信号の処理段階でわざわざ低周波成分を除去しているため，理論モデルの低周波側の盛り上がりが実データからは同定できないことに起因している．従って，同定したモデルから得られるスペクトルは，理論的に導かれるモデルから得られるスペクトラムとは低周波部分で異なる．

その，一例として，3.4.3 項で用いた未臨界度０．２８９＄における８８８．８８８…Ｈｚサンプリングの時系列データを，ＥＬＳ法によりＡＲＭＡ(２，２)モデル同定した場合の２４０，０００データ使用後の推定パラメータから得られるＰＳＤと，理論モデルから導出されるＰＳＤとの比較を Fig.3.11 に示す．なお，理論モデルの構築については Appendix C を参照されたい．

Fig.3.11 において明らかに０．１Ｈｚ以下でのＰＳＤは異なっているが，未臨界度推定の基本情報源となる高周波側での折れ点周波数及び１次遅れ特性はほぼ完全に一致していることから，このＡＲＭＡモデル同定手法による未臨界度推定が妥当であるといえる．ただし，個々のパラメータが持つ物理的な意味は失われていると考えなければならない．

3.5.3 ＡＲＭＡモデルの次数選択

実際のデータをＡＲＭＡモデル同定することによりシステムの未臨界度を推定する場合、ＡＲＭＡモデルの次数をどのように選ぶべきかを 3.5.1 及び 3.5.2 節の結果から再考察する.

理論的ＡＲＭＡモデルのＰＳＤ（Fig.3.8参照）では０.０１Ｈｚ以下の低周波領域に周波数特性の盛り上がりが見られるが，この様な特性を時系列データから同定しようとする場合，当然それに対して適切なサンプリング周波数を選択しなくてはならない．既に述べたとおり我々は理論的ＡＲＭＡモデルＰＳＤの高周波側の折れ点周波数に着目しており，この部分をよく表現するＡＲＭＡモデルの選択が課題である． Fig.3.11 に示したように，適当な信号処理を施したデータならば１次のＡＲＭＡモデルでも高周波側の周波数特性を同定するのには十分であるといえる．このことはむしろ，ＡＲＭＡモデル同定過程において必要とする計算機メモリ量及び計算量の観点からすれば，２次のモデルと比較して約４分の１で済む１次モデル同定の方が優れているとも言える．しかし，１次のモデル同定には次のような問題点がある.

① 多くの解析結果から判断すると，ＲＰＥ法による１次のＡＲＭＡモデル同定においてはパラメータの推定値に偏りを生じることがある．その場合でも２次以上のＡＲＭＡモデルであれば，パラメータ自体に偏りが生じても得られたモデルの２つのシステム極が互い情報を分け合い，結果的に未臨界度に対応したシステム極の偏りが正しい値から大きくずれることがない．それに対して，１次のモデルではパラメータの偏りは直接システム極の偏りとなって現れるためシステム極から未臨界度を推定する本手法には不適切である．3.4.3項 でＡＲＭＡ(２，２)モデル同定とＡＲＭＡ(１，１)モデル同定による比較を行ったが Fig.3.6 と Fig.3.9 を比較して分かるように，１次のモデルの方が推定値が偏っている．２次のモデルではＥＬＳ法を用い，１次のモデル同定ではＲＰＥ法を用いており，本来ならばＥＬＳ法による２次のモデルの推定の方がが偏りの影響を受け易いにも関わらず，１次のモデルの方に偏りが現れている結果となっており，本手法のようにシステム極を重視する場合，多少冗長であっても２次のモデル同定を行うことが必要であると言える.

② 非定常時系列データのＡＲＭＡモデルパラメータ同定の場合，ＡＲＭＡ（１，１）モデル同定のパラメータ変動に対する追随性が２次モデル同定よりも緩慢である[6]．これは，システム自体の変化を２つパラメータだけで表現しようとするため，システムの細かい変化を表現できなためである．

３．６　ＡＲＭＡモデルからの未臨界度の導出

同定されたＡＲＭＡモデルから未臨界度を推定するには，ＡＲＭＡモデルのＡＲパラメータから周波数応答の高周波側の折れ点周波数に対応するシステム極を計算し，それを未臨界度に較正する必要がある．

そのためには，色々な未臨界度に対する理論的ＡＲＭＡ（２，２）モデルパラメータを計算し，それぞれの未臨界度に対するシステム極（(3.3)式に対応する極）を計算する．そして各未臨界度に対する極を用いてシステム極から未臨界度へ変換するための較正曲線を作成する．Table 3.1 に理論モデル構築の際に用いた炉物理定数をに示す．

また，極－反応度較正曲線はサンプリング周波数に固有のものであり，Fig.3.12 にサンプリング周波数

８８８．８８８…Ｈｚ(Sampling Time = 0.001125 sec)
４４４．４４４…Ｈｚ(Sampling Time = 0.002250 sec)
２２２．２２２…Ｈｚ(Sampling Time = 0.004500 sec)

に対応する極－反応度較正曲線を示す．

Table 3.1 Constant of reactor physics.

l	1.35×10^{-4} (sec)	
$\gamma\beta$	0.0064	$\beta = \sum_{i=1}^{6}\beta_i$
λ	0.077 (sec^{-1})	$\lambda = \frac{\sum_{i=1}^{6}\beta_i}{\sum_{i=1}^{6}\frac{\beta_i}{\lambda_i}}$
ν	2.54 ($fission^{-1}$)	

Remarks : (1) "l" neutron life time.
(2) "λ" decay constant of precursors.
(3) "ν" number of neutrons produced per fission.

3．7 ＡＲＭＡ同定から推定された未臨界度の評価

本研究でＡＲＭＡモデル同定によって推定された未臨界度の推定精度の評価方法について述べる．

原子炉の深い未臨界度を精度よく推定する手法が整っているとは言えない現状ではあるが，ここでは原子炉は一点炉近似動特性方程式に従うものと仮定し，以下の方法により得られた未臨界度を基準としてＡＲＭＡモデルパラメータ推定による未臨界度の評価を行っている．

［ＰＲＯＣＥＳＳ１］

１Ｗ臨界状態からロッドドロップ法により制御棒の等価反応度価値の測定を行う．ロッドドロップ法により得られた制御棒の等価反応度価値の測定測定結果をTable 3.2 に示す．

なお，未臨界度推定に用いる時系列データの取得時と，ロッドドロップ法にる制御棒反応度価値測定時との炉心状態を同一にするため，中性子検出器としては未臨界度推定に用いたＣＩＣをそのまま用い，この出力（ＤＣ成分）をＡ／Ｄ変換して

データレコーダに記録して，反応度の一般的な計算方法である

$$\rho = \frac{n_0 \sum_{i=1}^{6} \frac{a_i}{\lambda_i}}{\int_0^{\infty} n(t)\,dt} \qquad (3.4)$$

に従って反応度を算出した.

Table 3.2 Equivalent reqctivity worth of the control rod.

Control rod	Equivalent reactivity worth [$]
Shim safety	0.7129
Shim safety + Safety #1 + Safety #2	2.259

［ＰＲＯＣＥＳＳ２］

Fig.3.13 には臨界時の制御棒体系，臨界体系からシム安全棒のみ全挿入した体系(未臨界体系１)，そして臨界体系からシム安全棒と安全棒＃１・＃２を全挿入した体系(未臨界体系２)を示している.

［ＰＲＯＣＥＳＳ１］で得た制御棒等価反応度価値の測定結果から，未臨界体系１の未臨界度は０．７１２９$，未臨界体系２の未臨界度は２．２５９$であるとの結果を得た.

また，未臨界体系１でのＣＩＣ出力レベル(Current Amp. の DC 出力)は１．９２Ｖ，未臨界体系２では０．６４Ｖであった.

体系が，一点炉近似に従うとき，外部中性子源が存在する場合の未臨界原子炉定常運転時における出力Nは，中性子寿命をl (sec)，中性子源強度をS，未臨界度をρ_0とすれば，

$$N = \frac{l}{(\rho_0)} S \qquad (3.5)$$

で表される．ここでl，Sは定数であるから，未臨界体系１及び未臨界体系２での未臨界度と出力レベルを式(3.5)に代入することによりlSの値を求めることができ，その平均をとると，lS＝１．４であった．

この結果を用いて，各未臨界体系における出力レベルから式(3.5)に従ってリファレンス未臨界度を設定している．

上記の方法により求めたＣＩＣ出力レベルに対するリファレンス未臨界度較正曲線を Fig.3.14 に示す．ただし，平均の段階で２．７５％の平均誤差を含み，それ以外にも認識不可能な誤差が含まれることを考慮すると，リファレンス未臨界度自体その精度は必ずしも良くないと思われる．

従って，以後の議論では上記の方法によって算出されるリファレンス未臨界度を参考にＡＲＭＡ同定から推定される未臨界度を評価する．

4　信号の低周波成分と未臨界度推定

この章では，未臨界定常状態にある近畿大学原子炉から取得した時系列データの解析を通して，信号に含まれる低周波成分が未臨界度推定値にどのような影響を及ぼすかを明らかにする．ここでは未臨界原子炉モデルとしてＡＲＭＡ(２，２)モデルを用い，このモデルのパラメータ推定から未臨界度の推定を行う．

第１節では，先ずサンプリング周波数が未臨界度推定にどのような影響を及ぼすかを調べるために，サンプリング周波数８８８．８８８…Ｈｚのマスター時系列データを等価的に４４４．４４４…Ｈｚ及び２２２．２２２…Ｈｚサンプリング時系列データに変換する方法について説明する．

第２節では，時系列データ中に含まれる低周波成分を除去するために導入したハイパスフィルタのカットオフ周波数の選択が未臨界度推定に与える影響について述べる．また，ハイパスフィルタがＤＣ成分の除去だけでなく不要な低周波成分を除去するという点で未臨界度推定に必要であることを説明する．

第３節では，時系列データ取得時のサンプリング周波数が未臨界度の推定時間に与える影響，特に高いサンプリング周波数でＡ／Ｄ変換して単位時間当たりに得られる情報量を見かけ上多くすることが，未臨界度推定にどのような影響を及ぼすかについて検討する．

第４節では，Ａ／Ｄ変換時の量子化誤差が未臨界度推定に与える影響を明らかにし，適正な推定精度を得るために必要なビット数について検討する．

４．１　マスター時系列データからのサンプリング周波数変換 －基本時系列データの作成－

第３章でも述べたように，近畿大学原子炉の実験では時系列データの取得に際し，アナログ信号段階で５００Ｈｚにカットオフ周波数を有するローパスフィルタを通

し，サンプリング周波数８８８．８８８…Ｈｚ(Sampling Time = 0.001125 sec)でマスター時系列データを作成している．ここでは，マスター時系列データから本節以降の解析に用いるサンプリング周波数４４４．４４４…Ｈｚ(Sampling Time = 0.002250 sec)データ，及び２２２．２２２…Ｈｚ(Sampling Time = 0.004500sec)時系列データを作成するサンプリング周波数変換法について説明する．

［１］

次の節で述べるマスター時系列データ分周サンプリングに際し発生する可能性があるエリアシングを防止するため，マスター時系列データをバタワース特性を持つディジタルローパスフィルタを通す．このフィルタの特性を Table 4.1 に示す．

Table 4.1 Characteristic of Low-Pass Filter for anti-aliasing.

	Cutoff frequency [Hz]	Filter order
for conversion from 888.888…Hz to 444.444…Hz	222.222…	4
for conversion from 888.888…Hz to 222.222…Hz	111.111…	4

［２］

アナログフィルタでは回路を構成する素子の誤差の影響などを受け，周波数特性が設計値を完全に満足することはできないが，計算機を用いて実現するディジタルフィルタは正確なカットオフ周波数と，急峻な減衰量が実現できる（Fig.4.1.a，Fig.4.1.b）．そこで， Table 4.1 に示したローパスフィルタで処理された８８８．８８８…Ｈｚサンプリングのマスター時系列データから，ディジタル信号データを２回に１回の割合で分周サンプリングする事により，４４４．４４４…Ｈｚサンプリングの時系列データを，同じく４回に１回の割合で分周サンプリングする事で２２２．２２２…Ｈｚサンプリングの時系列データを作成することができる．

以後，マスター時系列データ及び，上記のプロセスによって作成された時系列データを基本時系列データと呼ぶ．本節において未臨界度推定に用いる基本時系列データ値の特性と，以後の便宜上のためにデータの名前を Table 4.2 に示す．

Table 4.2 Reference table for the fundamental time series.

Reference subcriti-cality [$]	Sampling frequency [Hz]	Name of the fundamental time series	Minimum	Maximum	Bias
2.19	888.888…	ID003.888	-16908	20484	1907.0
	444.444…	ID003.444	-13562	16556	1949.5
	222.222…	ID003.222	-11719	13510	1947.4
1.25	888.888…	ID005.888	-19543	16626	255.13
	444.444…	ID005.444	-17695	15068	187.63
	222.222…	ID005.222	-15500	13144	188.98
0.289	888.888…	ID010.888	-19814	20085	807.33
	444.444…	ID010.444	-21768	18232	736.60
	222.222…	ID010.222	-18147	18744	743.15

4．2 低周波成分除去用ハイパスフィルタの選択と未臨界度推定

基本的な時系列データの取得にあたっては，エリアシング防止が重要であることは言うまでもないが，ＡＲＭＡモデル・オンラインパラメータ推定アルゴリズムによるシステム同定を行う場合，信号のバイアス成分（ＤＣ成分）並びに不必要な低周波成分を除去し，要求されるパラメータ推定に必要な情報のみを抽出する事が重要である．

炉内に配置された中性子検出器からの信号は炉出力としてのＤＣ成分がベースにあるため，ＡＲＭＡモデル・オンラインパラメータ推定を精度良く行うためにはこのＤＣ成分(バイアス成分)を可能な限り除去しなければならない．出力が一定で安定した定常状態の場合は，バイアス成分に見合う逆バイアスをかけることでこの問題に対処することも可能である．しかし，本研究が目的としている臨界安全監視モニターでは，そのバイアス成分自身が刻々と変化する場合にも適用できるアルゴリズムでなくてはならないため，逆バイアスを印加することによるバイアス成分除去法は実用的ではない．

そこで，ここでは検出器からの信号をハイパスフィルタに通すことによって低周

波成分を除去する方法を採用する．この方法は単にバイアス成分を除去するだけでなく，未臨界度推定に不要な低周波成分を減衰させるという目的にも効果がある[5]．しかし，パラメータ同定される信号をハイパスフィルタに通すということは，第３章第４節第２項で述べたローパスフィルタの場合と同様，本来同定すべき未臨界原子炉のシステムにハイパスフィルタの特性が加わった複合形システムのパラメータ推定となる恐れがある．

昨年度の報告書で述べたようにこれまでの研究では，「同定する未臨界原子炉システムの高周波側の折れ点周波数の１／１０～１／１００にカットオフ周波数を有するハイパスフィルタを導入する事により比較的良い未臨界度推定値が得られる」という大略的な知見を得ており，本節では原子炉から得られた定常時系列データを用いて，ハイパスフィルタの特性が未臨界度推定の精度・推定時間に与える影響を，時系列データのサンプリング周波数と関連させて詳細に検討する．

4.2.1 データ解析の概要

この項では，Table 4.2 に示した各基本時系列データに対して，ハイパスフィルタのカットオフ周波数を変化させて低周波成分を除去し，それぞれの時系列データに対しＥＬＳ法によるＡＲＭＡ(２，２)モデル同定を行い，未臨界度の推定を行う．

ハイパスフィルタには１次のバタワース特性を有するディジタルフィルタを用いている．Table 4.3，Table 4.4，Table 4.5 に解析に用いたハイパスフィルタのカットオフ周波数とフィルタ通過後の時系列データのバイアスを示す．

ＩＤ００３シリーズのデータに対する未臨界度推定結果を Fig.4.2.a，Fig.4.2.b，Fig4.2.cに，ＩＤ００５のシリーズに付いては Fig.4.3.a，Fig.4.3.b，Fig.4.3.c に，ＩＤ０１０のシリーズ付いては Fig.4.4.a，Fig.4.4.b，Fig.4.4.c にそれぞれ示す．ただし，グラフを見やすくするために Table 4.3，Table 4.4，Table 4.5 で表記したハイパスフィルタのうち一部の結果の表示は割愛している．

Table 4.3 Cutoff frequency of High-Pass filter and the corresponding bias of the time series(ID003 series).

ID003.888		ID003.444		ID003.222	
Cutoff freq. [Hz]	Bias	Cutoff freq. [Hz]	Bias	Cutoff freq. [Hz]	Bias
0.005	782.26	0.02	110.40	0.04	30.820
0.01	439.19	0.04	52.403	0.06	20.454
0.02	228.35	0.06	34.132	0.08	15.171
0.04	119.89	0.08	25.461	0.10	11.960
0.06	81.981	0.10	20.455	0.20	5.5739
0.08	62.114	0.20	10.719	0.40	2.6761
0.10	49.800	0.40	5.5334	0.60	1.8176
0.20	24.299	0.60	3.6089	0.80	1.4069
		0.80	2.5982	1.00	1.1597
		1.00	1.9904	2.00	0.66345

Table 4.4 Cutoff frequency of High-Pass filter and the corresponding bias of the time series(ID005 series).

ID005.888		ID005.444		ID005.222	
Cutoff freq. [Hz]	Bias	Cutoff freq. [Hz]	Bias	Cutoff freq. [Hz]	Bias
0.005	102.98	0.02	29.568	0.04	4.3688
0.01	55.641	0.04	16.810	0.06	3.1576
0.02	28.306	0.06	11.725	0.08	2.5584
0.04	14.503	0.08	9.2271	0.10	2.1957
0.06	9.2132	0.10	7.7962	0.20	1.3955
0.08	6.4955	0.20	5.0850	0.40	0.87895
0.10	4.9138	0.40	3.5657	0.60	0.70218
0.20	1.7505	0.60	2.8663	0.80	0.62528
		0.80	2.4111	1.00	0.58678
		1.00	2.0916	2.00	0.50188

Table 4.5 Cutoff frequency of High-Pass filter and the corresponding bias of the time series(ID010 series).

ID010.888		ID010.444		ID010.222	
Cutoff freq. [Hz]	Bias	Cutoff freq. [Hz]	Bias	Cutoff freq. [Hz]	Bias
0.005	141.46	0.02	8.8243	0.04	9.0910
0.01	52.273	0.04	6.5841	0.06	6.7565
0.02	26.272	0.06	5.6998	0.08	5.2169
0.04	9.3434	0.08	4.8337	0.10	4.1501
0.06	-0.3954	0.10	4.0350	0.20	1.7035
0.08	-5.9669	0.20	1.5919	0.40	0.36150
0.10	-9.2496	0.40	1.7708	0.60	0.006750
0.20	-13.516	0.60	-0.070899	0.80	-0.093033
		0.80	-0.089799	1.00	-0.14635
		1.00	-0.068665	2.00	-0.12300

4.2.2 ハイパスフィルタの選択と推定値における定性的知見

Fig.4.2.a～ Fig.4.4.c から，ハイパスフィルタのカットオフ周波数のわずかな違いが，未臨界度の推定値の精度に及ぼす影響は，極めて大きいものであることが確認される．定性的には次のようなことが言える．

① ハイパスフィルタのカットオフ周波数が低すぎて十分なバイアス除去ができない場合は，実際よりも浅く未臨界度を推定する．

② ハイパスフィルタのカットオフ周波数が適正値より高い場合，バイアスの低減効果は大きいが推定値は実際よりも深い方へ偏る．これはハイパスフィルタ自身の影響が強くなり，本来同定すべき未臨界原子炉のシステムがハイパスフィルタとの複合形のシステムとなっていることが原因であると考えられる．

③ サンプリング周波数の観点から推定状況を考察した場合，計算に用いるデータ数が同じであれば，当然サンプリング周波数が低い程より低周波の情報を抽出できることになり，低周波除去用のハイパスフィルタの特性に対

して敏感になる.

4.2.3 ハイパスフィルタの適正カットオフ周波数

各基本時系列データに対して適正と思われるハイパスフィルタのカットオフ周波数をTable 4.6に示す.

Table 4.6 Reasonable cutoff frequency of High-Pass filter to the fundamental time series.

Time series	Cutoff freq.[Hz]	Time series	Cutoff freq.[Hz]	Time series	Cutoff freq.[Hz]
ID003.888	0.02	ID005.888	0.02	ID010.888	0.02
ID003.444	0.2	ID005.444	0.2	ID010.444	0.2
ID003.222	0.4	ID005.222	0.4 or 0.6	ID010.222	0.2

Table 4.2より，各基本時系列のシリーズにおける初期バイアスはＩＤ００３，ＩＤ０１０，ＩＤ００５の順に大きいが，初期バイアスに応じてハイパスフィルタのカットオフ周波数を選択することよりも，Table 4.6 はむしろサンプリング周波数との関連によるハイパスフィルタのカットオフ周波数の選択の必要性を示している.

Table 4.6 に示したハイパスフィルタの適正カットオフ周波数の決定条件として，元の時系列データに対するハイパスフィルタのＤＣ成分低減率(DC component reduction ratio)の面から考察を加える．まずＤＣ成分低減率を，

$$\text{ＤＣ成分低減率} = \frac{\text{ハイパスフィルタ通過後のバイアス}}{\text{基本時系列の初期バイアス(Table 4.2に表示)}} \qquad (4.1)$$

と定義する.

Fig.4.5に，８８８.８８８…Ｈｚ，４４４.４４４…Ｈｚ，２２２.２２２…Ｈｚの各サンプリング周波数に対するＩＤ００３，ＩＤ００５，ＩＤ０１０の各シリー

ズのＤＣ成分低減率を示す．これらの図はカットオフ周波数が高くなるにつれて効果的にバイアスは低減してくるが，Table 4.6に示した適正カットオフ周波数を超えると，それ以上にカットオフ周波数を上昇させてもＤＣ成分低減率はさほど改善されなくなることを示している．

このことから，低周波成分除去用のハイパスフィルタはに付いては，次の２点を考慮する事が必要である．

① システム同定の立場からは，ハイパスフィルタ自身の存在は複合形のシステム同定を必要とする可能性があるという点で好ましくなく，サンプリング周波数に比べてカットオフ周波数はなるべく低周波側で，かつフィルタ次数も低次であることが好ましい．

② を用いたＡＲＭＡモデルパラメータ逐次推定法による未臨界度推定を行う場合，時系列データのバイアスを一定以上除去する必要がある．

したがって，ハイパスフィルタの最適なカットオフ周波数は，「適正なＤＣ成分低減率を得られる保証の下で，可能な限り低い周波数」という条件から決定されるべきであり，Fig.4.5 に示すようなＤＣ成分低減率曲線において，ＤＣ成分低減率Rをハイパスフィルタのカットオフ周波数f_cの関数としたとき，その微係数，

$$\frac{dR(f_c)}{df_c} \tag{4.2}$$

が大きく変化する近傍のカットオフ周波数を選択することにより，より適正な推定精度を維持できる．

4.2.4 不要低周波成分除去という側面からのハイパスフィルタの必要性

ＡＲＭＡモデル同定による未臨界度推定を行うにあたって，バイアスの除去という側面からのハイパスフィルタの必要性は上に述べてきた．この項では，時系列データ中に含まれるＤＣ成分以外の不要な低周波成分を除去するという側面からハイ

パスフィルタの必要性を述べる.

バイアスの値が零ということが保証されている定常シミュレーション時系列データにおいても，適当なカットオフ周波数を有するハイパスフィルタを導入することによって未臨界度推定能力が改善されることがこれまでの研究で既に明らかになっている[6]．本研究では，実データにおいても単なるＤＣ成分の除去だけではなく，ハイパスフィルタの導入が，不要な低周波成分を同時に除去することによって未臨界度推定性能に貢献することを示す.

時系列データからＤＣ成分のみを除去する方法で最も確実なのは，その時系列データの平均値を求め，各データから引くという作業を行うことである．ただしこの方法は，ＤＣ成分が一定値の定常時系列データにおいてのみ適用可能である．この方法によって各基本時系列データを処理し，再度バイアスを計算した結果をTable 4.7に示す．なお，以後の説明の便宜上この方法を平均値除去法(Zero Mean Value Method)と呼ぶ.

Table 4.7 Bias of the fundamental time series processed by Zero Mean Value Method.

Time series	Bias	Time series	Bias
ID003.888	-5.99098×10^{-11}	ID005.222	-9.96937×10^{-12}
ID003.444	8.27373×10^{-12}		
ID003.222	5.10935×10^{-12}	ID010.888	-4.15994×10^{-11}
		ID010.444	-4.83144×10^{-11}
ID005.888	1.63704×10^{-11}	ID010.222	-2.03885×10^{-11}
ID005.444	-3.43012×10^{-11}		

Table 4.7に示す結果では，各時系列データのバイアスは必ずしも零とはなっていない．これは定常状態の時系列データを取得したとはいえ，実際の原子炉出力はＤＣ成分そのものが多少の変動をしているためであると考えられる．しかし，一応にハイパスフィルタの処理によるバイアス(Table 4.3, Table 4.4, Table 4.5)と比較して相当小さな値が実現できる.

基本時系列データを，平均値除去法と，Table 4.6に示した適正カットオフ周波数を有するハイパスフィルタを用いて処理する．この各時系列データをＥＬＳ法を用

いてＡＲＭＡ(２，２)モデル同定し，未臨界度の推定を行った結果を，ＩＤ００３シリーズはFig.4.6.aに，ＩＤ００５シリーズはFig.4.6.bに，ＩＤ０１０シリーズはFig.4.6.cに各サンプリング周波数ごとに示す．なお，参考のために低周波成分の除去を行わずに各基本時系列データをそのまま同定した場合の結果も含まれている．

これらの結果から，次のような知見を得ることができる．

① 平均値除去法によって単にＤＣ成分のみを除去した場合に比べて，適正カットオフ周波数を有するハイパスフィルタで処理したものの方が未臨界度を精度良く推定できる．このことは特に，サンプリング周波数が低い場合において顕著である．

② 未臨界度が浅い場合すなわち炉出力自体が高い場合，両者には大きな違いは認められない．

上記の知見から，出力レベルが低く，かつサンプリング周波数が低い場合に，ＤＣ成分以外の低周波成分が未臨界度推定に与える影響が大きくなることが理解される．未臨界度推定の原理に立ち返ると，時系列データに含まれる情報として，未臨界度に対応する高周波側折れ点周波数近傍での１次遅れの特性が信号のＰＳＤに顕著に現れていることが精度の良い未臨界度推定をするための必要条件である．逆に，低周波側で周波数特性に顕著な特徴がある場合，ＡＲＭＡモデル同定ではその特徴もモデルとして同定しようとするため，本来の目的である未臨界原子炉システムの高周波側の折れ点の情報は不正確となる．以下に，特に時系列データのサンプリング周波数が低い場合にその影響を受けやすいことを説明する．

例えば，０．２Ｈｚ付近に何らかの特徴があるような信号をサンプリングすると考える．この信号を離散化した場合，時系列データにおいて０．２Ｈｚ成分のうねりは８８８．８８８…Ｈｚサンプリング時には約４４４４データに１回のうねりとなるのに対して，４４４．４４４…Ｈｚサンプリング時では約２２２２データに１回，２２２．２２２…Ｈｚサンプリング時では約１１１１データに１回となる．従って，単位データ量に占める低周波成分の情報量は，サンプリング周波数が低いときほど多くなるといえる．サンプリングという行為によって離散化された時系列データにはサンプリング周波数の情報は含まれていない．従って，低周波成分の情報が単位データ量に占める割合が相対的に多くなる低いサンプリング周波数で得た時系列データをＡＲＭＡモデル同定する場合には，効果的に不要な低周波成分を除去することが

精度の良い未臨界度推定を行うために重要であると考えられる.

4．3　時系列データのサンプリング周波数と推定時間

近畿大学原子炉で取得した時系列データの高周波側の折れ点周波数は，未臨界度２．１９＄で２４．１Ｈｚ付近，未臨界度０．２８９＄で９．７Ｈｚ付近である．また，この折れ点周波数以上の周波数特性はほぼ１次遅れ特性を示すことから，近畿大学原子炉の場合，時系列データを取得する際のサンプリング周波数としては，折れ点周波数の１デカード高周波側の２５０～３００Ｈｚでも良いことになる．しかし，ＲＰＥ法によるオンラインＡＲＭＡモデル同定を行った場合，同定を開始した後，推定値が真値へ漸近するまでには，ある一定量以上のデータ数が必要であり，そのデータ数を実時間に換算すると１００秒を超えることもある．これでは本来のオンライン推定の意味が失われることになる．そこで，単位時間当たりのデータ数を増加させる手段としてサンプリング周波数を高くするということが考えられる.

例えば，１秒間のアナログ信号をサンプリングしてディジタル時系列データとした場合，サンプリング周波数８８８．８８８…Ｈｚでは約８８９データ，４４４．４４４…Ｈｚでは約４４４データ，２２２．２２２…Ｈｚでは約２２２データとなる．すなわち情報源は同一でもサンプリング周波数を高くすることで見かけ上のデータ量は増加したことになる.

本節では，サンプリング周波数を高く(高周波サンプリングと呼ぶ)することにより単位時間当たりの時系列データ量を増加させ，同定アルゴリズムにパラメータ推定の機会をより多く与え，推定値の真値への漸近を時間的に加速させることが可能かどうかを検討する.

4.3.1　解析の概要

まず，Table 4.2に示した各基本時系列データをTable 4.6で示した適正カットオフ周波数を有する低周波成分除去１次バタワース特性ディジタルフィルタにより処理し，次にこの時系列データを，Table 4.8に示すアルゴリズムを用いて，ＡＲＭＡ

（２，２）同定をし，未臨界度推定を行う．

Table 4.8 Identification algorithm for analysis.

Time series	Algo-rithm	F. factor λ_0	F. factor μ_λ	C. factor μ_K	Projection Algorithm
ID003.888	RPE	1.0	1.0	0.9999	ON
ID005.888	ELS	1.0	---	1.0	OFF
ID010.888	ELS	1.0	---	1.0	OFF
ID003.444	ELS	1.0	---	1.0	OFF
ID005.444	ELS	1.0	---	1.0	OFF
ID010.444	ELS	1.0	---	1.0	OFF
ID003.222	ELS	1.0	---	1.0	OFF
ID005.222	ELS	1.0	---	1.0	OFF
ID010.222	ELS	1.0	---	1.0	OFF

Remarks : (1) "F. factor" means "Forgetting factor".
(2) "C. factor" means "Contraction factor".
(3) Refer to Chapter 2 for λ_0, μ_λ, and μ_K.
(4) "---" means invalidity of the application.

なお，アルゴリズムの選択にあたっては，定常時系列データの推定であることと，経験的な知識を要するファクターを除外するということを考慮してＲＰＥ法の中では最もシンプルな同定手法であるＥＬＳ法を優先的に用いることにした．ただし，推定値の偏りが大きく生じる場合には，残差の白色化を行うＲＰＥ法を採用した．また，その場合にはProjection Algorithmを用いている．

推定に用いたデータ数はいずれも実時間で約２７０秒であり，それぞれのサンプリング周波数の時系列データにおける同定に使用したデータ数は，

Sampling frequency 888.888…Hz → 240000 Data

Sampling frequency 444.444…Hz → 120000 Data

Sampling frequency 222.222…Hz → 60000 Data

である．

未臨界度推定の結果は各サンプリング周波数ごとに８８８．８８８…Ｈｚのシリーズを Fig.4.7.aに，４４４．４４４…Ｈｚのシリーズを Fig.4.7.bに，２２２．２２２…Ｈｚのシリーズを Fig.4.7.cにそれぞれ示す．

4.3.2 時系列データのサンプリング周波数と推定時間に関する定性的知見

4.3.1項における計算結果を基に，推定値が同定開始後，リファレンス未臨界度に漸近し，リファレンス未臨界度±１０¢領域に入った後，収束に向かい始めるおおよその時間をTable 4.9にまとめる．

Table 4.9 Time required for estimation of subcriticality .

Time series	Time(sec)	Time series	Time(sec)	Time series	Time(sec)
ID003.888	7.2	ID003.444	16.2	ID003.222	105.3
ID005.888	11.7	ID005.444	11.7	ID005.222	99.9
ID010.888	4.5	ID010.444	4.5	ID010.222	5.4

Fig.4.7.a，Fig.4.7.b，Fig.4.7.c及びTable 4.9から，定性的に次の知見が得られる．

① サンプリング周波数を高くすることは，推定に要する時間を短縮する上で効果がある方法であると言える．しかし，単位時間当たりのデータ量の増加は必ずしも線形的に推定時間の短縮にはつながらない．このことについては高周波サンプリング上の問題点と共に，次項で詳しく考察する．

② 近畿大学原子炉を模擬して，未臨界度が０．５＄，１＄，２＄の場合で，各々，サンプリング周波数２００Ｈｚサンプリング及び５００Ｈｚのシミュレーション時系列データを作成し，ＡＲＭＡモデル同定による未臨界度推定を行った場合，サンプリング周波数による推定性能には違いはないことを確認している．しかし，本研究では特に未臨界度の深い時には，時系列データのサンプリング周波数の違いが推定時間に大きな影響を与えることが観察される．

4.3.3 高周波サンプリングの問題点と限界

4.3.2項に示したように，高いサンプリング周波数でサンプリングする事は推定時間を短縮する上で有効な手段である．しかしその一方で，サンプリング周波数を高くすることにより，単位時間当たりに処理すべきデータ量が線形的に増加するのに対して，必ずしもそのデータ量の増加が線形的に推定時間の短縮となって顕在化してこない．この項では高周波サンプリングの問題点，及びその限界について考察する．

4.3.2項で示した結果より，高周波サンプリングの推定時間に与える効果は未臨界度が深くなるにつれて顕著であり，未臨界度が浅い側ではその効果は小さい．この原因の一つとして未臨界原子炉の高周波側の折れ点周波数が未臨界度が深くなるにつれて高周波側へシフトすることが考えられる．しかし，上述したシミュレーションの結果を考慮すると，それのみが原因とは考えられない．そこで，もう一つのファクターとして，時系列データ中に含まれる外部雑音の相対的な大きさが考えられる．リファレンス未臨界度が２．１９$，１．２５$，０．２８９$での，８８８．８８８…Ｈｚサンプリング時系列データをＦＦＴ法(1024 Point, 600 Ave-raging)により計算したＰＳＤをFig.4.8に示す．この図から，未臨界度が浅いときには未臨界度が深いときに比べて，未臨界原子炉システムの高周波側の折れ点周波数近傍のＰＳＤが外部雑音源のレベルより高いことが観察される．

これらのことから，未臨界原子炉のシステムの高周波側の折れ点近傍のＰＳＤが相対的に高い場合，すなわち外部雑音源のレベルが相対的に低い場合の時系列データに関しては，低いサンプリング周波数においても比較的短時間に未臨界度推定が可能であるといえる．

定常状態の推定において，サンプリング周波数は高ければ必ずしも良いというわけではなく，観測対象の状況により，その効果に差異がでると考えられる．また一方で，このことは未臨界度が変化する非定常なシステムにおいて一定の推定精度と追従速度を維持するためには，高周波サンプリングの必要性を示唆している．

次に，高周波サンプリングにおける時系列データの特性と信号処理上での問題点について述べる．

① 第３章第４節でも述べたように時系列データがローパスフィルタなどの外

部フィルタの特性を含む場合，正確な未臨界度の推定を行うことが難しくなる．従って，同定対象とは無関係な高周波成分を除去してからモデル同定しなくてはならない．サンプリング周波数を高くすることは，ただ単にサンプリングポイントを増やすだけではなくナイキスト周波数以下の高周波成分は全て信号として取り込むことになる．Fig.4.9に，サンプリング周波数の違いによる未臨界度推定時間の差異がほとんど見られなかったＩＤ０１０シリーズ(リファレンス未臨界度０．２８９＄)における各サンプリング周波数での時系列データの一部を示す．この時系列データにおいて，Table 4.9 の推定に要する時間から未臨界度推定に必要な情報としては２２２．２２２…Ｈｚサンプリングの情報で十分である．従ってこのとき８８８．８８８…Ｈｚサンプリングでは結果的にサンプリングポイントの増加量は高周波成分を拾うためにのみに費やされ，推定時間の短縮には寄与しないことになる．高周波サンプリングは単位時間当たりのデータ量を増加させて，ＲＰＥ法による推定アルゴリズムに対し，パラメータ推定の機会を多く与えると共に，不必要な高周波成分の情報も与えることになる．

② 信号計測系のハードウェアの問題としては，高速な信号の変化に対応できる立ち上がり時間を持ったものが必要となる．検出器からの信号を増幅する Current Amp.の立ち上がり時間が充分小さくないと，信号は高周波成分除去型のフィルタを通過したものとなり，Current Amp. 特性がシステムの同定に反映されて，より複雑なモデルの同定が要求されることになる．

③ 計算処理系のハードウェア上の問題として，サンプリング周波数の上昇に比例して単位時間当たりの処理データ量も増加する問題がある．オンラインで未臨界度推定するためには，この様な計算機負荷に十分応えることのできる計算機能力性能が必要となる．

高周波サンプリングによって時系列データを取得する場合には，観測対象及び測定環境に注意を払うと共に，以上のような問題点を考慮する必要がある．

5 結論

シミュレーションデータ解析に用いたアルゴリズムをそのまま近畿大学原子炉から取得された時系列データに適用した場合，未臨界度推定値の精度がシミュレーション時系列データに対する結果に比べて良くない．これは，測定系に未知のの外乱が加わる上，検出器からの信号は，増幅器，フィルター，A／Dコンバータ等を通過する度に各種の処理を施されるため，信号の質(quality)がシミュレーションデータとは本質的に異なっているため考えられる．特にパラメトリックなスペクトル解析では，限られた数のパラメータでモデルを表現するため，雑音によるスペクトルの歪がパラメータの推定に大きく影響を及ぼす．したがって，観測信号から未臨界度の情報を含む周波数領域のみを最適に抽出する信号処理技術の開発が必要となる。

そこで，本年度は時系列データ中に含まれる低周波成分の未臨界度推定に及ぼす影響を検討するため，

① 時系列データ中に含まれる低周波成分の適切な除去法．

② 時系列データのサンプリング周波数が推定値に及ぼす効果．

の２点に問題を絞り，近畿大学原子炉において取得したＣＩＣ検出器からの出力信号の揺らぎをＲＰＥ法を主に用いてＡＲＭＡモデル同定し，以下の結論を得た．

1．低周波成分除去用ハイパスフィルタの選択と未臨界度推定

本研究が目的としている臨界安全監視モニターでは，そのバイアス成分自身が刻々と変化する場合にも適用できるアルゴリズムでなくてはならない．そこで，ここでは検出器からの信号をハイパスフィルタに通すことによって低周波成分を除去する方法を採用した．この方法は単にバイアス成分（ＤＣ成分）除去するだけでなく，不要な低周波成分を減衰させるという目的にも効果がある．しかし，パラメータ同定される信号をハイパスフィルタに通すということは，本来同定すべき未臨界原子炉のシステムにハイパスフィルタの特性が加わった複合形システムのパラメータ推定の問題を生じる．

そこで，基本時系列データに対し１次のバタワース特性を有するディジタルフィルタを用い，ハイパスフィルタのカットオフ周波数を変化させて低周波成分を除去

し，それぞれの時系列データに対しＥＬＳ法によるＡＲＭＡ(２，２)モデル同定を行い，未臨界度の推定を行った結果，ハイパスフィルタのカットオフ周波数のわずかな違いが未臨界度の推定値の精度に及ぼす影響は極めて大きいものであることが確認され，定性的には次のようま知見を得た．

① ハイパスフィルタのカットオフ周波数が低すぎて十分なバイアス除去ができない場合は，実際よりも浅く未臨界度を推定する．
② ハイパスフィルタのカットオフ周波数が適正値より高い場合，バイアスの低減効果は大きいが推定値は実際よりも深い方へ偏る．これはハイパスフィルタ自身の影響が強くなり，本来同定すべき未臨界原子炉のシステムがハイパスフィルタとの複合形のシステムとなっていることが原因であると考えられる．
③ サンプリング周波数の観点から推定状況を考察した場合，計算に用いるデータ数が同じであれば，当然サンプリング周波数が低い程より低周波の情報を抽出できることになり，未臨界度推定精度低周波除去用のハイパスフィルタの特性に対して敏感になる．

２．ハイパスフィルタの適正カットオフ周波数

基本時系列データに対して適正と思われる各種のカットオフ周波数を持つハイパスフィルタを適用しそのデータを解析から，低周波成分除去用のハイパスフィルタはに付いては次の２点を考慮する事が必要であることが分かった．

① システム同定の立場からは，ハイパスフィルタ自身の存在は複合形のシステム同定を要求する可能性があるという観点から有害であるから，サンプリング周波数に比べてカットオフ周波数はなるべく低周波側で，かつフィルタ次数も低次であることが好ましい．
② 逐次推定法を用いたＡＲＭＡモデル同定による未臨界度推定を行うには，時系列データのバイアスを一定以上除去する必要がある．

したがって，低周波成分除去用のハイパスフィルタの最適なカットオフ周波数は，「適正なＤＣ成分低減率を得られる保証の下で，可能な限り低い周波数」という条

件から決定されるべきであるが，ハイパスフィルタのカットオフ周波数f_cの関数であるＤＣ成分低減率の微係数が最も大きく変化する周波数をカットオフ周波数として選択すべきである．

3．不要低周波成分除去という側面からのハイパスフィルタの必要性

バイアスの除去という意味でハイパスフィルタは必要であるが，また，時系列データ中に含まれるＤＣ成分以外の不要な低周波成分を除去するという側面からもハイパスフィルタは必要である．本研究では基本時系列データを，平均値除去法と適正カットオフ周波数を有するハイパスフィルタを用いて処理した実データをＥＬＳ法を用いてＡＲＭＡ(２，２)モデル同定して未臨界度推定を行い，ハイパスフィルタの導入が不要な低周波成分を除去し未臨界度推定の性能向上に貢献することが示され次のような知見を得た．

① 平均値除去法によって単にＤＣ成分のみを除去した場合に比べて，適正カットオフ周波数を有するハイパスフィルタで処理したものの方が未臨界度を精度良く推定できる．このことは特に，サンプリング周波数が低い場合において顕著である．

② 未臨界度が浅い場合すなわち炉出力自体が高い場合，両者には大きな違いは認められない．

このことは逆に，出力レベルが低くかつサンプリング周波数が低い場合に，ＤＣ成分以外の低周波成分が，未臨界度推定に与える影響が大きくなることを示しているとも言える．従って，低周波成分の情報が単位データ量に占める割合が相対的に多くなる低いサンプリング周波数で得た時系列データをＡＲＭＡモデル同定する場合には，効果的に不要な低周波成分を除去することが精度の良い未臨界度推定を行うために重要である．

4．時系列データのサンプリング周波数と推定時間

サンプリング周波数を高く（高周波サンプリングと呼ぶ）することにより単位時

間当たりの時系列データ量を増加させ，同定アルゴリズムに単位時間当たりのパラメータ推定の機会をより多く与え，推定値の真値への漸近を時間的に加速させることが可能かどうかを検討し，以下の結論を得た．

基本時系列データを適正カットオフ周波数を有する低周波成分除去用１次バタワース特性ディジタルフィルタにより処理し，次にこの時系列データをＡＲＭＡ（２，２）モデル同定した場合

① 高周波サンプリングの未臨界度推定時間に与える効果は未臨界度が深くなるにつれて顕著であり，未臨界度が浅い側ではその効果は小さい．サンプリング周波数を高くすることは，推定に要する時間を短縮する上で効果があると言える．しかし，単位時間当たりのデータ量の増加は必ずしも線形的に推定時間の短縮にはつながらない．

② 近畿大学原子炉を模擬したシミュレーション時系列データでは，サンプリング周波数による推定性能の違いは生じないが，本研究で用いたデータの場合，未臨界度の深い時には，時系列データのサンプリング周波数の違いが推定時間に大きな影響を与えることが観察された．

これらは，未臨界原子炉のシステムの高周波側の折れ点周波数近傍のＰＳＤが相対的に高い場合，すなわち外部雑音源のレベルが相対的に低い場合の時系列データに関しては，低いサンプリング周波数においても比較的短時間に未臨界度推定が可能であるとことを示している．したがって，定常状態の未臨界度推定においては，サンプリング周波数は高ければ必ずしも良いというわけではなく，観測対象の状況により，その効果に差異がでると考えられる．このことは未臨界度が変化する非定常なシステムにおいては，一定の推定精度と追従速度を維持するためには高周波サンプリングの必要性を示唆している．しかし，高周波サンプリングには次のような問題点がある．

① 高周波サンプリングは単位時間当たりのデータ量を増加させて，ＲＰＥ法による推定アルゴリズムに対し，パラメータ推定の機会を多く与えると共に，不必要な高周波成分の情報も与えることになる．

② 検出器からの信号を増幅する Current Amp.には高速な信号の変化に対応できる立ち上がり特性を持ったものが必要となる．この立ち上がり時間が充分小さくないと信号は高周波成分除去型のフィルタ特性を持つことになり，

Current Amp. 特性がシステムの同定に反映され，より複雑なモデルの同定が要求されることになる．

③ 計算処理系のハードウェア上の問題として，サンプリング周波数の上昇に比例して単位時間当たりの処理データ量も増加する問題がある．オンラインで未臨界度推定するためには，この様な計算機負荷に十分応えることのできる計算機能力が必要となる．

6. エリアシング防止用ローパスフィルタの選択について

ＡＲＭＡモデルパラメータ逐次推定の手法を用いて未臨界度推定を行うにあたっては，スペクトル解析用時系列データ取得の際の常識であるエリアシング防止用ローパスフィルタの使用に関しては，充分な配慮が必要である．取得した時系列データがローパスフィルタの折れ点周波数近傍の周波数特性を含む場合は，それに見合った高次のＡＲＭＡモデ同定を行わなければならなくなり，高速な未臨界度推定が行い難くなるばかりでなく，未臨界度の推定値に大きな影響を及ぼす．時系列データをできるだけ低次数のＡＲＭＡモデルで表現し，高速かつ高精度の未臨界度推定を行うためには，意識的にエリアシングを生じるようなサンプリング周波数を選ぶことにより，エリアシング防止用フィルタの特性がスペクトルに現れないようにすることが重要になる．したがって，エリアシング防止用フィルタの特性に対応したサンプリング周波数の選択が重要である．

7. 今後の課題

本研究で最後に残された最も重要な課題が，非定常な時系列データに対する本手法の適用である．既に，ある程度の研究を進めているが，変動するパラメータをいかに素早く追跡できるかはアルゴリズムの中の色々な因子が関連しており，新しい手法の研究と併せて継続的な研究を行う必要がある．

参考文献

[1] 通商産業省・資源エネルギー庁，「関西電力(株)美浜発電所２号機蒸気発生機伝熱管損傷事故について」，PP.58-pp.61

[2] K.Kishida, Physical Langevin model and the time series model in sys-tems far from equilibrium., Physical Review A, vol 25, Num 1, PP.496-507, JAN(1982)

[3] Takeo Umeda et al., Experience of ON-LINE Surveilance at ONAGAWA-1 BWR Plant., Proceeding of SMORN-V, Progress in Neuclear Energy, vol.21, pp.35-41, 1988

[4] 森田昌裕・山田澄・住田健二，「適応フィルターによる臨界安全監視システムの基礎研究」，(社)日本原子力学会「昭和６３年秋の大会」予稿集，A29, PP.29

[5] 森田昌裕，「適応フィルターによる臨界安全監視手法の開発」，大阪大学修士論文，1991

[6] Lennart Ljung and Torsten Soderstrom, Theory and Practice of Recursive Identification, Massachusetts Institute of Technology, 1983

[7] 谷萩隆嗣，「ディジタル信号処理の理論３　推定・適応信号処理」，PP.74, コロナ社(1986)

[8] V.Solo, Time series recursions and stochastic approximation., PhD dissertation, The Australian National University, Canberra, Australia, 1978

[9] 木村英紀，「ディジタル信号処理と制御」，昭晃堂，1982

[10] 山田澄，「適応フィルタによる臨界安全監視システム開発の基礎研究」，近畿大学原子炉等利用共同研究経過報告書平成２年度，大阪大学工学部

[11] J.E. Hoogen et al., Summary of the Benchmark Test on Artificial Noise Data, Proceeding of SMORN-V, Progress in Nuclear Energy, vol.21, pp.815-824, 1988

[12] Zhong-Qi Qiu et al., An adaptive LMS Algorithm with a Quick Tracking Property for Time VArying Parameter Estimation., 電子情報通信学会論文誌 A, vol. J72-A, No.7, PP.1038-1044, JUL(1989)

Appendix A
MAパラメータによる式誤差の白色化

Fig.2.1に示す線形システムモデルにおいてその出力は，

$$y(t) = \frac{B(z^{-1})}{A(z^{-1})}u_t + v(t) \tag{A.1}$$

で表され，一方，推定システムモデルによる式誤差$v(t)$は，

$$v(t) = A(z^{-1})y(t) - B(z^{-1})u(t) \tag{A.2}$$

で表される．一般に式誤差は白色でないため，推定値に偏りを生じる．そこで式誤差を線形システムとして近似しＭＡモデルで表現すると，

$$v(t) = C(z^{-1})e(t) = e(t) + \sum_{i=1}^{n} c_i e(t-i) \tag{A.3}$$

表わされる．ただし，$e(t)$は白色時系列である．

式(A.2)，(A.3)よりＡＲＭＡＸモデル

$$A(z^{-1})y(t) = B(z^{-1})u(t) + C(z^{-1})e(t) \tag{A.4}$$

を得る．式(A.4)より，

$$A(z^{-1})\frac{1}{C(z^{-1})}y(t) = B(z^{-1})\frac{1}{C(z^{-1})}u(t) + e(t) \tag{A.5}$$

とし，ここで，

$$\tilde{y}(t) = \frac{1}{C(z^{-1})}y(t) \tag{A.6}$$

$$\tilde{u}(t) = \frac{1}{C(z^{-1})}u(t) \tag{A.7}$$

と置き換えると，式(A.5)は，

$$A(z^{-1})\tilde{y}(t) = B(z^{-1})\tilde{u}(t) + e(t) \tag{A.8}$$

となり，式誤差が白色化されることから不偏一致推定量が得られる．このときに，$\tilde{y}(t)$，$\tilde{u}(t)$，$\tilde{v}(t)$は，

$$\tilde{y}(t) = y(t) - \sum_{i=1}^{n} c_i \tilde{y}(t-i) \tag{A.9}$$

$$\tilde{u}(t) = u(t) - \sum_{i=1}^{n} c_i \tilde{u}(t-i) \tag{A.10}$$

$$\tilde{v}(t) = v(t) - \sum_{i=1}^{n} c_i \tilde{v}(t-i) \qquad (A.11)$$

によって生成される.

Appendix B
縮約係数（Contraction Factor）の作用

時刻 t におけるＭＡ部の推定値を用いて，その伝達関数を

$$\hat{C}_t(z^{-1}) = 1 + c_{t1}z^{-1} + c_{t2}z^{-2} + \cdots + c_{tn}z^{-n} \qquad (B.1)$$

と表したとき，式(B.1)で表されたシステムが安定であるためには，

$$\hat{C}_t(z^{-1}) = 0 \qquad (B.2)$$

としたとき，

$$1 + c_{t1}z^{-1} + c_{t2}z^{-2} + \cdots + c_{tn}z^{-n} = 0$$

$$z^n + c_{t1}z^{n-1} + c_{t2}z^{-2} + \cdots + c_{tn} = 0$$

$$(z-p_1)(z-p_2)\cdots(z-p_n) = 0 \qquad (B.3)$$

から求められる根，p_1，p_2，…，p_n が全て，z 領域の単位円内にある事が必要十分条件である.

ここで，z^{-1} に縮約係数K_tを作用させると，

$$\hat{C}_t(K_t z^{-1}) = 0 \qquad (B.4)$$

の根は，

$$(z-K_t p_1)(z-K_t p_2)\cdots(z-K_t p_n) = 0 \qquad (B.5)$$

より，$K_t p_1$，$K_t p_2$，…，$K_t p_n$ が根として得られ，縮約係数は，

$$0 < K_t < 1 \qquad (B.6)$$

で，与えられているので，根の存在領域が縮約される事になる，従って，式(B.2)が安定な場合，縮約されたモデルも安定である．また，式(B.5)を展開する事によって元の方程式に焼き直すと，

$$1+c_{t1}K_t z^{-1}+c_{t2}K_t{}^2 z^{-2}+\cdots+c_{tn}K_t{}^n z^{-n} = 0 \qquad (B.7)$$

となり，縮約されたモデルは，

$$\hat{C}_t(K_t z^{-1}) = 1+c_{t1}K_t z^{-1}+c_{t2}K_t{}^2 z^{-2}+\cdots+c_{tn}K_t{}^n z^{-n} \qquad (B.8)$$

で表され，式(B.4)の左辺に帰結する．

Appendix C
未臨界原子炉のＡＲＭＡモデルパラメータの導出

定常状態にある未臨界原子炉の物理現象が，遅発中性子１群・一点炉近似動特性方程式に従う場合、中性子密度の平均値からの揺らぎδN，遅発中性子先行核密度の平均値からの揺らぎδCは次のMarkov-Langevin方程式を満足する．

$$\frac{d\delta N}{dt} = \frac{k(1-\beta)-1}{l}\delta N+\lambda\delta C+r_1(t) \qquad (C.1.a)$$

$$\frac{d\delta C}{dt} = \frac{k\gamma\beta}{l}\delta N-\lambda\delta C+r_2(t) \qquad (C.1.b)$$

ここで，

$$r(t) = [r_1(t)\, r_2(t)]^T \qquad (C.2)$$

は，平均値０の白色不規則力で，その分散は，

$$\langle r(t)\, r(\tau)^T\rangle = D\delta(t-\tau) \qquad (C.3)$$

で与えられる．ここで，$\langle\cdot\rangle$は期待値を表す．また，Dは拡散定数行列であり，拡張Shcottky公式

$$D = Q+FE+EF^T \qquad (C.4)$$

によって与えられる．ただし，

$$F = \begin{bmatrix} \dfrac{k(1-\beta)-1}{l} & \lambda \\ \dfrac{k\gamma\beta}{l} & -\lambda \end{bmatrix} \qquad (C.5)$$

$$E = diag\left[\frac{Sl}{1-k} \quad \frac{Sl\beta}{(1-k)\lambda}\right] \tag{C.6}$$

$$Q = \frac{k(1-\beta)}{l}\begin{bmatrix}\langle\nu_0(\nu_0-1)\rangle & \langle\nu_0\nu_1\rangle \\ \langle\nu_0\nu_1\rangle & \langle\nu_1(\nu_1-1)\rangle\end{bmatrix} \tag{C.7}$$

であり，ここで用いる，k，β，λ，l，γ は，慣用のものである．ν は，^{235}Uの核分裂により生じる中性子数であり，$\nu_0 = \nu(1-\beta)$，$\nu_1 = \nu\beta$ であり，S，は外部中性子源の強さを表す．

ここで，$x(t) = [\delta N \delta C]^T$ とし，観測雑音を$s(t)$ とすると，式(C.1)は，

$$\frac{d}{dt}s(t) = Fx(t)+r(t) \tag{C.8.a}$$

$$y(t) = Hx(t)+s(t) \tag{C.8.b}$$

と書ける．なお，$H = [1 \quad 0]$である．

式(C.8)をサンプリング時間Δt で離散時間化すると，

$$x(n+1) = \Phi x(n)+f(n+1) \tag{C.9.a}$$

$$y(n) = Hx(n)+v(n) \tag{C.9.b}$$

と書ける．ここで，

$$x(n) = x(n\Delta t) \tag{C.10}$$

$$y(n) = y(n\Delta t) \tag{C.11}$$

$$\Phi = exp(F\Delta t) \tag{C.12}$$

$$f(n) = \int_{(n-1)\Delta t}^{n\Delta t} exp[F\cdot(n\Delta t-\tau)]r(t)d\tau \tag{C.13}$$

$$v(n) = \int_{(n-1)\Delta t}^{n\Delta t} exp[F\cdot(n\Delta t-\tau)]s(\tau)d\tau \tag{C.14}$$

$$E[f(m)\cdot f(n)^T] = V\delta_{mn} \tag{C.15}$$

$$E[v(m)\cdot v(n)^T] = R\delta_{mn} \tag{C.16}$$

$$V = \int_0^{\Delta t} exp[F\tau]\cdot D\cdot exp[F^T\tau]d\tau \tag{C.17}$$

$$R = \int_0^{\Delta t} exp[F\tau]\cdot d\cdot exp[F^T\tau]dr \tag{C.18}$$

である．δ_{mn} は，クロネッカのデルタである．

次に，Φの固有値を，η_+，η_-とすると，Sylvesterの固有値展開公式を用いて，

$$\Phi = exp(F\Delta t) = \begin{bmatrix}\Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22}\end{bmatrix}$$

$$= \frac{exp(\eta_+\Delta t)}{\eta_+-\eta_-}\begin{bmatrix}-\alpha-\eta_- & \lambda\\ k\beta/l & -\lambda-\eta_-\end{bmatrix} - \frac{exp(\eta_-\Delta t)}{\eta_+-\eta_-}\begin{bmatrix}-\alpha-\eta_+ & \lambda\\ k\beta/l & -\lambda-\eta_+\end{bmatrix} \tag{C.19}$$

と書ける．ただし，$\alpha = [1-k(1-\beta)]/l$ である．同様にして，Vは，

$$\begin{aligned} V = {} & \frac{exp(2\eta_+\Delta t)-1}{2\eta_+(\eta_+-\eta_-)^2}(F-\eta_-I)D(F^T-\eta_-I)\\ & -\frac{exp(\eta_++\eta_-)\Delta t-1}{(\eta_++\eta_-)(\eta_+-\eta_-)^2}(F-\eta_-I)D(F^T-\eta_+I)\\ & -\frac{exp(\eta_++\eta_-)\Delta t-1}{(\eta_++\eta_-)(\eta_+-\eta_-)^2}(F-\eta_+I)D(F^T-\eta_-I)\\ & +\frac{exp(2\eta_-\Delta t)-1}{2\eta_-(\eta_+-\eta_-)^2}(F-\eta_+I)D(F^T-\eta_+I) \end{aligned} \tag{C.20}$$

で与えられる．

式(C.9.a)，(C.9.b)に適当な物理定数と白色時系列$f(n) = [f_1(n) \quad f_2(n)]$，$v(n)$を用意する事により，未臨界原子炉のシミュレーションデータを作成する事ができる．

この様にして，物理モデルはMarkov-Langevin方程式(C.9.a)，(C.9.b)で記述できたが，直接観測できるのは$\{y(n)\}$だけであって，$f(n)$，$v(n)$は，直接観測できないため，このままではモデルパラメータΦを求める事はできない．そこで，$\{y(k); k=\ldots n-3, n-2, n-1, n\}$が与えられたときの$x(n)$の最適推定値を，$x(n \mid n)$から確率システムの物理モデルを再構築する必要がある．この操作は，過去から現在まで蓄積された時系列データ$\{y(n)\}$の張るデータ空間へ現在のデータ$y(n)$を影像する事で，

$x(n \mid m)$　：m時点迄の観測データから得られるn時点での状態変数ベクトル$x(n)$の最適推定値

$y(n \mid n-1)$　：n－1時点までの観測データから得られるn時点での観測値$y(n)$の最適推定値

とし，イノベーション$\nu(n)$を，

$$\nu(n) = y(n)-y(n \mid n-1) \tag{C.21}$$

と定義すると，物理モデルは等価的に，

$$x(n+1 \mid n+1) = \Phi x(n \mid n)+K\nu(n+1) \tag{C.22.a}$$

$$y(n) = Hx(n \mid n)+(I-HK)\nu(n) \tag{C.22.b}$$

と表現できる．これがイノベーションモデルである．Kはカルマンゲインと呼ばれる定数ベクトル$[k_1 \quad k_2]^T$で，２×２行列Ｐに関するリカッチ(Riccatti)型方程式

$$P = \Phi\{P-KHP\}\Phi^T+V \tag{C.23}$$

の一意解

$$P = \begin{bmatrix} P_{11} & P_{12} \\ P_{12} & P_{22} \end{bmatrix} \tag{C.24}$$

によって，

$$K = PH^T\Gamma^{-1} \tag{C.25}$$

で与えられる．Γは，イノベーションの分散で，

$$\Gamma = HP^TH+R \tag{C.26}$$

で与えられる．

最後に，式(C.22.a)，(C.22.b)より$x(n+1 \mid n+1)$を消去し，$\nu(n)$から$y(n)$への伝達関数を求めると，遅発中性子１群・一点炉近似動特性方程式で記述できる原子炉のＡＲＭＡ(２，２)モデル，

$$G(z^{-1}) = \frac{1+b_1z^{-1}+b_2z^{-2}}{1+a_1z^{-1}+a_2z^{-2}} \tag{C.27}$$

を得る．このとき，ＡＲＭＡモデルパラメータは，

$$a_1 = -(\Phi_{11}+\Phi_{22}) \tag{C.28.a}$$

$$a_2 = \Phi_{11}\Phi_{22}-\Phi_{12}\Phi_{21} \tag{C.28.b}$$

$$b_1 = \{P_{12}\Phi_{12}-P_{11}\Phi_{22}-R(\Phi_{11}+\Phi_{22})\}/(P_{11}+R) \tag{C.28.c}$$

$$b_2 = (\Phi_{11}\Phi_{22}-\Phi_{12}\Phi_{21})R/(P_{11}+R) \tag{C.28.d}$$

で与えられ，これらは全て一点炉近似動特性方程式の物理定数とサンプリング周波数などのみから算出される．

従って，原子炉から観測される時系列をＡＲＭＡ(２，２)過程，

$$y(n) = -a_1y(n-1)-a_2y(n-2)+e(n)+b_1e(n-1)+b_2e(n-2) \tag{C.29}$$

として，モデル化すればこのモデルのパラメータから原子炉の物理定数を逆算できる事になる．

Appendix D

ＡＲＭＡＸ１モデル
オンラインパラメータ推定プログラム

ＡＲＭＡＸ１．Ｈ

```
#include <stdio.h>
#include <time.h>
#define IGNORE 0
#define ABORT 2
#define ON  1
#define OFF 0
#define SUCCESS 1
#define NOTHING 0
#define FAULT -1
#define NOCONVERGE 2
#define RPEM 1
#define PLR 2
#define ELS 3
#define RML 4
#define PMA 1
#define PMDA 2
#define PARAMETERFILE 1
#define BINARYFILE  2
#define INITFILE    3
#define PICK 0
#define DUMP 1
#define CONTINUE  1
#define FREE    2
#define MAXBOTTOM 23
#define DOSSAVE   1
#define MAINMENU  2
#define PARAMCHANGE 3
#define FILECHANGE 4
#define RPEMCALL1 5
#define RPEMCALL2 6
#define EMPTY    7
#define DOSLOAD   9
#define NN 0
#define NE 1
#define EN 2
#define EE 3
#define WITH_EXTENTION   0
#define WITHOUT_EXTENTION 1

typedef struct {
  char  name[256];      /* データが格納されているファイル名*/
  int   byte;          /* データ長*/
  long  number;        /* データ数*/
} BINARY_FILE;         /* 時系列Ｕ，Ｙの情報に関する構造体*/

typedef struct {
  char  name[256];
  int   order;
} INIT_FILE;

typedef struct {
  double lambda_mu;      /* 忘却係数定数部μ*/
  double lambda_init;    /* 忘却係数の初期値λ(0)*/
  double contraction_mu; /* 縮約係数定数部μ*/
  int   activation;      /* 活性化の選択スイッチ*/
  long  activation_interval; /* 活性化の間隔*/
  double activation_power;  /* 活性化量*/
} RPEM_SUB;              /* ＲＰＥＭ法の補助パラメータ*/

typedef struct {
  double theta[25];      /* ＡＲＭＡＸパラメータ*/
  double UDL[25][25];    /* ＵＤＬマトリックス*/
  double P[25][25];      /* 予測誤差分散マトリックス*/
  double L[25];          /* ゲインベクトル*/
  double phi[25];        /* WORK*/
  double psi[25];        /* WORK*/
  int   n;               /* 同定次数*/
  double u;              /* ｕの時系列（最新値）*/
  double y;              /* ｙの時系列（最新値）*/
  double lambda;         /* 忘却係数*/
  double K;              /* 縮約係数*/
  double c;              /* 発散防止パラメータ*/
  int   stable;          /* 安定性チェックスイッチ*/
  double prediction_error;  /* The prediction error */
  double residual;       /* The residual */
  double pmatrix0;       /* Ｐマトリックスの初期値*/
} RPEM_PARAMETER;        /* ＲＰＥＭ主要パラメータ*/

typedef struct {
  double real;           /* 複素数実部*/
  double image;          /* 複素数虚部*/
} COMPLEX;

/* AMX1MAIN.C */
int   Main_Menu(void);
int   Hard_Error(int errval, int ax, int bp, int si);
void  Drive_Message(void);

/* AMX1MENU.C*/
void  Title(void);
void  FunctionKey(int param);
int   Key_Judge(void);
void  Message(char *ptr, int param, int color, int y);

/* AMX1AUTO.C*/
int   AutoMain(char *argv[]);
void  Report_Writing(int count, FILE *fil, char *set_filename,
                     int result, char *report_message);

/* AMX1READ.C*/
int   ReadFileCall(char*filnam);
long  Binary_File_Check(FILE *fin, int byte, char *ptr1);
int   Std_Reading(FILE *fil, char *ptr1, int counter);

/* AMX1DIR.C*/
int   getdir(char *search_dir, char *search_path, char *dptr[],
             int max_dir, int file_kind);
int   GetfileSub(char *finam, char *drive, char *dir, char *name,
                 char *ext, int file_kind);
int   getfile(char *finam, int file_kind);
```

```
int   File_Selection(char *ptr, int file_kind);

/* AMX1PARA.C*/
int   ParamChangeCall(void);
int   File_Reset(int file_kind, char *ptr1);
void  Crt_Reset(void);
void  Useable_Renumber(void);

/* AMX11FILE.C*/
int   FileChangeCall(void);

/* AMX1INPU.C */
double Manual_Set_D(double pool, double min, double max,
                    int type, unsigned char length);
int   Manual_Set_I(int pool, int min, int max,
                    int type, unsigned char length);
long  Manual_Set_L(long pool, long min, long max,
                    int type, unsigned char length);
int   Manual_Set_F(char *filename, int extention,
                    unsigned char length, int line);

/* AMX1FUNC.C*/
int   Method(RPEM_SUB*subp);
char  *String_Adjust_1(char *ptr1, int max_length);
char  *String_Adjust_2(char *ptr1, int max_length);
int   Filename_Check(char *ptr, int maxlen, int check_type);
void  File_Close(void);
void  Udl_to_P(int order, double udl[][25],  double p[][25]);

/* AMX1CALL.C*/
int   Armax_Call(void);
int   Starting_Check(void);
int   Parameter_Init(void);
void  Time_Infomation(long number, time_t first_time,
                time_t second_time, time_t injury_time);
void  Parameter_Std_Init(void);
double Read_Data(FILE *fin, int byte);

/* AMX1RPEM.C*/
void  Rpem_Routine(void);
int   Stable_Check(double *work, int n);
int   UDL_Factorization(RPEM_PARAMETER  *rpemp);
int   P_Matrix_Calc(RPEM_PARAMETER  *rpemp);

/* AMX1OUTP.C*/
void  Output_Result(long loop);
void  Parameter_Writing(int number, int start_point,  double theta[]);
void  Writing_Info(int order, int u_order, char *method1,  char *method2,
                    int force_break);
void  Setupfile_Writing(FILE *fil);
void  Output_Pole(void);
void  Output_PL(long loop, int activation_method);

/* AMX1POLE.C*/
int   Pole_Calculation(int order, double theta[],  COMPLEX pole[]);
int   Bairstow(double ta[],  int n, COMPLEX x[]);
void  Equation_2(double a[],  COMPLEX x[]);

/* AMX1OPTI.C*/
int   Remake_File(void);
void  Help(void);
```


AMX1MAIN.C

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <dos.h>
#include "armax1.h"

extern unsigned _stklen = 32768;

char  *read_param[] ={"IDENTIFICATION_ORDER::::::::::::::",
            "RECORD_LENGTH::::::::::::::::::::",
            "U_DATA_FILENAME::::::::::::::::::",
            "Y_DATA_FILENAME::::::::::::::::::",
            "INIT_PARAMETER_SETTING_FILENAME::",
            "NUMBER_OF_SKIPPING_DATA::::::::::",
            "NUMBER_OF_IDENTIFICATION::::::::",
            "FORGETTING_FACTOR_MU::::::::::::",
            "FORGETTING_FACTOR_INIT::::::::::",
            "CONTRACTION_FACTOR_MU::::::::::::",
            "PREVENTION_OF_DIVERGNECE::::::::",
            "STABLE_CHECK_SWITCH::::::::::::::",
            "INIT_VALUE_OF_DIAGONAL_P_MATRIX::",
            "ACTIVATION_METHOD_SWITCH::::::::",
            "ACTIVATION_INTERVAL::::::::::::::",
            "ACTIVATION_POWER::::::::::::::::",
            "OUTPUT_FILENAME::::::::::::::::::",
            "NUMBER_OF_OUTPUT_INTERVAL::::::::",
            "BINARY_POLE_OUTPUT_FILENAME::::::",
            "P/L_OUTPUT_FILENAME::::::::::::::"};
char  set_filename[128], out_filename[128], pole_filename[128],
    pl_filename[128], tmp_buff1[256], tmp_buff2[256], error_drive;

int   AutoSystem = OFF, AutoStop = OFF,
    DriveError = OFF, DriveErrorNumber;

long  identification_number, skip_number, interval_number,
    abs_useable_number, useable_number;

FILE  *fin_u, *fin_y, *fin_init, *fout_result, *fout_pole[10], *fout_pl;

BINARY_FILE   u_file, y_file;
INIT_FILE     init_file;
RPEM_SUB      subp;
RPEM_PARAMETER rpemp;

/*-------------------------------------------------------------*/
int main(int argc, char *argv[])
{
  int   check;
  clrscr();
  _setcursortype(_NOCURSOR);
  harderr(Hard_Error);
  FunctionKey(DOSSAVE);

  if (argc == 1)
    Main_menu();
  else {
    AutoSystem = ON;
    check = AutoMain(argv);
  }
  FunctionKey(DOSLOAD);
  _setcursortype(_SOLIDCURSOR);
  clrscr();
```

```
    if (AutoSystem == ON) {
      Title();
      if (check == SUCCESS){
        sprintf(tmp_buff1, "PROGRAMMODE: NORMAL TERMINATION");
        Message(tmp_buff1, FREE, LIGHTCYAN, 4);
      } else {
        sprintf(tmp_buff1, "PROGRAMMODE: ABNORMAL TERMINATION");
        Message(tmp_buff1, FREE, LIGHTRED, 4);
      }
    }
    return SUCCESS;
}

/*-------------------------------------------------------------------*/
int Main_menu(void)
{
    int i, j;

    while (1) {
      Title();
      FunctionKey(MAINMENU);
      i = Key_Judge();
      switch (i) {
        case 1 :
          if ( ReadFileCall(set_filename) == FAULT)
            *set_filename = '\0';
          window(1, 1, 80, 24);
          break;
        case 2 :
          if (*set_filename != NULL)
            ParamChangeCall();
          break;
        case 3 :
          if (*set_filename != NULL)
            FileChangeCall();
          break;
        case 5 :
          if (*set_filename != NULL)
            Armax_Call();
          break;
        case 8 :
          if (*set_filename != NULL)
            Remake_File();
          break;
        case 10 :
          Help();
          break;
        case 27 :
          gotoxy(60, 2);
          textcolor(YELLOW);
          cprintf("END ... Sure ? (y/n)");
          if ( toupper( getch() ) == 'Y' )
            return 0;
          break;
      }
    }
}

/*-------------------------------------------------------------------*/
int Hard_Error(int errval, int ax, int bp, int si)
{
    unsigned   di;
    di = _DI;

    if (ax < 0) {
      sprintf(tmp_buff1,"Device  Error Has Happened.");
      Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
      hardretn(ABORT);
      return FAULT;
    }

    error_drive = (ax & 0x00FF) + 65;
    DriveErrorNumber = di & 0x00FF;
    DriveError = ON;
    hardretn(IGNORE);
    return FAULT;
}
/*-------------------------------------------------------------------*/
void  Drive_Message(void)
{
    DriveError = OFF;

    if (DriveErrorNumber == 2)
      sprintf(tmp_buff1, "%c: Drive Is Not Ready.", error_drive);
    else
      sprintf(tmp_buff1, "Something Is Wrong with The %c: Drive.",
                                    error_drive);
    Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
}
```

AMX1MENU.C

```
#include <stdio.h>
#include <conio.h>
#include <pc98.h>
#include <dos.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include "armax1.h"

struct key_code {
    unsigned  code;
    unsigned  output;
};

/* ------------------------------------------------------------------ */
void Title(void)
{
    extern char set_filename[];
    int  i, filename_length;
    char *ptr1;
    window(1, 1, 80 ,24);
    clrscr();
    textattr(LIGHTCYAN | REVERSE);
    gotoxy(1, 1);
    cprintf("  ARMAX1 ON-LINE IDENTIFICATION BY RECURSIVE PREDICTION ERROR METHOD
Ver1.01  ");
    gotoxy(1, 2);
    textattr(WHITE);
    cprintf("Set  Up File Name  = ");
    textcolor(LIGHTMAGENTA);
    ptr1 = String_Adjust_1(set_filename,  23);
    cputs(ptr1);
    textattr(WHITE);
}
```

```
/* ---------------------------------------------------------- */
void  FunctionKey(int param)
{
  extern char set_filename[];
  static char save_functionkey[11][16];
  int  i, j;
  char mainmenu_functionkey[11][16]  = {"",
                        " READ ", "CHANGE",
                        "OUTFIL", "",
                        " CALC ", "",
                        ""      , "REMAKE",
                        ""      , " HELP "};
  char paramchange_functionkey[11][16]  = {"",
                        "SKIP 0", "MAXNUM",
                        ""      , "U-NULL",
                        "I-NULL", "",
                        ""      , "PLRSET",
                        "ELSSET", "RMLSET"};
  char filechange_functionkey[11][16]  = {"",
                        "POLE_N", "P/L_NU",
                        ""      , ""
                        ""      , ""
                        ""      , ""
                        ""      , ""};
  char repmcall1_functionkey[11][16]  = {"",
                        " STOP ", "",
                        ""      , ""
                        ""      , ""
                        ""      , ""
                        ""      , ""};
  char repmcall2_functionkey[11][16]  = {"",
                        " STOP ", "",
                        ""      , ""
                        ""      , ""
                        ""      , ""
                        "ASTOP1", "ASTOP2"};
  char empty_functionkey[11][16]  = {"",
                        "", ""
                        "", ""
                        "", ""
                        "", ""
                        "", ""};
  switch (param){
    case DOSSAVE:
      for (i = 1; i <= 10; i++)
        pc98getfkey(i,  &save_functionkey[i][0]);
      break;
    case MAINMENU:
      if (*set_filename  == NULL)
        for (i = 2; i <= 8; i++)
          for (j = 0; j <= 15; j++)
            mainmenu_functionkey[i][j]  = '¥0';
      for (i = 1; i <= 10; i++)
        pc98setfkey(i,  &mainmenu_functionkey[i][0]);
      break;
    case PARAMCHANGE:
      for (i = 1; i <= 10; i++)
        pc98setfkey(i,  &paramchange_functionkey[i][0]);
      break;
    case FILECHANGE:
      for (i = 1; i <= 10; i++)
        pc98setfkey(i,  &filechange_functionkey[i][0]);
      break;
    case RPEMCALL1:
      for (i = 1; i <= 10; i++)
        pc98setfkey(i,  &repmcall1_functionkey[i][0]);
      break;
    case RPEMCALL2:
      for (i = 1; i <= 10; i++)
        pc98setfkey(i,  &repmcall2_functionkey[i][0]);
      break;
    case EMPTY:
      for (i = 1; i <= 10; i++)
        pc98setfkey(i,  &empty_functionkey[i][0]);
      break;
    case DOSLOAD:
      for (i = 1; i <= 10; i++)
        pc98setfkey(i,  &save_functionkey[i][0]);
       break;
  }
}

/* ---------------------------------------------------------- */
int Key_Judge(void)
{
  struct key_code  set[39] = { {0x6200, 1}, {0x6300, 2}, {0x6400, 3},
                {0x6500, 4}, {0x6600, 5}, {0x6700, 6},
                {0x6800, 7}, {0x6900, 8}, {0x6A00, 9},
                {0x6B00, 10},
                {0x001B, 27},
                {0x4A31, '1'}, {0x4B32, '2'},
                {0x4C33, '3'}, {0x4634, '4'},
                {0x4735, '5'}, {0x4836, '6'},
                {0x4237, '7'}, {0x4338, '8'},
                {0x4439, '9'}, {0x4E30, '0'},
                {0x322F, '/'}, {0x412F, '/'},
                {0x2D42, 'B'}, {0x2D62, 'B'},
                {0x2046, 'F'}, {0x2066, 'F'},
                {0x1749, 'I'}, {0x1769, 'I'},
                {0x2E4E, 'N'}, {0x2E6E, 'N'},
                {0x1E53, 'S'}, {0x1E73, 'S'},
                {0x1655, 'U'}, {0x1675, 'U'},
                {0x2C56, 'V'}, {0x2C76, 'V'},
                {0x1559, 'Y'}, {0x1579, 'Y'} };
  int         i;
  unsigned    code;
  delay(300);
  pc98key(3);
  code = pc98key(0);
  for (i = 0; i < 39; i++)
    if (code == set[i].code)
      return set[i].output;
  return -1;
}

/* ---------------------------------------------------------- */
void  Message(char *ptr, int param, int color, int y)
{
  extern int  AutoSystem;
  char     message_buff[128];
  char     *ptr1;
  int      i, add_space, length, fine;
  ptr1 = message_buff;
  if (param == CONTINUE && AutoSystem  == OFF)
    strcat(ptr, "    [HIT ANY KEY]");
  length = strlen(ptr);
  add_space = (80 - length) / 2;
```

```
    for (i = 1; i <= add_space; i++, ptr1++)
      *ptr1 = ' ';
    for (i = 1; i <= length; i++, ptr++, ptr1++)
      *ptr1 = *ptr;
    for (i = 1; i <= add_space; i++, ptr1++)
      *ptr1 = ' ';
    *ptr1 = '¥0';
    fine = 80 - strlen(message_buff);
    for (i = 1; i <= fine; i++, ptr1++)
      *ptr1 = ' ';
    *ptr1 = '¥0';
    gotoxy(1, y);
    textattr(REVERSE | color);
    cprintf("%s", message_buff);
    textattr(WHITE);
    if (AutoSystem == ON)
      delay(1000);
    else
      if (param == CONTINUE)
        getch();
}
```

AMX1AUTO.C

```
#include <string.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include "armax1.h"

/* ---------------------------------------------------------------- */
int AutoMain(char *argv[])
{
  extern int    AutoStop;
  extern char   set_filename[];
  FILE       *fcontrol, *freport;
  int        count = 0, check;
  char       control_filename[128], report_filename[128],
             message_buff[128], *ptr1, answer;
  strcpy(control_filename, argv[1]);
  strcpy(report_filename, argv[2]);

  if ( (fcontrol = fopen(control_filename, "rt")) == NULL) {
    ptr1 = String_Adjust_1(control_filename, 20);
    sprintf(message_buff, "Auto Control File [ %s ] Cannot Open",
                          ptr1);
    Message(message_buff, FREE, LIGHTRED, 5);
    return FAULT;
  }
  if ( (freport = fopen(report_filename, "wt")) == NULL) {
    ptr1 = String_Adjust_1(control_filename, 20);
    sprintf(message_buff, "Auto System Report File [ %s ] Cannot Open",
                          ptr1);
    Message(message_buff, FREE, LIGHTRED, 5);
    return FAULT;
  }

  while (!feof(fcontrol)) {
    count++;
    fscanf(fcontrol, "%s", set_filename);
    if (feof(fcontrol))
      break;
    Title();
    if (ReadFileCall(set_filename) == FAULT){
      sprintf(message_buff, "On Reading Set Up File");
      Report_Writing(count, freport, set_filename, 0,
                             message_buff);
      continue;
    }
    check = Armax_Call();
    File_Close();
    if (check == FAULT){
      sprintf(message_buff, "On Start Check Routine");
      Report_Writing(count, freport, set_filename, 0,
                             message_buff);
      continue;
    }
    Report_Writing(count, freport, set_filename, SUCCESS,
                             message_buff);
    if (AutoStop == ON) {
      AutoStop = OFF;
      gotoxy(18, 23);
      textattr(YELLOW | BLINK | REVERSE);
      cprintf(" Will You Stop This Curriculum Really ? (Y/N) ¥r");
      textattr(WHITE);
      answer = '¥0';
      while (answer != 'Y' && answer != 'N')
        answer = toupper(getch());
      if (answer == 'Y') {
        count++;
        sprintf(message_buff,
                "This Curriculum Was Stopped by Force");
        Report_Writing(count, freport, "Manual Termintion",
                       SUCCESS, message_buff);
        break;
      }
    }
  }

  fclose(fcontrol);
  fclose(freport);
  return SUCCESS;
}

/* ---------------------------------------------------------------- */
void Report_Writing(int count, FILE *fil, char *set_filename,
                    int result, char *report_message)
{
  char  *string1[] = {"ERROR","SUCCESS"};
  fprintf(fil, "Process Number  : %d¥n", count);
  fprintf(fil, "Set Up File Name : %s¥n", set_filename);
  fprintf(fil, "Result        : %s¥n", string1[result]);
  if (result == 0)
    fprintf(fil, "Comment       : %s¥n", report_message);
  fprintf(fil, "¥n¥n");
}
```

AMX1READ.C

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
```

```
#include "armax1.h"

/* ------------------------------------------------------------ */
int ReadFileCall(char*filename)
{
  extern int    AutoSystem, DriveError;
  extern char   error_drive, out_filename[], pole_filename[],
          pl_filename[], tmp_buff1[], tmp_buff2[];
  extern long   abs_useable_number, useable_number, skip_number,
          identification_number, interval_number;
  extern FILE   *fin_u, *fin_y, *fin_init;
  extern BINARY_FILE    u_file, y_file;
  extern INIT_FILE      init_file;
  extern RPEM_SUB       subp;
  extern RPEM_PARAMETER rpemp;
  FILE  *fin_set;
  char  c_dummy, *ptr1, input_buff[256];
  int   i, i_dummy;
  double d_dummy;
  FunctionKey(EMPTY);

  if (AutoSystem == OFF){
    textcolor(YELLOW);
    gotoxy(55, 2);
    cprintf("PATH = ");
    gets(input_buff);
    if (*input_buff == '.')
      return 0;
    i = Getfile(input_buff, PARAMETERFILE);
    switch (i) {
      case NOTHING :
        sprintf(tmp_buff1,
        "Fitting File Dose Not Exist on The Specified PATH.");
        Message(tmp_buff1, CONTINUE, LIGHTRED,MAXBOTTOM);
        return 0;
      case FAULT:
        return 0;
    }
    strcpy(filename, input_buff);
  }

  /* Set Up File Open */
  if ( (fin_set = fopen(filename, "rt")) == NULL ) {
    if (DriveError == ON)
      Drive_Message();
    ptr1 = String_Adjust_1(filename, 20);
    sprintf(tmp_buff1, "Cannot Open Set Up File [ %s ].", filename);
    Message(tmp_buff1, CONTINUE, LIGHTRED,MAXBOTTOM);
    return FAULT;
  }
  window(1, 4, 80, 24);
  clrscr();
  gotoxy(1, 1);

  /* Reading of Identification Order */
  if (Std_Reading(fin_set, input_buff, 0) == FAULT)
    return FAULT;
  rpemp.n = atoi(input_buff);
  if (rpemp.n < 0 || rpemp.n > 9 ||
              *input_buff < '0' || *input_buff > '9') {
    sprintf(tmp_buff1, "Illegal Value as Identification Order.");
    Message(tmp_buff1, CONTINUE, LIGHTRED,MAXBOTTOM);
    fclose(fin_set);
    return FAULT;
  }

  /* Reading of Record Length of Binary Time Series */
  if (Std_Reading(fin_set,input_buff, 1) == FAULT)
    return FAULT;
  u_file.byte = y_file.byte = atoi(input_buff);
  if (u_file.byte != 2 && u_file.byte != 8) {
    sprintf(tmp_buff1,
        "Illegal Value as Record Length of Time Series Data.");
    Message(tmp_buff1, CONTINUE, LIGHTRED,MAXBOTTOM);
    fclose(fin_set);
    return FAULT;
  }

  /* Reading of U Filename */
  *u_file.name = NULL;
  if (Std_Reading(fin_set,input_buff, 2) == FAULT)
    return FAULT;
  if (strcmp(input_buff, "NULL") == 0)
    useable_number = 10000000;
  else {
    if ( (fin_u = fopen(input_buff, "rb")) == NULL ) {
      if (DriveError == ON) {
        Drive_Message();
        return FAULT;
      }
      sprintf(tmp_buff1,
      "Cannot Open Time Series Data File Inputted to System.");
      Message(tmp_buff1, CONTINUE, LIGHTRED,MAXBOTTOM);
      fclose(fin_set);
      return FAULT;
    } else {
      u_file.number =
          Binary_File_Check(fin_u, u_file.byte, input_buff);
      fclose(fin_u);
      if (u_file.number == FAULT){
        fclose(fin_set);
        return FAULT;
      }
      useable_number = u_file.number;
      strcpy(u_file.name, input_buff);
    }
  }

  /* Reading of Y Filename */
  if (Std_Reading(fin_set,input_buff, 3) == FAULT)
    return FAULT;
  if ( (fin_y = fopen(input_buff, "rb")) == NULL ) {
    if (DriveError == ON) {
      Drive_Message();
      return FAULT;
    }
    sprintf(tmp_buff1,
      "Cannot Open Time Series Data File Outputted from System.");
    Message(tmp_buff1, CONTINUE, LIGHTRED,MAXBOTTOM);
    fclose(fin_set);
    return FAULT;
  } else {
    strcpy(y_file.name, input_buff);
    y_file.number =
        Binary_File_Check(fin_y, y_file.byte, y_file.name);
    fclose(fin_y);
    if (y_file.number == FAULT){
      fclose(fin_set);
```

```
        return FAULT;
      }
      if (y_file.number < useable_number)
        abs_useable_number = useable_number = y_file.number;
    }

    /* Reading of Initial Parameter Setting Filename */
    *init_file.name = NULL;
    if (Std_Reading(fin_set,input_buff, 4) == FAULT)
      return FAULT;
    if (strcmp(input_buff, "NULL") != 0) {
      if ( (fin_init = fopen(input_buff, "rt")) == NULL ) {
        if (DriveError == ON) {
          Drive_Message();
          return FAULT;
        }
        sprintf(tmp_buff1, "Canont Open Initial Parameter File.");
        Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
        fclose(fin_set);
        return FAULT;
      }
      fscanf(fin_init, "%s %s", tmp_buff1, tmp_buff2);
      fclose(fin_init);
      if (strcmp(tmp_buff1, "INITIAL_PARAMETER_FILE") != 0) {
        sprintf(tmp_buff1,
        "First Declaration of The Initial Parameter File Is Unsuitable.");
        Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
        return FAULT;
      }
      if (((int)(*tmp_buff2) - 48) != rpemp.n) {
        sprintf(tmp_buff1,
        "There Is Discrepancy As to Identification Order.");
        Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
        return FAULT;
      }
      strcpy(init_file.name, input_buff);
      init_file.order = (int)(*tmp_buff2) - 48;
    }

    /* Reading of Number of Skipping Data */
    if (Std_Reading(fin_set,input_buff, 5) == FAULT)
      return FAULT;
    skip_number = atol(input_buff);
    if (skip_number < 0 || skip_number > useable_number) {
      sprintf(tmp_buff1, "Illegal Value as Number of Skipping Data.");
      Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
      fclose(fin_set);
      return FAULT;
    }
    useable_number -= skip_number;

    /* Reading of Number of Identification */
    if (Std_Reading(fin_set,input_buff, 6) == FAULT)
      return FAULT;
    identification_number = atol(input_buff);
    if (identification_number <= 0 ||
                identification_number > useable_number) {
      sprintf(tmp_buff1, "Illegal Value as Number of Identification.");
      Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
      fclose(fin_set);
      return FAULT;
    }

    /* Reading of Forgetting Factor Mu */
    if (Std_Reading(fin_set,input_buff, 7) == FAULT)
      return FAULT;
    subp.lambda_mu = atof(input_buff);
    if (subp.lambda_mu <= 0 || subp.lambda_mu > 1 ||
              *input_buff < '0' || *input_buff > '9') {
      sprintf(tmp_buff1,
          "Illegal Value as Constant Part of Forgetting Factor.");
      Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
      fclose(fin_set);
      return FAULT;
    }

    /* Reading of Forgetting Factor Init */
    if (Std_Reading(fin_set,input_buff, 8) == FAULT)
      return FAULT;
    subp.lambda_init = atof(input_buff);
    if (subp.lambda_init <= 0 || subp.lambda_init > 1 ||
              *input_buff < '0' || *input_buff > '9') {
      sprintf(tmp_buff1, "Illegal Value as Initial Forgetting Factor.");
      Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
      fclose(fin_set);
      return FAULT;
    }

    /* Reading of Contraction Factor Mu */
    if (Std_Reading(fin_set,input_buff, 9) == FAULT)
      return FAULT;
    subp.contraction_mu = atof(input_buff);
    if (subp.contraction_mu < 0 || subp.contraction_mu > 1 ||
              *input_buff < '0' || *input_buff > '9') {
      sprintf(tmp_buff1, "Illegal Value as Contraction Factor Mu.");
      Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
      fclose(fin_set);
      return FAULT;
    }

    /* Reading of Prevention Value */
    if (Std_Reading(fin_set,input_buff, 10) == FAULT)
      return FAULT;
    rpemp.c = atof(input_buff);
    if (rpemp.c <= 0 || *input_buff < '0' || *input_buff > '9') {
      sprintf(tmp_buff1,
          "Illegal Value as Constant for Divergent Prevention.");
      Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
      fclose(fin_set);
      return FAULT;
    }

    /* Reading of Stable Check Switch */
    if (Std_Reading(fin_set,input_buff, 11) == FAULT)
      return FAULT;
    ptr1 = input_buff;
    ptr1++;
    c_dummy = toupper(*ptr1);
    switch (c_dummy) {
      case 'N' :
        rpemp.stable = ON;
        break;
      case 'F' :
        rpemp.stable = OFF;
        break;
      default :
        sprintf(tmp_buff1, "Illegal Switch as Stable Check.");
        Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
```

```
    fclose(fin_set);
    return FAULT;
}

/* Reading of Initialization Value of P Matrix */
if (Std_Reading(fin_set,input_buff, 12) == FAULT)
  return FAULT;
rpemp.pmatrix0 = atof(input_buff);
if (rpemp.pmatrix0 <= 0 || *input_buff < '0' || *input_buff > '9') {
  sprintf(tmp_buff1,
    "Illegal Value as Diagonal Elements of Initial P Matrix.");
  Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
  fclose(fin_set);
  return FAULT;
}

/* Reading of Activation Method Switch */
if (Std_Reading(fin_set,input_buff, 13) == FAULT)
  return FAULT;
ptr1 = input_buff;
ptr1 += 2;
c_dummy = toupper(*ptr1);
switch (c_dummy) {
  case 'D':
    subp.activation = PMDA;
    break;
  case 'A':
    subp.activation = PMA;
    break;
  case 'F':
    subp.activation = OFF;
    break;
  default :
    sprintf(tmp_buff1,
        "Illegal Switch as Selection of Activation Method.");
    Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
    fclose(fin_set);
    return FAULT;
}

/* Reading of Activation Interval */
if (Std_Reading(fin_set,input_buff, 14) == FAULT)
  return FAULT;
subp.activation_interval = atol(input_buff);
if (subp.activation_interval <= 0 ||
          subp.activation_interval > useable_number) {
  sprintf(tmp_buff1,
        "Illegal Value as Number of Activation Interval.");
  Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
  fclose(fin_set);
  return FAULT;
}

/* Reading of Activation Power*/
if (Std_Reading(fin_set,input_buff, 15) == FAULT)
  return FAULT;
subp.activation_power = atof(input_buff);
if (subp.activation_power < 1 || *input_buff < '0'
                    || *input_buff > '9') {
  sprintf(tmp_buff1, "Illegal Value as Activation Power.");
  Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
  fclose(fin_set);
  return FAULT;
}
```



```
/* Reading of Output Filename of Result*/
if (Std_Reading(fin_set,input_buff, 16) == FAULT)
  return FAULT;
if ( Filename_Check(input_buff, 13, 0) == FAULT) {
  sprintf(tmp_buff1, "Unsuitable Output File Name. Please Ovey MS-DOS Format.");
  Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
  fclose(fin_set);
  return FAULT;
}
strcpy(out_filename, input_buff);

/* Reading of Output Interval */
if (Std_Reading(fin_set,input_buff, 17) == FAULT)
  return FAULT;
interval_number = atol(input_buff);
if (interval_number <= 0 || interval_number > useable_number) {
  sprintf(tmp_buff1, "Illegal Value as Number of Interval.");
  Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
  fclose(fin_set);
  return FAULT;
}

/* Reading of Output Filename of Binary Pole */
*pole_filename = NULL;
if (Std_Reading(fin_set,input_buff, 18) == FAULT)
  return FAULT;
if (strcmp(input_buff, "NULL") != 0) {
  if ( Filename_Check(input_buff, 9, 1) == FAULT) {
    sprintf(tmp_buff1, "Unsuitable Output File Name. Please Ovey MS-DOS Format.");
    Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
    fclose(fin_set);
    return FAULT;
  }
  strcpy(pole_filename, input_buff);
}
*pl_filename = NULL;
if (Std_Reading(fin_set,input_buff, 19) == FAULT)
  return FAULT;
if (strcmp(input_buff, "NULL") != 0) {
  if ( Filename_Check(input_buff, 13, 0) == FAULT) {
    sprintf(tmp_buff1,
      "Unsuitable P/L File Name. Please Obey MS-DOS Format.");
    Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
    fclose(fin_set);
    return FAULT;
  }
  strcpy(pl_filename, input_buff);
}
fclose(fin_set);
return SUCCESS;
}

/* ---------------------------------------------------------- */
long Binary_File_Check(FILE *fin, int byte, char *ptr1)
{
  extern char  tmp_buff1[];
  char  *ptr2;
  int  i_dummy;
  long  file_size, data_number;
  double d1_dummy, d2_dummy;
  fseek(fin, 0, SEEK_END);
  file_size = ftell(fin);
  d1_dummy = (double)file_size / (double)byte;
```

```
    d2_dummy = floor(d1_dummy);
    fread(ptr2, 1, 1, fin);
    if (d1_dummy != d2_dummy || *ptr2 == EOF) {
      ptr2 = String_Adjust_1(ptr1, 13);
      sprintf(tmp_buff1,
      "Probably, [ %s ] Is Not Binary Data File of %d Byte", ptr2, byte);
      Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
      return FAULT;
    }
    data_number = file_size / byte;
    return data_number;
}

/* ---------------------------------------------------------- */
int Std_Reading(FILE *fil, char *ptr1, int counter)
{
    extern char   tmp_buff1[], *read_param[];
    int       check;
    check = fscanf(fil, "%s %s", tmp_buff1, ptr1);
    cprintf("%s %s ¥r¥n", tmp_buff1, ptr1);

    if ( check != 2 ) {
      sprintf(tmp_buff1, "Format of Set Up File Is Unsuitable");
      Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
      fclose(fil);
      return FAULT;
    }
    if (strcmp(tmp_buff1, read_param[counter]) != 0) {
      sprintf(tmp_buff1, "Format of Set Up File Is Unsuitable");
      Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
      fclose(fil);
      return FAULT;
    }
    return SUCCESS;
}
```


AMX1DIR.C

```
#include <dir.h>
#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <pc98.h>
#include "armax1.h"
#define MAX_DIR5*19
#define MAX_X  5

char   buffer[MAX_DIR][15];
char   search_dir[120], search_path[120];
char   *pdir[5*19];
char   drive[3], dir[100], name[9], ext[5];
struct  ffblk  ffblk;

/* ---------------------------------------------------------- */
int Getdir(char *search_dir, char *search_path, char *dptr[], int max_dir, int file_kind)
{
    int check, done, i = -1, j, k;
    char *wdptr;
    done = findfirst(search_path, &ffblk, FA_RDONLY);
    while (!done){
      i++;
      if (max_dir <= i)
        break;
      check = File_Selection(ffblk.ff_name, file_kind);
      if (check == PICK)
        dptr[i] = strcpy(buffer[i], ffblk.ff_name );
      else
        i--;
      done = findnext(&ffblk);
    }
    done = findfirst(search_dir, &ffblk, FA_DIREC);
    while (!done) {
      if (max_dir <= i)
        break;
      if (ffblk.ff_attrib & FA_DIREC){
        i++;
        dptr[i] = strcpy(buffer[i], ffblk.ff_name );
        strcat(dptr[i], "¥¥");
      }
      done = findnext(&ffblk);
    }
    i++;
    for(j = 0; j < i; j++){
      for(k = j; k < i; k++){
        if (strcmp(dptr[j], dptr[k]) > 0){
          wdptr = dptr[j];
          dptr[j] = dptr[k];
          dptr[k] = wdptr;
        }
      }
    }
    return i;
}

/* ---------------------------------------------------------- */
int GetfileSub(char *finam, char *drive, char *dir, char *name, char *ext, int file_kind)
{
    int f, i, k;
    clrscr();
    Title();
    gotoxy(40, 2);
    textcolor(YELLOW);
    cprintf("SELECT OBJECTIVE FILE ::: OK → [RETURN]");
    delay(100);
    fnmerge(search_dir, drive, dir, "*", "");
    fnmerge(search_path, drive, dir, name, ext);
    f = Getdir(search_dir, search_path, pdir, 5*19, file_kind);
    textcolor(WHITE);
    for(i=0; i<f; i++) {
      gotoxy( (i % MAX_X)*16 + 1, (i / MAX_X) + 5 );
      cprintf("%-14s", pdir[i]);
    }
    if (f == 0)
      return NOTHING;
    i = 0;
    do {
      textattr(LIGHTGREEN | REVERSE);
      gotoxy( (i % MAX_X)*16 + 1, (i / MAX_X) + 5 );
      cprintf("%-14s", pdir[i]);
      pc98key(3);
      k = pc98key(0) / 256;
      textattr(WHITE);
      gotoxy( (i % MAX_X)*16 + 1, (i / MAX_X) + 5 );
      cprintf("%-14s", pdir[i]);
      delay(200);
```

```
    switch(k) {
      case 0x3c : if (i < f-1) i++; break;
      case 0x3b : if (i > 0)  i--; break;
      case 0x3a : if (i >= MAX_X) i -= MAX_X; break;
      case 0x3d : if (i+MAX_X <= f-1) i += MAX_X; break;
      case 0x00 : return FAULT;
    }
  } while (k != 0x1c);
  if ( *(pdir[i]+strlen(pdir[i])-1)  == '¥' ){
    *(pdir[i] + strlen(pdir[i])-1) = 0;
    if (strcmp(dir,"")!=0)  strcat(dir,"¥");
    strcat(dir,pdir[i]);
    i = GetfileSub(finam,drive,dir,name,ext,file_kind);
    return i;
  }
  fnmerge(finam,  drive, dir, "", "");
  strcat(finam,  pdir[i]);
  return SUCCESS;
}

/* ---------------------------------------------------------------- */
int  Getfile(char *finam,  int file_kind)
{
  int i, flags;
  flags = fnsplit(finam,  drive, dir, name, ext);
  if (!(flags & (FILENAME | EXTENSION))) {
    strcpy(name,  "*");
    strcpy(ext,  ".*");
  }
  i = GetfileSub(finam,  drive, dir, name, ext, file_kind);
  return i;
}

/* ---------------------------------------------------------------- */
int  File_Selection(char *ptr, int file_kind)
{
  int i, search_end;
  char *extract_extention  [] = {"C", "PAS", "FOR", "EXE", "OBJ", "FFT"
                    ,"BAT", "BAK", "PRJ", "COM", "H", "BAS"
                    ,"MAK", "DSK", "CFG", "SYS", "DOC"
                    ,"111", "222", "333","BIN"};
  char *select_extention  [] = {"111", "222", "333", "444", "555",
                    "666", "777", "888", "999"};
  search_end = 0;
  switch (file_kind) {
    case PARAMETERFILE:
      search_end++;
    case BINARYFILE :
      search_end += 20;
      while(*ptr != '¥0' && *ptr != '.')
        ptr++;
      if (*ptr == '¥0')
        return PICK;
      ptr++;
      for (i = 0; i < search_end; i++)
        if ( (strcmp(ptr,  extract_extention[i]))  == 0 )
          return DUMP;
      return PICK;
    case INITFILE :
      while(*ptr != '¥0' && *ptr != '.')
        ptr++;
      if (*ptr == '¥0')
        return DUMP;
      ptr++;
      for (i = 0; i < 9; i++)
        if ( (strcmp(ptr,  select_extention[i]))  == 0 )
          return PICK;
      return DUMP;
  }
  return SUCCESS;
}
#include <dir.h>
#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <pc98.h>
#include "armax1.h"
#define MAX_DIR 5*19
#define MAX_X  5

char  buffer[MAX_DIR][15];
char  search_dir[120],  search_path[120];
char  *pdir[5*19];
char  drive[3],  dir[100],  name[9],  ext[5];
struct  ffblk  ffblk;

/* ---------------------------------------------------------------- */
int  Getdir(char *search_dir,  char *search_path,  char *dptr[],  int max_dir,  int file_kind)
{
  int  check,  done,  i = -1,  j,  k;
  char *wdptr;
  done = findfirst(search_path,  &ffblk,  FA_RDONLY);
  while (!done){
    i++;
    if (max_dir <= i)
      break;
    check = File_Selection(ffblk.ff_name,  file_kind);
    if (check == PICK)
      dptr[i] = strcpy(buffer[i],  ffblk.ff_name );
    else
      i--;
    done = findnext(&ffblk);
  }
  done = findfirst(search_dir,  &ffblk,  FA_DIREC);
  while (!done) {
    if (max_dir <= i)
      break;
    if (ffblk.ff_attrib  & FA_DIREC){
      i++;
      dptr[i] = strcpy(buffer[i],  ffblk.ff_name );
      strcat(dptr[i],  "¥¥");
    }
    done = findnext(&ffblk);
  }
  i++;
  for(j = 0; j < i; j++){
    for(k = j; k < i; k++){
      if (strcmp(dptr[j],  dptr[k]) > 0){
        wdptr = dptr[j];
        dptr[j] = dptr[k];
        dptr[k] = wdptr;
      }
    }
  }
  return i;
}
```

```
/* ---------------------------------------------------------- */
int GetfileSub(char *finam, char *drive, char *dir, char *name, char *ext, int file_kind)
{
  int f, i, k;
  clrscr();
  Title();
  gotoxy(40, 2);
  textcolor(YELLOW);
  cprintf("SELECT OBJECTIVE FILE ::: OK --> [RETURN]");
  delay(100);
  fnmerge(search_dir, drive, dir, "*", "");
  fnmerge(search_path, drive, dir, name, ext);
  f = Getdir(search_dir, search_path, pdir, 5*19, file_kind);
  textcolor(WHITE);
  for(i=0; i<f; i++) {
    gotoxy( (i % MAX_X)*16 + 1, (i / MAX_X) + 5 );
    cprintf("%-14s", pdir[i]);
  }
  if (f == 0)
    return NOTHING;
  i = 0;
  do {
    textattr(LIGHTGREEN | REVERSE);
    gotoxy( (i % MAX_X)*16 + 1, (i / MAX_X) + 5 );
    cprintf("%-14s", pdir[i]);
    pc98key(3);
    k = pc98key(0) / 256;
    textattr(WHITE);
    gotoxy( (i % MAX_X)*16 + 1, (i / MAX_X) + 5 );
    cprintf("%-14s", pdir[i]);
    delay(200);

    switch(k) {
      case 0x3c : if (i < f-1) i++; break;
      case 0x3b : if (i > 0)  i--; break;
      case 0x3a : if (i >= MAX_X) i -= MAX_X; break;
      case 0x3d : if (i+MAX_X <= f-1) i += MAX_X; break;
      case 0x00 : return FAULT;
    }
  } while (k != 0x1c);
  if ( *(pdir[i]+strlen(pdir[i])-1)  == '¥¥' ){
    *(pdir[i] + strlen(pdir[i])-1)  = 0;
    if (strcmp(dir,"")!=0)  strcat(dir,"¥¥");
    strcat(dir,pdir[i]);
    i = GetfileSub(finam,drive,dir,name,ext,file_kind);
    return i;
  }
  fnmerge(finam, drive, dir, "", "");
  strcat(finam, pdir[i]);
  return SUCCESS;
}

/* ---------------------------------------------------------- */
int Getfile(char *finam, int file_kind)
{
  int i, flags;
  flags = fnsplit(finam, drive, dir, name, ext);
  if (!(flags & (FILENAME | EXTENSION))) {
    strcpy(name, "*");
    strcpy(ext, ".*");
  }
  i = GetfileSub(finam, drive, dir, name, ext, file_kind);
  return i;
}

/* ---------------------------------------------------------- */
int File_Selection(char *ptr, int file_kind)
{
  int i, search_end;
  char *extract_extention  [] = {"C", "PAS", "FOR", "EXE", "OBJ", "FFT"
                ,"BAT", "BAK", "PRJ", "COM", "H", "BAS"
                ,"MAK", "DSK", "CFG", "SYS", "DOC"
                ,"111", "222", "333", "BIN"};
  char *select_extention  [] = {"111", "222", "333", "444", "555",
                "666", "777", "888", "999"};
  search_end = 0;
  switch (file_kind) {
    case PARAMETERFILE:
      search_end++;
    case BINARYFILE :
      search_end += 20;
      while(*ptr != '¥0' && *ptr != '.')
        ptr++;
      if (*ptr == '¥0')
        return PICK;
      ptr++;
      for (i = 0; i < search_end; i++)
        if ( (strcmp(ptr, extract_extention[i]))  == 0 )
          return DUMP;
      return PICK;
    case INITFILE :
      while(*ptr != '¥0' && *ptr != '.')
        ptr++;
      if (*ptr == '¥0')
        return DUMP;
      ptr++;
      for (i = 0; i < 9; i++)
        if ( (strcmp(ptr, select_extention[i]))  == 0 )
          return PICK;
      return DUMP;
  }
  return SUCCESS;
}
```

AMX1PARA.C

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <math.h>
#include <dos.h>
#include <dir.h>
#include "armax1.h"

/* ---------------------------------------------------------- */
int ParamChangeCall(void)
{
  extern char   tmp_buff1[], tmp_buff2[], error_drive;
  extern long   abs_useable_number, useable_number, skip_number,
          identification_number;
  extern FILE   *fin_u, *fin_y, *fin_init;
  extern BINARY_FILE    u_file, y_file;
  extern INIT_FILE      init_file;
```

```
extern RPEM_SUB      subp;
extern RPEM_PARAMETER rpemp;
char  *string1[] ={"", "RPE", "PLR", "ELS", "RML"},
     *string2[] ={"      ", "& PMA ", "& PMDA"},
     *string3[] ={"OFF", "ON  "};
char  clear_buff[77] ={"                                                                    "};
char  c, *ptr1, input_buff[256];
int   i, j, k, method1, u_order;
long  l_dummy;
Crt_Reset();

while (1) {
  method1 = Method(&subp);
  if (*u_file.name != NULL)
    u_order = rpemp.n;
  else
    u_order = 0;

  window(1, 23, 80, 24);
  clrscr();
  window(1, 1, 80 ,24);

  gotoxy(1, 4);
  textcolor(LIGHTCYAN);
  cprintf("┌──────────────────────────────────────────────────────────────────────────┐");
  cprintf("│  ARMAX(%d, %d, %d) Identification   :  Using Algorithm = %s %s  │",
    rpemp.n, u_order, rpemp.n, string1[method1],
                  string2[subp.activation]);
  cprintf("└──────────────────────────────────────────────────────────────────────────┘");
  gotoxy(2, 7);
  printf("(0)  Order       → %d ", rpemp.n);
  gotoxy(44, 7);
  printf("(B)  Record Length  → %d Byte", y_file.byte);
  gotoxy(2, 9);
  if (*u_file.name == NULL) {
    textcolor(LIGHTGREEN);
    cprintf("(U)  U Filename   → UNDEFINED                    ");
  } else {
    ptr1 = String_Adjust_1(u_file.name, 35);
    printf("(U)  U Filename   → %s  %ld Data",
                    ptr1, u_file.number);
  }
  gotoxy(2, 10);
  if (*y_file.name == NULL) {
    textcolor(LIGHTRED);
    cprintf("(Y)  Y Filename   → UNDEFINED                    ");
  } else {
    ptr1 = String_Adjust_1(y_file.name, 35);
    printf("(Y)  Y Filename   → %s  %ld Data",
                    ptr1, y_file.number);
  }
  gotoxy(2, 11);
  if (*init_file.name == NULL) {
    textcolor(LIGHTGREEN);
    cprintf("(I)  Initial File  → UNDEFINED                   ");
  } else {
    if (rpemp.n != init_file.order)
      textcolor(LIGHTRED);
    else
      textcolor(WHITE);
    ptr1 = String_Adjust_1(init_file.name, 20);
    cprintf("(I)  Initial File  → %s  for (%d, %d, %d) Identification",
      ptr1, init_file.order, init_file.order, init_file.order);
  }
```

```
  gotoxy(2, 13);
  printf("(S)  Skip Data     → % -10ld", skip_number);
  gotoxy(2, 14);
  if (identification_number > useable_number)
    textcolor(LIGHTRED);
  else
    textcolor(WHITE);
  cprintf("(N)  Using Number  → %ld / ", identification_number);
  textcolor(LIGHTCYAN);
  cprintf("%ld [Useable Number]      ", useable_number);
  gotoxy(2, 16);
  printf("(1)  F. Factor Mu  → % -11.9f ", subp.lambda_mu);
  gotoxy(44, 16);
  printf("(2)  F. Factor Init → % -11.9f ", subp.lambda_init);
  gotoxy(2, 17);
  printf("(3)  C. Factor Mu  → % -11.9f ", subp.contraction_mu);
  gotoxy(44, 17);
  printf("(4)  Divergent Prev. → % -11.6G ", rpemp.c);
  gotoxy(2, 18);
  printf("(5)  Stability Check → %s", string3[rpemp.stable]);
  gotoxy(44, 18);
  if (*init_file.name != NULL) {
    textcolor(LIGHTGREEN);
    cprintf("( )  P Matrix Init. → INEFFECTIVE");
  } else
    printf("(6)  P Matrix Init. → % -11.6G ", rpemp.pmatrix0);
  gotoxy(2, 20);
  if (subp.activation == 0)
    ptr1 = string3[0];
  else {
    ptr1 = string2[subp.activation];
    ptr1 += 2;
  }
  printf("(7)  Activation    → %s", ptr1);
  gotoxy(44, 20);
  if (subp.activation == OFF){
    textcolor(LIGHTGREEN);
    cprintf("( )  Act. Interval  → INEFFECTIVE");
  } else {
    if (subp.activation_interval > identification_number)
      textcolor(YELLOW);
    else
      textcolor(WHITE);
    cprintf("(8)  Act. Interval  → % -10ld ",
                  subp.activation_interval);
  }
  gotoxy(2, 21);
  if (subp.activation == OFF){
    textcolor(LIGHTGREEN);
    cprintf("( )  Act. Power   → INEFFECTIVE");
  } else
    printf("(9)  Act. Power   → % -11.9f ",
                  subp.activation_power);
  gotoxy(2, 23);
  textcolor(YELLOW);
  j = Key_Judge();
  switch (j) {
    case '0' :
      cprintf("Identification  Order [1 ≦integer ≦8] → ");
      rpemp.n = Manual_Set_I(rpemp.n, 1, 8, EE, 20);
      break;
    case 'B' :
      if (y_file.byte == 2) {
        u_file.byte = y_file.byte = 8;
```

```
        u_file.number /= 4;
        y_file.number /= 4;
      } else {
        u_file.byte = y_file.byte = 2;
        u_file.number *= 4;
        y_file.number *= 4;
      }
      Useable_Renumber();
      gotoxy(2, 9);
      printf("%s", clear_buff);
      gotoxy(2, 10);
      printf("%s", clear_buff);
      break;
    case 'U':
      if (File_Reset(BINARYFILE, input_buff) == FAULT){
        Crt_Reset();
        break;
      }
      if ( (fin_u = fopen(input_buff, "rb")) == NULL ) {
        ptr1 = String_Adjust_1(input_buff, 20);
        sprintf(tmp_buff1,
            "Cannot Open Specified File [ %s ].", ptr1);
        Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
        break;
      }
      l_dummy =
        Binary_File_Check(fin_u, u_file.byte, input_buff);
      fclose(fin_u);
      if (l_dummy == FAULT){
        Crt_Reset();
        break;
      }
      strcpy(u_file.name, input_buff);
      u_file.number = l_dummy;
      Useable_Renumber();
      gotoxy(2, 9);
      printf("%s", clear_buff);
      Crt_Reset();
      break;
    case 'Y':
      if (File_Reset(BINARYFILE, input_buff) == FAULT){
        Crt_Reset();
        break;
      }
      if ( (fin_y = fopen(input_buff, "rb")) == NULL ) {
        ptr1 = String_Adjust_1(input_buff, 20);
        sprintf(tmp_buff1,
            "Cannot Open Specified File [ %s ].", ptr1);
        Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
        break;
      }
      l_dummy =
        Binary_File_Check(fin_y, y_file.byte, input_buff);
      fclose(fin_y);
      if (l_dummy == FAULT){
        Crt_Reset();
        break;
      }
      strcpy(y_file.name, input_buff);
      y_file.number = l_dummy;
      Useable_Renumber();
      gotoxy(2, 10);
      printf("%s", clear_buff);
      Crt_Reset();
      break;
    case 'I':
      if (File_Reset(INITFILE, input_buff) == FAULT){
        Crt_Reset();
        break;
      }
      if ( (fin_init = fopen(input_buff, "rt")) == NULL ) {
        ptr1 = String_Adjust_1(input_buff, 20);
        sprintf(tmp_buff1,
            "Cannot Open Specified File [ %s ].", ptr1);
        Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
        break;
      }
      fscanf(fin_init, "%s %s", tmp_buff1, tmp_buff2);
      if (strcmp(tmp_buff1, "INITIAL_PARAMETER_FILE") != 0) {
        sprintf(tmp_buff1,
        "First Declaration of The Initial Parameter File Is Unsuitable.");
        Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
        fclose(fin_init);
        Crt_Reset();
        break;
      }
      if (*tmp_buff2 <= '0' || *tmp_buff2 >= '9') {
        sprintf(tmp_buff1,
        "Second Declaration of The Initial Parameter File Is Unsuitable.");
        Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
        fclose(fin_init);
        Crt_Reset();
        break;
      }
      fclose(fin_init);
      strcpy(init_file.name, input_buff);
      init_file.order = (int)(*tmp_buff2) - 48;
      Crt_Reset();
      break;
    case 'S':
      cprintf("Number of Skipping Data [0 ≦ integer ≦ %ld] → ",
                    (abs_useable_number-1));
      skip_number =
        Manual_Set_L(skip_number, 0,
              (abs_useable_number-1), EE, 20);
      Useable_Renumber();
      break;
    case 'N':
      cprintf("Number of Identification [1 ≦ integer ≦ %ld] → ",
                          useable_number);
      identification_number =
          Manual_Set_L(identification_number, 1,
                    useable_number, EE, 20);
      break;
    case 'f':
      cprintf("Forgetting Factor Mu [0 < real ≦ 1] → ");
      subp.lambda_mu =
          Manual_Set_D(subp.lambda_mu, 0.0, 1.0, NE, 20);
      break;
    case '2':
      cprintf("Forgetting Factor Init [0 < real ≦ 1] → ");
      subp.lambda_init =
          Manual_Set_D(subp.lambda_init, 0.0, 1.0, NE, 20);
      break;
    case '3':
      cprintf("Contraction Factor [0 ≦ real ≦ 1] → ");
      subp.contraction_mu =
        Manual_Set_D(subp.contraction_mu, 0.0, 1.0, EE, 20);
```

```
    break;
  case '4' :
    cprintf("Constant of Divergent Prevention [0 < real] --> ");
    rpemp.c =
      Manual_Set_D(rpemp.c, 0.0, 1E+307, NN, 20);
    break;
  case '5' :
    if (rpemp.stable == ON)
      rpemp.stable = OFF;
    else
      rpemp.stable = ON;
    break;
  case '6' :
    if (*init_file.name != NULL)
      break;
    cprintf("Initialization of Diagonal of P Matrix [0 < real] --> ");
    rpemp.pmatrix0 =
      Manual_Set_D(rpemp.pmatrix0, 0.0, 1E+307, NN, 20);
    break;
  case '7' :
    subp.activation++;
    if (subp.activation > 2)
      subp.activation = 0;
    break;
  case '8' :
    if (subp.activation == OFF)
      break;
    cprintf("Interval of Activation of P Matrix [1 ≦ integer ≦ %d] --> ",
                    useable_number);
    subp.activation_interval =
        Manual_Set_L(subp.activation_interval, 1,
                    useable_number, EE, 20);
    break;
  case '9' :
    if (subp.activation == OFF)
      break;
    cprintf("Power of Activation of P Matrix [1 ≦ real] --> ");
    subp.activation_power =
    Manual_Set_D(subp.activation_power, 1.0, 1E+307, NN, 20);
    break;
  case 1 :
    skip_number = 0;
    useable_number = abs_useable_number;
    break;
  case 2 :
    identification_number = useable_number;
    break;
  case 4 :
    *u_file.name = NULL;
    break;
  case 5 :
    *init_file.name = NULL;
    break;
  case 8 :
    cprintf("Setting up PLR ... Sure ? (y/n) --> ");
    c = toupper( getch() );
    if ( c == 'Y' )
      subp.contraction_mu = 1.0;
    break;
  case 9 :
    cprintf("Setting up ELS ... Sure ? (y/n) --> ");
    c = toupper( getch() );
    if ( c == 'Y' ) {
      subp.contraction_mu = 1.0;
      subp.lambda_init = 1.0;
    }
    break;
  case 10 :
    cprintf("Setting up RML ... Sure ? (y/n) --> ");
    c = toupper( getch() );
    if ( c == 'Y' ) {
      subp.contraction_mu = 0.0;
      subp.lambda_init = 1.0;
    }
    break;
  case 27 :
    return 0;
  }
 }
}

/* ---------------------------------------------------------------- */
int File_Reset(int file_kind, char *ptr1)
{
  extern int    DriveError;
  extern char   tmp_buff1[];
  int   i;
  char  *ptr2;
  FunctionKey(EMPTY);
  cprintf("PATH = ");
  gets(ptr1);
  if (*ptr1 == '.')
    return FAULT;
  i = Getfile(ptr1, file_kind);
  if (DriveError == ON) {
    Drive_Message();
    return FAULT;
  }
  switch (i) {
    case NOTHING :
      sprintf(tmp_buff1,
        "Fitting File Dose Not Exist on The Specified PATH.");
      Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
    case FAULT:
      return FAULT;
  }
  return SUCCESS;
}

/* ---------------------------------------------------------------- */
void Crt_Reset(void)
{
  clrscr();
  FunctionKey(PARAMCHANGE);
  Title();
  gotoxy(51, 2);
  textcolor(YELLOW);
  cprintf("SELECT FUNCTION ::: OK --> ESC");
}

/* ---------------------------------------------------------------- */
void Useable_Renumber(void)
{
  extern long   abs_useable_number, useable_number, skip_number;
  extern BINARY_FILE    u_file, y_file;
  if (*u_file.name != NULL)
    abs_useable_number = (long)min(u_file.number, y_file.number);
  else
```

```
    abs_useable_number = y_file.number;
  if (abs_useable_number < skip_number)
    skip_number = 0;
  useable_number = abs_useable_number - skip_number;
}
```

AMX1FILE.C

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <math.h>
#include <dos.h>
#include <dir.h>
#include "armax1.h"

/* ---------------------------------------------------------- */
int FileChangeCall(void)
{
  extern char   out_filename[], pole_filename[], pl_filename[],
          tmp_buff1[];
  extern long   useable_number, identification_number,
          interval_number;
  extern BINARY_FILE    u_file;
  extern RPEM_SUB     subp;
  extern RPEM_PARAMETER rpemp;
  char  *string1[] = {"", "RPE", "PLR", "ELS", "RML"},
      *string2[] = {"    ", "& PMA", "& PMDA"};
  char  *ptr1, input_buff[256];
  int   i, j, k, method1, u_order;
  clrscr();
  FunctionKey(FILECHANGE);
  Title();
  gotoxy(51, 2);
  textcolor(YELLOW);
  cprintf("SELECT FUNCTION ::: OK --> ESC");
  method1 = Method(&subp);
  if (*u_file.name != NULL)
    u_order = rpemp.n;
  else
    u_order = 0;
  gotoxy(1, 4);
  textcolor(LIGHTCYAN);
  cprintf("  ┌──────────────────────────────────────────────────────────────┐ ");
  cprintf("  |   ARMAX(%d, %d, %d) Identification                           | ",
          rpemp.n, u_order, rpemp.n);
  cprintf("  |     Using Algorithm    = %s %s                    | ",
          string1[method1], string2[subp.activation]);
  cprintf("  |     Used Number of Data = %-10ld                      | ", identification_number);
  cprintf("  └──────────────────────────────────────────────────────────────┘ ")0;
  while (1) {
    window(1, 21, 80, 24);
    clrscr();
    window(1, 1, 80 ,24);
    textcolor(LIGHTCYAN);
    gotoxy(2, 10);
    cprintf("TEXT FILE OF RESULT");
    gotoxy(2, 11);
    textcolor(WHITE);
    ptr1 = String_Adjust_1(out_filename, 40);
    printf("(0)  Filename      --> %-40s", ptr1);
    gotoxy(2, 12);
    if (interval_number > identification_number ||
            interval_number > useable_number)
      textcolor(LIGHTRED);
    else
      textcolor(WHITE);
    cprintf("(1)  Output Interval --> % -10ld", interval_number);
    textcolor(LIGHTCYAN);
    gotoxy(2, 14);
    cprintf("BINARY FILE OF REAL POLE");
    gotoxy(2, 15);
    if (*pole_filename == NULL) {
      textcolor(LIGHTGREEN);
      cprintf("(2)  Filename      --> UNDEFINED");
    } else {
      textcolor(WHITE);
      ptr1 = String_Adjust_1(pole_filename, 40);
      printf("(2)  Filename      --> %-40s", ptr1);
    }
    textcolor(LIGHTCYAN);
    gotoxy(2, 17);
    cprintf("TEXT FILE OF P MATRIX AND GAIN VECTOR");
    gotoxy(2, 18);
    if (*pl_filename == NULL) {
      textcolor(LIGHTGREEN);
      cprintf("(3)  Filename      --> UNDEFINED");
    } else {
      textcolor(WHITE);
      ptr1 = String_Adjust_1(pl_filename, 40);
      printf("(3)  Filename      --> %-40s", ptr1);
    }
    gotoxy(2, 21);
    textcolor(YELLOW);
    j = Key_Judge();
    switch (j) {
      case '0' :
        cprintf("Output File Name --> ");
        Manual_Set_F(out_filename, WITH_EXTENTION,
                     102, MAXBOTTOM);
        break;
      case '1' :
        cprintf("Interval of Output [1 ≦integer ≦%ld] --> ",
                identification_number);
        interval_number =
          Manual_Set_L(interval_number, 1,
                identification_number, EE, 20);
        break;
      case '2' :
        cprintf("Output File Name [without Extention Symbol] --> ");
        Manual_Set_F(pole_filename, WITHOUT_EXTENTION,
                     102, MAXBOTTOM);
        break;
      case '3' :
        cprintf("P/L File Name --> ");
        Manual_Set_F(pl_filename, WITH_EXTENTION,
                     102, MAXBOTTOM);
        break;
      case 1 :
        *pole_filename = NULL;
        break;
      case 2 :
        *pl_filename = NULL;
        break;
```

```
        case 27 :
          return 0;
      }
    }
}
```

AMX1INPU.C

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "amax1.h"

/* ------------------------------------------------------------ */
double Manual_Set_D(double pool, double min, double max, int type,
                    unsigned char length)
{
  char   message_buff[80],  input_buff[128], *ptr1, *ptr2;
  double new_value;
  ptr1 = input_buff;
  ptr1 += 2;
  *input_buff = length;
  cgets(input_buff);
  if (*ptr1 == '¥0')
    return pool;
  new_value = atof(ptr1);
  if (*ptr1 >= '.' && *ptr1 <= '9')
    switch (type) {
      case NN :
        if (new_value > min && new_value < max)
          return new_value;
        break;
      case NE :
        if (new_value > min && new_value <= max)
          return new_value;
        break;
      case EN :
        if (new_value >= min && new_value < max)
          return new_value;
        break;
      case EE :
        if (new_value >= min && new_value <= max)
          return new_value;
        break;
    }
  ptr2 = String_Adjust_2(ptr1, 20);
  sprintf(message_buff, "[ %s ] is Illegal Value. Cannot Renew.", ptr2);
  Message(message_buff, CONTINUE, LIGHTRED, MAXBOTTOM);
  return pool;
}

/* ------------------------------------------------------------ */
int Manual_Set_I(int pool, int min, int max, int type,
                 unsigned char length)
{
  char   message_buff[80],  input_buff[128], *ptr1, *ptr2;
  int    new_value;
  ptr1 = input_buff;
  ptr1 += 2;
  *input_buff = length;
  cgets(input_buff);
  if (*ptr1 == '¥0')
    return pool;
  new_value = atoi(ptr1);
  if (*ptr1 >= '0' && *ptr1 <= '9')
    switch (type) {
      case NN :
        if (new_value > min && new_value < max)
          return new_value;
        break;
      case NE :
        if (new_value > min && new_value <= max)
          return new_value;
        break;
      case EN :
        if (new_value >= min && new_value < max)
          return new_value;
        break;
      case EE :
        if (new_value >= min && new_value <= max)
          return new_value;
        break;
    }
  ptr2 = String_Adjust_2(ptr1, 20);
  sprintf(message_buff, "[ %s ] is Illegal Value. Cannot Renew.", ptr2);
  Message(message_buff, CONTINUE, LIGHTRED, MAXBOTTOM);
  return pool;
}

/* ------------------------------------------------------------ */
long Manual_Set_L(long pool, long min, long max, int type,
                  unsigned char length)
{
  char   message_buff[80],  input_buff[128], *ptr1, *ptr2;
  long   new_value;
  ptr1 = input_buff;
  ptr1 += 2;
  *input_buff = length;
  cgets(input_buff);

  if (*ptr1 == '¥0')
    return pool;
  new_value = atol(ptr1);
  if (*ptr1 >= '0' && *ptr1 <= '9')
    switch (type) {
      case NN :
        if (new_value > min && new_value < max)
          return new_value;
        break;
      case NE :
        if (new_value > min && new_value <= max)
          return new_value;
        break;
      case EN :
        if (new_value >= min && new_value < max)
          return new_value;
        break;
      case EE :
        if (new_value >= min && new_value <= max)
          return new_value;
        break;
    }
  ptr2 = String_Adjust_2(ptr1, 20);
  sprintf(message_buff, "[ %s ] is Illegal Value. Cannot Renew.", ptr2);
  Message(message_buff, CONTINUE, LIGHTRED, MAXBOTTOM);
```

```
    return pool;
}

/* ------------------------------------------------------------------ */
int Manual_Set_F(char *filename, int extention, unsigned char length,
                                                int line)
{
    char  message_buff[80], input_buff[128], *ptr1, *ptr2;
    ptr1 = input_buff;
    ptr1 += 2;
    *input_buff = length;
    cgets(input_buff);
    if (*ptr1 == '¥0')
        return FAULT;
    if (Filename_Check(ptr1, 12, extention) == FAULT){
        ptr2 = String_Adjust_1(ptr1, 18);
        sprintf(message_buff,
                "[ %s ] Is Unsuitable Filename.", ptr2);
        Message(message_buff, CONTINUE, LIGHTRED, line);
        return FAULT;
    }
    strcpy(filename, ptr1);
    return SUCCESS;
}
```

AMX1FUNC.C

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "armax1.h"

/* ------------------------------------------------------------------*/
int Method(RPEM_SUB*subp)
{
    int    method;
    if (subp->contraction_mu == 1.0) {
        method = PLR;
        if (subp->lambda_init == 1.0)
            method = ELS;
    } else {
        if (subp->contraction_mu == 1.0 && subp->lambda_init == 1.0)
            method = RML;
        else
            method = RPEM;
    }
    return method;
}

/* ------------------------------------------------------------------*/
char *String_Adjust_1(char *ptr1, int max_length)
{
    int    i, length;
    if ((length = strlen(ptr1)) > max_length)
        for (i = 1; i <= (length - max_length); i++)
            ptr1++;
    return ptr1;
}

/* ------------------------------------------------------------------*/
char *String_Adjust_2(char *ptr1, int max_length)
{
    int    i, length;
    char   *ptr2;
    ptr2 = ptr1;
    if ((length = strlen(ptr1)) > max_length) {
        for (i = 1; i <= (length - max_length); i++)
            ptr2++;
        *ptr2 = NULL;
    }
    return ptr1;
}

/* ------------------------------------------------------------------*/
int Filename_Check(char *ptr, int maxlen, int extention)
{
    extern char tmp_buff1[];
    char      *ptr1;
    int       i, counter = 1;
    for (i = 0; i < 256; i++)
        tmp_buff1[i] = '¥0';
    strcpy(tmp_buff1, ptr);
    if ((ptr1 = strrchr(tmp_buff1, '¥¥')) == NULL)
        ptr1 = tmp_buff1;
    else
        ptr1++;
    if (*ptr1 == '¥0' || strlen(ptr1) >= maxlen)
        return FAULT;
    while (*ptr1 != '.' && *ptr1 != '¥0') {
        ptr1++;
        counter++;
    }
    if (*ptr1 == '.' && extention == WITHOUT_EXTENTION)
        return FAULT;
    if (counter > 9 || strlen(ptr1) > 4)
        return FAULT;
    return SUCCESS;
}

/* ------------------------------------------------------------------*/
void File_Close(void)
{
    extern FILE         *fin_y, *fin_u, *fin_init, *fout_result,
                        *fout_pole[], *fout_pl;
    extern char         pl_filename[], pole_filename[];
    extern INIT_FILE    init_file;
    extern BINARY_FILE  u_file;
    extern RPEM_PARAMETER rpemp;
    int    i;
    fclose(fin_y);
    fclose(fout_result);
    if (*pole_filename != NULL)
        for (i = 1; i <= rpemp.n; i++)
            fclose(fout_pole[i]);
    if (*pl_filename != NULL)
        fclose(fout_pl);
    if (*u_file.name != NULL)
        fclose(fin_y);
    if (*init_file.name != NULL)
        fclose(fin_init);
}

/* ------------------------------------------------------------------*/
void Udl_to_P(int order, double udl[][25], double p[][25])
{
```

```
    int   i, j, k, start_row;
    double u[25][25], d[25][25], l[25][25], dummy[25][25];
    for (i = 0; i <= 24; i++)
      for (j = 0; j <= 24; j++) {
        u[i][j]   = 0.0;
        d[i][j]   = 0.0;
        l[i][j]   = 0.0;
        dummy[i][j]  = 0.0;
        p[i][j]   = 0.0;
      }
    for (i = 1; i < order; i++) {
      start_row = i + 1;
      for (j = start_row; j <= order; j++) {
        u[i][j] = udl[i][j];
        l[j][i] = udl[i][j];
      }
    }
    for (i = 1; i <= order; i++) {
      u[i][i] = 1.0;
      l[i][i] = 1.0;
    }
    for (i = 1; i <= order; i++)
      d[i][i] = udl[i][i];
    for (i = 1; i <= order; i++)
      for (j = 1; j <= order; j++)
        for (k = 1; k <= order; k++)
          dummy[i][j] += u[i][k] * d[k][j];
    for (i = 1; i <= order; i++)
      for (j = 1; j <= order; j++)
        for (k = 1; k <= order; k++)
          p[i][j] += dummy[i][k] * l[k][j];
}
```


AMX1CALL.C

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <math.h>
#include <dos.h>
#include <time.h>
#include <pc98.h>
#include "armax1.h"

/* ---------------------------------------------------------- */
int Armax_Call(void)
{
  extern FILE       *fin_y, *fin_u, *fin_init, *fout_result,
                    *fout_pole[], *fout_pl;
  extern char       tmp_buff1[], pl_filename[], pole_filename[];
  extern long       skip_number, identification_number,
                    interval_number;
  extern int        AutoSystem, AutoStop;
  extern INIT_FILE     init_file;
  extern RPEM_PARAMETER rpemp;
  extern RPEM_SUB      subp;
  extern BINARY_FILE   u_file, y_file;
  struct  time   start_time;
  time_t       first_time, second_time, injury_time = 0L, inju1,
               inju2;
  char         answer,
               *string1[] ={"", "RPE", "PLR", "ELS", "RML"},
               *string2[] ={"    ", "& PMA", "& PMDA"};
  int          method1, u_order, j, nnn, force_break = OFF;
  unsigned     code;
  long         i;
  clrscr();
  if (AutoSystem == ON)
    FunctionKey(RPEMCALL2);
  else
    FunctionKey(RPEMCALL1);
  window(1, 1, 80, 24);
  Title();
  method1 = Method(&subp);
  if (*u_file.name != NULL)
    u_order = rpemp.n;
  else
    u_order = 0;
  gotoxy(1, 4);
  textcolor(LIGHTCYAN);
  if (AutoSystem == ON)
    cprintf("┌──────────────────────□ PROGRAM MODE □──────────────────────┐
");
  else
    cprintf("┌──────────────────────────────────────────────────────────┐");
  cprintf("│   ARMAX(%d, %d, %d) Identification    :   Using Algorithm = %s %s   │",
  rpemp.n, u_order, rpemp.n, string1[method1],string2[subp.activation]);
  cprintf("└──────────────────────────────────────────────────────────┘");
  if (Starting_Check() == FAULT){
    File_Close();
    return FAULT;
  }
  /* Parameter Initilization */
  if (*init_file.name != NULL) {
    if ( Parameter_Init() == FAULT) {
      File_Close();
      return FAULT;
    }
  } else
    Parameter_Std_Init();
  /* Data Skip */
  fseek(fin_y, (skip_number*y_file.byte), SEEK_SET);
  if (*u_file.name != NULL)
    fseek(fin_u, (skip_number*u_file.byte), SEEK_SET);
  gettime(&start_time);
  second_time = first_time = time(NULL);
  gotoxy(49, 2);
  textcolor(YELLOW);
  cprintf("Start of Calculation = %2d:%02d:%02d",
      start_time.ti_hour, start_time.ti_min, start_time.ti_sec);
  textcolor(WHITE);
  window(1, 10, 80, 24);

  /* Start of Identification */
  nnn = rpemp.n * 3;
  rpemp.lambda = subp.lambda_init;
  rpemp.K    = 0.0;
  for (i = 1; i <= identification_number; i++) {
    rpemp.y = Read_Data(fin_y, y_file.byte);
    if (*u_file.name != NULL)
      rpemp.u = Read_Data(fin_u, u_file.byte);
    else
      rpemp.u = 0.0;
    Rpem_Routine();
    if (*pole_filename != NULL)
      Output_Pole();
```

```
    if (i % interval_number == 0) {
      Output_Result(i);
      if (*pl_filename != NULL)
        Output_PL(i, subp.activation);
      second_time = time(NULL);
      Time_Infomation(i, first_time, second_time, injury_time);
      code = pc98key(5);
      if (code == 0x6200 || code == 0x6B00) {
        inju1 = time(NULL);
        gotoxy(18, 14);
        textattr(YELLOW | BLINK | REVERSE);
        cprintf(" Will You Stop This Calculation Really ? (Y/N) ¥r");
        answer = '¥0';
        while (answer != 'Y' && answer != 'N')
          answer = toupper(getch());
        if (answer == 'Y') {
          force_break = ON;
          if (code == 0x6B00)
            AutoStop = ON;
          break;
        }
        else {
          gotoxy(18, 14);
          textattr(WHITE);
          cprintf("                                ¥r¥n");
        }
        inju2 = time(NULL);
        injury_time += difftime(inju2, inju1);
      }
      if (code == 0x6A00)
        AutoStop = ON;
      pc98key(3);
    }
    if (subp.activation != OFF)
      if (i % subp.activation_interval == 0)
        switch (subp.activation) {
          case PMA:
            for (j = 1; j <= nnn; j++)
              rpemp.UDL[i][j] *= subp.activation_power;
            break;
          case PMDA:
            for (j = 1; j <= nnn; j++)
              rpemp.P[i][j] *= subp.activation_power;
            break;
        }
    rpemp.lambda = subp.lambda_mu * rpemp.lambda +
                   1.0 - subp.lambda_mu;
    rpemp.K      = subp.contraction_mu * rpemp.K +
                   1.0 - subp.contraction_mu;
  }
  /* Writing Infomation to Result File */
  Writing_Info(rpemp.n, u_order, string1[method1],
               string2[subp.activation], force_break);
  sprintf(tmp_buff1, "NORMAL TERMINATION.");
  Message(tmp_buff1, CONTINUE, LIGHTCYAN, MAXBOTTOM);
  window(1, 1, 80, 24);
  File_Close();
  return SUCCESS;
}

/* ---------------------------------------------------------- */
int Starting_Check(void)
{
  extern int    DriveError;

extern char   out_filename[], pole_filename[], pl_filename[],
              tmp_buff1[], tmp_buff2[];
extern long   identification_number, useable_number,
              interval_number;
extern FILE   *fin_y, *fin_u, *fin_init, *fout_result,
              *fout_pole[], *fout_pl;
extern RPEM_PARAMETER rpemp;
extern BINARY_FILE    u_file, y_file;
extern INIT_FILE      init_file;
char          *ptr1, c;
unsigned char ptr2, ptr3;
int           i, check_byte;
double        d1_dummy;

if (identification_number > useable_number) {
  sprintf(tmp_buff1,
      "Number to Identify Exceed Number of Useable Data.");
  Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
  return FAULT;
}
if (interval_number > identification_number) {
  sprintf(tmp_buff1,
      "Interval of Output Exceeds Number to Identify.");
  Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
  return FAULT;
}
if ((fout_result = fopen(out_filename, "wt")) == NULL) {
  if (DriveError == ON)
    Drive_Message();
  sprintf(tmp_buff1,
    "Check PATH of Output Filename(Result) or Disk Condition");
  Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
  return FAULT;
}
if (*pole_filename != NULL) {
  for (i = 1; i <= rpemp.n; i++) {
    strcpy(tmp_buff1, pole_filename);
    strcat(tmp_buff1, ".");
    itoa(i, tmp_buff2, 10);
    strcat(tmp_buff1, tmp_buff2);
    if ((fout_pole[i] = fopen(tmp_buff1, "wb")) == NULL) {
      if (DriveError == ON)
        Drive_Message();
      sprintf(tmp_buff1,
      "Check PATH of Output Filename(Pole) or Disk Condition");
      Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
      return FAULT;
    }
  }
}
if (*pl_filename != NULL)
  if ((fout_pl = fopen(pl_filename, "wt")) == NULL) {
    if (DriveError == ON)
      Drive_Message();
    sprintf(tmp_buff1,
    "Check PATH of Output File(P/L) or Disk Condition");
    Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
    return FAULT;
  }
if ((fin_y = fopen(y_file.name, "rb")) == NULL) {
  ptr1 = String_Adjust_1(y_file.name, 10);
  sprintf(tmp_buff1,
  "Cannot Open Time Series File [ %s ].  Check Your Machine.", ptr1);
  Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
```

```
    return FAULT;
  }
  check_byte = 0;
  if (y_file.byte == 8) {
    fseek(fin_y, 6, SEEK_CUR);
    fread(&ptr2, 1, 1, fin_y);
    fread(&ptr3, 1, 1, fin_y);
    if (ptr2 > 0xEFu && (ptr3 == 0x7Fu || ptr3 == 0xFFu))
      check_byte = 1;
    else {
      fseek(fin_y, 0, SEEK_SET);
      fread(&d1_dummy, 8, 1, fin_y);
      d1_dummy = fabs(d1_dummy);
      if (d1_dummy > 1E+10 || d1_dummy < 1E-10)
        check_byte = 2;
    }
    switch (check_byte) {
      case 2 :
        sprintf(tmp_buff1,
          "Now, Setting of Record Length Is 8 Byte.  But It Seems to Be Your Mistake.");
        Message(tmp_buff1, FREE, LIGHTMAGENTA, 22);
        sprintf(tmp_buff1,
          "Do you Want to Continue This Process ? [Y/N]");
        Message(tmp_buff1, FREE, YELLOW, MAXBOTTOM);
        c = toupper( getch() );
        if ( c == 'Y' )
          break;
        else
          return FAULT;
      case 1 :
        sprintf(tmp_buff1,
          "Now, Setting of Record Length Is 8 Byte.  But It is Your Mistake Clearly.");
        Message(tmp_buff1, FREE, LIGHTRED, 22);
        sprintf(tmp_buff1,
          "So, This Process Is Terminated by Force.");
        Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
        return FAULT;
    }
  }
  if (*u_file.name != NULL) {
    if (strcmp(u_file.name, y_file.name) == 0) {
      sprintf(tmp_buff1,
      "U-Filename = Y-Filename.  The Same Name Is Used.", ptr1);
      Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
      return FAULT;
    }
    if ((fin_u = fopen(u_file.name, "rb")) == NULL) {
      ptr1 = String_Adjust_1(u_file.name, 10);
      sprintf(tmp_buff1,
      "Cannot Open Time Series File [ %s ].  Check Your Machine.", ptr1);
      Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
      return FAULT;
    }
  }
  if (*init_file.name != NULL) {
    if ((fin_init = fopen(init_file.name, "rt")) == NULL) {
      if (DriveError == ON)
        Drive_Message();
      sprintf(tmp_buff1,
      "Check PATH of Output File (P/L) or Disk Condition");
      Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
      return FAULT;
    }
    if (init_file.order != rpemp.n) {
      sprintf(tmp_buff1,
        "There Is Discrepancy as to Identification  Order.");
      Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
      return FAULT;
    }
  }
  return SUCCESS;
}

/* ---------------------------------------------------------------- */
int Parameter_Init(void)
{
  extern char        tmp_buff1[], tmp_buff2[];
  extern FILE        *fin_init;
  extern RPEM_PARAMETER rpemp;
  int            i, j, check, loop;
  double         d1_dummy;

loop = rpemp.n * 3;
  check = 1;
  while (fscanf(fin_init, "%s", tmp_buff1) != EOF)
    if ( (check = strcmp(tmp_buff1, "PARAMETER"))== 0 )
      break;
  if (check != 0) {
    sprintf(tmp_buff1, "Iintial Parameter File Dose Not Obey Defined Format.");
    Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
    return FAULT;
  }
  for (i = 1; i <= loop; i++) {
    fscanf(fin_init, "%s %s", tmp_buff1, tmp_buff2);
    d1_dummy = atof(tmp_buff2);
    if (*tmp_buff2 < '-' || *tmp_buff2 > '9') {
      sprintf(tmp_buff1, "Illegal Value as Initial Parameter.");
      Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
      return FAULT;
    }
    rpemp.theta[i] = d1_dummy;
  }
  check = 1;
  while (fscanf(fin_init, "%s", tmp_buff1) != EOF)
    if ( (check = strcmp(tmp_buff1, "P_MATRIX_or_UDL_MATRIX"))== 0 )
      break;
  if (check != 0) {
    sprintf(tmp_buff1, "Iintial Parameter File Dose Not Obey Defined Format.");
    Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
    return FAULT;
  }
  for (i = 1; i <= loop; i++)
    for (j = 1; j <= loop; j++) {
      fscanf(fin_init, "%s %s", tmp_buff1, tmp_buff2);
      d1_dummy = atof(tmp_buff2);
      if (*tmp_buff2 < '-' || *tmp_buff2 > '9') {
        sprintf(tmp_buff1, "Illegal Value as Initial P Matrix.");
        Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
        return FAULT;
      }
      rpemp.UDL[i][j] = rpemp.P[i][j] = d1_dummy;
    }
  check = 1;
  while (fscanf(fin_init, "%s", tmp_buff1) != EOF)
    if ( (check = strcmp(tmp_buff1, "L_VECTOR"))== 0 )
      break;
  if (check != 0) {
    sprintf(tmp_buff1, "Iintial Parameter File Dose Not Obey Defined Format.");
```

```
    Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
    return FAULT;
  }
  for (i = 1; i <= loop; i++) {
    fscanf(fin_init, "%s %s", tmp_buff1, tmp_buff2);
    d1_dummy = atof(tmp_buff2);
    if (*tmp_buff2 < '-' || *tmp_buff2 > '9') {
      sprintf(tmp_buff1, "Illegal Value as Initial Kalman Gain.");
      Message(tmp_buff1, CONTINUE, LIGHTRED, MAXBOTTOM);
      return FAULT;
    }
    rpemp.L[i] = d1_dummy;
  }
  return SUCCESS;
}

/* ---------------------------------------------------------- */
void Parameter_Std_Init(void)
{
  extern RPEM_PARAMETER rpemp;
  int   i, j, loop;
  loop = rpemp.n * 3;
  for (i = 1; i <= loop; i++) {
    rpemp.theta[i] = 0.0;
    rpemp.L[i]   = 0.0;
    rpemp.phi[i]   = 0.0;
    rpemp.psi[i]   = 0.0;
    for (j = 1; j <= loop; j++) {
      rpemp.UDL[i][i] = 0.0;
      rpemp.P[i][i]   = 0.0;
    }
    rpemp.P[i][i]  = rpemp.pmatrix0;
    rpemp.UDL[i][i]  = rpemp.pmatrix0;
  }
}

/* ---------------------------------------------------------- */
void Time_Information(long number, time_t first_time,
                      time_t second_time, time_t injury_time)
{
  extern long   identification_number;
  extern char   tmp_buff1[];
  struct time   t1;
  int        i_hour, i_minute, i_second, previous_y;
  double     remain_number, passed_time, nesessary_time,
             remain_time, hour, minute;
  remain_number = (double)(identification_number - number);
  passed_time = difftime(second_time, first_time) - injury_time;
  nesessary_time = passed_time / (double)number;
  remain_time = remain_number * nesessary_time;
  remain_time = ceil(remain_time);
  hour = floor(remain_time / 3600.0);
  remain_time -= hour*3600.0;
  i_hour = (int)hour;
  minute = floor(remain_time / 60.0);
  remain_time -= minute*60.0;
  i_minute = (int)minute;
  i_second = (int)remain_time;
  gettime(&t1);
  i_second += t1.ti_sec;
  if (i_second >= 60) {
    i_second -= 60;
    i_minute++;
  }
  i_minute += t1.ti_min;
  if (i_minute >= 60) {
    i_minute -= 60;
    i_hour++;
  }
  i_hour += t1.ti_hour;
  if (i_hour >= 24)
    i_hour -= 24;
  previous_y = wherey();
  window(1, 1, 80, 24);
  sprintf(tmp_buff1,
    "The Current Time = %2d:%02d:%02d      Expected Time of Finish = %2d:%02d:%02d",
    t1.ti_hour, t1.ti_min, t1.ti_sec, i_hour, i_minute, i_second);
  Message(tmp_buff1, FREE, LIGHTGREEN, 8);
  window(1, 10, 80, 24);
  gotoxy(1, previous_y);
}

/* ---------------------------------------------------------- */
double Read_Data(FILE *fin, int byte)
{
  int   i_dummy;
  double value;
  switch (byte) {
    case 2 :
      fread(&i_dummy, 2, 1, fin);
      value = (double)i_dummy;
      break;
    case 8 :
      fread(&value, 8, 1, fin);
      break;
  }
  return value;
}
```

AMX1RPEM.C

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <math.h>
#include "armax1.h"

/* ---------------------------------------------------------- */
void Rpem_Routine(void)
{
  extern RPEM_PARAMETER rpemp;
  extern RPEM_SUB     subp;
  int   i, j, k, nnn, ex_part_s, ma_part_s;
  double d1_dummy, amy, ma_part[28], y1_hat, u1_hat, e1_hat, c_i;
  nnn = rpemp.n * 3;
  ex_part_s = rpemp.n + 1;
  ma_part_s = 2*rpemp.n + 1;
  /* Calculation of Prediction Error */
  rpemp.prediction_error = rpemp.y;
  for (i = 1; i <= nnn; i++)
    rpemp.prediction_error -= rpemp.phi[i] * rpemp.theta[i];

  /* Stablility Check and Decision of AMY */
  amy = 1.0;
  if (rpemp.stable == ON) {
```

```
    while (1) {
      for (i = 0; i < rpemp.n; i++) {
        j = ma_part_s + i;
        ma_part[i+2] = rpemp.theta[j] + rpemp.L[j]*
                       rpemp.prediction_error*amy;
      }
      ma_part[1] = 1.0;
      if (Stable_Check(ma_part, rpemp.n) == SUCCESS)
        break;
      amy /= 2.0;
    }
  }

  /* Renewal of Parameter */
  for (i = 1; i <= nnn; i++)
    rpemp.theta[i] += rpemp.L[i] * rpemp.prediction_error * amy;

  /* Calculation of Current Residual */
  rpemp.residual = rpemp.y;
  for (i = 1; i <= nnn; i++)
    rpemp.residual -= rpemp.phi[i] * rpemp.theta[i];

  /* Whitning of Residual */
  y1_hat = rpemp.y;
  u1_hat = rpemp.u;
  e1_hat = rpemp.residual;
  for (i = 0; i < rpemp.n; i++) {
    d1_dummy = (double)i + 1.0;
    c_i = rpemp.theta[ma_part_s + i] * pow(rpemp.K, d1_dummy);
    y1_hat += c_i * rpemp.psi[1 + i];
    u1_hat -= c_i * rpemp.psi[ex_part_s + i];
    e1_hat -= c_i * rpemp.psi[ma_part_s + i];
  }

  /* Update of Time Series Containment Vector phi[] psi[] */
  for (i = 2; i <= rpemp.n; i++)
    for (j = 1; j <= 3; j++) {
      k = j*rpemp.n + 2 - i;
      rpemp.phi[k] = rpemp.phi[k-1];
      rpemp.psi[k] = rpemp.psi[k-1];
    }
  rpemp.phi[1]       = -rpemp.y;
  rpemp.phi[ex_part_s] = rpemp.u;
  rpemp.phi[ma_part_s] = rpemp.residual;
  rpemp.psi[1]       = -y1_hat;
  rpemp.psi[ex_part_s] = u1_hat;
  rpemp.psi[ma_part_s] = e1_hat;

  /* Update of P Matrix or UDL Matrix and Gain Vector L*/
  switch (subp.activation) {
    case OFF:
    case PMA:
      UDL_Factorization(&rpemp);
      break;
    case PMDA:
      P_Matrix_Calc(&rpemp);
      break;
  }
}

/* ---------------------------------------------------------- */
int Stable_Check(double *work, int n)
{
  int   i, k, n1, nk1, nk;
  double al;
  n1 = n + 1;
  for (i = 1; i <= n1; i++) {
    work[n1+i] = 0.0;
  }
  for (k = 0; k < n; k++) {
    nk1 = n1 - k;
    for (i = 1; i <= nk1; i++)
      work[n1+i] = work[nk1-i+1];
    if (work[n1+nk1] == 0.0)
      return FAULT;
    al = work[nk1] / work[n1+nk1];
    if (fabs(al) >= 1.0)
      return FAULT;
    nk = n - k;
    for (i = 1; i <= nk; i++)
      work[i] -= al*work[n1+i];
  }
  return SUCCESS;

}

/* ---------------------------------------------------------- */
int UDL_Factorization(RPEM_PARAMETER *rpemp)
{
  int   i, j, k, l, nnn;
  double f[25], g[25], previous_beta, beta_inverse, beta, previous_g,
      previous_udl, d1_dummy;
  nnn = rpemp->n * 3;
  for (i = 2; i <= nnn; i++) {
    j = nnn + 2 - i;
    f[i] = rpemp->psi[j];
    k = j - 1;
    for (l = 1; l <= k; l++)
      f[i] += rpemp->UDL[l][j] * rpemp->psi[l];
    g[i] = rpemp->UDL[i][i] * f[i];
  }
  g[1] = rpemp->UDL[1][1] * rpemp->psi[1];
  f[1] = rpemp->psi[1];
  previous_beta = rpemp->lambda + f[1]*g[1];
  beta_inverse = 0.0;
  if (previous_beta > 0.0)
    beta_inverse = 1.0 / previous_beta;
  if (g[1] != 0.0)
    rpemp->UDL[1][1] = beta_inverse * rpemp->UDL[1][1];
  for (i = 2; i <= nnn; i++) {
    beta = previous_beta;
    previous_g = g[i];
    previous_beta += previous_g * f[i];
    if (previous_beta == 0.0)
      continue;
    d1_dummy = -f[i] * beta_inverse;
    j = i - 1;
    for (k = 1; k <= j; k++) {
      previous_udl = rpemp->UDL[k][i];
      rpemp->UDL[k][i] += d1_dummy * g[k];
      g[k] += previous_udl * previous_g;
    }
    beta_inverse = 1.0 / previous_beta;
    rpemp->UDL[i][i] *= beta * beta_inverse / rpemp->lambda;
    rpemp->UDL[i][i] = (double)min(rpemp->UDL[i][i], rpemp->c);
  }
  for (i = 1; i <= nnn; i++)
    rpemp->L[i] = g[i] / previous_beta;
```

```
    return SUCCESS;
}

/* ---------------------------------------------------------- */
int P_Matrix_Calc(RPEM_PARAMETER *rpemp)
{
    int    i, j, nnn;
    double p_psi[25], psi_p[25], p_psi_psi_p[25][25], bunbo;
    nnn = rpemp->n * 3;
    for (i = 1; i <= nnn; i++) {
        rpemp->L[i] = 0.0;
        p_psi[i] = 0.0;
        psi_p[i] = 0.0;
    }

    for (i = 1; i <= nnn; i++)
        for (j = 1; j <= nnn; j++)
            p_psi[i] += rpemp->P[i][j] * rpemp->psi[j];
    for (i = 1; i <= nnn; i++)
        for (j = 1; j <= nnn; j++)
            psi_p[i] += rpemp->psi[j] * rpemp->P[j][i];
    for (i = 1; i <= nnn; i++)
        for (j = 1; j <= nnn; j++)
            p_psi_psi_p[i][j] = p_psi[i] * psi_p[j];
    bunbo = rpemp->lambda;
    for (i = 1; i <= nnn; i++)
        bunbo += rpemp->psi[i] * p_psi[i];
    for (i = 1; i <= nnn; i++)
        for (j = 1; j <= nnn; j++)
            p_psi_psi_p[i][j] /= bunbo;
    for (i = 1; i <= nnn; i++)
        for (j = 1; j <= nnn; j++)
            rpemp->P[i][j] -= p_psi_psi_p[i][j];
    for (i = 1; i <= nnn; i++)
        for (j = 1; j <= nnn; j++)
            rpemp->P[i][j] /= rpemp->lambda;
    for (i = 1; i <= nnn; i++)
        for (j = 1; j <= nnn; j++)
            rpemp->L[i] += rpemp->P[i][j] * rpemp->psi[j];
    return SUCCESS;
}
```

AMX1OUT.C

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include "armax1.h"

/* ---------------------------------------------------------- */
void Output_Result(long loop)
{
    extern RPEM_PARAMETER rpemp;
    extern BINARY_FILE    u_file;
    extern FILE           *fout_result;
    COMPLEX               pole[10];
    int                   i, start_point, converge_check;
    char                  *string1[] = {"FATALERROR","CONVERGE",
                                        "NOCONVERGE"};
    converge_check = Pole_Calculation(rpemp.n, rpemp.theta, pole);
    cprintf("%-7ld POLE:%s¥r¥n", loop, string1[converge_check]);
    fprintf(fout_result, "%-7ld POLE:%s¥n", loop, string1[converge_check]);

    /* Writing of AR Parameters */
    Parameter_Writing(rpemp.n, 1, rpemp.theta);
    /* Writing of Exgeneous Parameters */
    if (*u_file.name != NULL) {
        start_point = rpemp.n + 1;
        Parameter_Writing(rpemp.n, start_point, rpemp.theta);
    }
    /* Writing of MA Parameters */
    start_point = rpemp.n*2 + 1;
    Parameter_Writing(rpemp.n, start_point, rpemp.theta);

    /* Writing of Pole */
    for (i = 0; i < rpemp.n; i++) {
        cprintf("    % 15.6e% 15.6e¥r¥n",
                        pole[i].real, pole[i].image);
        fprintf(fout_result, "    % 15.6e% 15.6e¥n",
                        pole[i].real, pole[i].image);
    }
    cprintf("¥r¥n");
    fprintf(fout_result, "¥n¥n");
}

/* ---------------------------------------------------------- */
void Parameter_Writing(int number, int start_point, double theta[])
{
    extern FILE   *fout_result;
    int           i, j;
    if (number > 4) {
        for (i = 0; i <= 3; i++) {
            j = i + start_point;
            cprintf("% 19.6e", theta[j]);
            fprintf(fout_result, "% 19.6e", theta[j]);
        }
        cprintf("¥r¥n");
        fprintf(fout_result, "¥n");
        for (i = 4; i < number; i++) {
            j = i + start_point;
            cprintf("% 19.6e", theta[j]);
            fprintf(fout_result, "% 19.6e", theta[j]);
        }
    } else
        for (i = 0; i < number; i++) {
            j = i + start_point;
            cprintf("% 19.6e", theta[j]);
            fprintf(fout_result, "% 19.6e", theta[j]);
        }
    cprintf("¥r¥n");
    fprintf(fout_result, "¥n");
}

/* ---------------------------------------------------------- */
void Writing_Info(int order, int u_order, char *method1, char *method2,
                  int force_break)
{
    extern FILE   *fout_result;
    fprintf(fout_result, "¥n¥n");
    fprintf(fout_result, "                                        ¥n");
    fprintf(fout_result, "  Condition of Identification  ¥n");
    fprintf(fout_result, "                                        ¥n¥n");
    fprintf(fout_result, "ARMAX(%d, %d, %d) Identification¥n",
                    order, u_order, order);
    fprintf(fout_result, "Used Algorithm = %s %s¥n", method1, method2);
    if (force_break == ON)
        fprintf(fout_result, "This Calculation Was Stopped by Force¥n");
```

```
  fprintf(fout_result, "¥n¥n");
  Setupfile_Writing(fout_result);
}

/* ------------------------------------------------------------ */
void Setupfile_Writing(FILE *fil)
{
  extern char         *read_param[];
  extern RPEM_PARAMETER rpemp;
  extern RPEM_SUB      subp;
  extern BINARY_FILE    u_file, y_file;
  extern INIT_FILE      init_file;
  extern char         out_filename[], pole_filename[], pl_filename[];
  extern long         identification_number, skip_number,
                      interval_number;
  char     *string1[] = {"OFF", "ON"},
           *string2[] = {"OFF", "PMA", "PMDA"};
  fprintf(fil, "%s %d¥n", read_param[0], rpemp.n);
  fprintf(fil, "%s %d¥n", read_param[1], y_file.byte);
  if (*u_file.name != NULL)
    fprintf(fil, "%s %s¥n", read_param[2], u_file.name);
  else
    fprintf(fil, "%s NULL¥n", read_param[2]);
  fprintf(fil, "%s %s¥n", read_param[3], y_file.name);
  if (*init_file.name != NULL)
    fprintf(fil, "%s %s¥n", read_param[4], init_file.name);
  else
    fprintf(fil, "%s NULL¥n", read_param[4]);
  fprintf(fil, "%s %ld¥n", read_param[5],  skip_number);
  fprintf(fil, "%s %ld¥n", read_param[6],  identification_number);
  fprintf(fil, "%s %f¥n",  read_param[7],  subp.lambda_mu);
  fprintf(fil, "%s %f¥n",  read_param[8],  subp.lambda_init);
  fprintf(fil, "%s %f¥n",  read_param[9],  subp.contraction_mu);
  fprintf(fil, "%s %f¥n",  read_param[10], rpemp.c);
  fprintf(fil, "%s %s¥n",  read_param[11], string1[rpemp.stable]);
  fprintf(fil, "%s %f¥n",  read_param[12], rpemp.pmatrix0);
  fprintf(fil, "%s %s¥n",  read_param[13], string2[subp.activation]);
  fprintf(fil, "%s %ld¥n", read_param[14], subp.activation_interval);
  fprintf(fil, "%s %f¥n",  read_param[15], subp.activation_power);
  fprintf(fil, "%s %s¥n",  read_param[16], out_filename);
  fprintf(fil, "%s %ld¥n", read_param[17], interval_number);
  if (*pole_filename != NULL)
    fprintf(fil, "%s %s¥n", read_param[18], pole_filename);
  else
    fprintf(fil, "%s NULL¥n", read_param[18]);
  if (*pl_filename != NULL)
    fprintf(fil, "%s %s¥n", read_param[19], pl_filename);
  else
    fprintf(fil, "%s NULL¥n", read_param[19]);
}

/* ------------------------------------------------------------ */
void Output_Pole(void)
{
  extern FILE         *fout_pole[];
  extern RPEM_PARAMETER rpemp;
  COMPLEX             pole[10];
  int                 i, j;
  Pole_Calculation(rpemp.n, rpemp.theta, pole);
  for (i = 0; i < rpemp.n; i++) {
    j = i + 1;
    fwrite(&pole[i].real, 8, 1, fout_pole[j]);
  }
}
```



```
/* ------------------------------------------------------------ */
void Output_PL(long loop, int activation_method)
{
  extern RPEM_PARAMETER rpemp;
  extern FILE         *fout_pl;
  int                 i, j, nnn;
  nnn = rpemp.n * 3;
  fprintf(fout_pl,
      "Value of Y_Data at %ld Data        = %13.6e¥n",
                                  loop, rpemp.y);
  fprintf(fout_pl,
      "Value of Prediction Error at %ld Data = %13.6e¥n",
                  loop, rpemp.prediction_error);
  fprintf(fout_pl,
      "Value of The Residual at %ld Data    = %13.6e¥n¥n",
                      loop, rpemp.residual);
  fprintf(fout_pl, "Elements of P Matrix at %ld Data¥n", loop);
  switch (activation_method) {
    case OFF:
    case PMA:
      Udl_to_P(nnn, rpemp.UDL, rpemp.P);
      for (i = 1; i <= nnn; i++)
        for (j = 1; j <= nnn; j++)
          fprintf(fout_pl, "P[%d][%d] = % 13.6e    UDL[%d][%d] = % 13.6e¥n",
            i, j, rpemp.P[i][j],  i, j, rpemp.UDL[i][j]);
      break;
    case PMDA:
      for (i = 1; i <= nnn; i++)
        for (j = 1; j <= nnn; j++)
          fprintf(fout_pl, "P[%d][%d] = % 13.6e¥n",
                          i, j, rpemp.P[i][j]);
      break;
  }
  fprintf(fout_pl, "Elements of L Vector at %ld Data¥n", loop);
  for (i = 1; i <= nnn; i++)
    fprintf(fout_pl, "L[%d]    = % 13.6e¥n", i, rpemp.L[i]);
  fprintf(fout_pl, "¥n¥n");
}
```

AMX1POLE.C

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "armax1.h"
#define EPS    1.0e-6
#define MAXLOOP 30000

/* ------------------------------------------------------------ */
int Pole_Calculation(int order, double theta[], COMPLEX pole[])
{
  int    i, converge_check;
  double coefficient[10];
  coefficient[0] = 1.0;
  for (i = 1; i <= order; i++)
    coefficient[i] = theta[i];
  for (i = 0; i <= order; i++)
    pole[i].image = 0.0;
  converge_check = Bairstow(coefficient, order, pole);
  if (converge_check == MAXLOOP)
    return NOCONVERGE;
```

```
    if (converge_check == FAULT)
      return NOTHING;
    return SUCCESS;
}

/* ------------------------------------------------------------ */
int Bairstow(double ta[], int n, COMPLEX x[])
{
    double a[10], b[10], c[10];
    double p0, q0, p, q, D, delta_p, delta_q;
    double tmp[3];
    int    i, j, k, l, max_loop = 0, count = 0;
    for (i = 1; i <= n; i++)
      a[i] = ta[i] / ta[0];
    a[0] = 1.0;
    tmp[0] = 1.0;
    while (n > 2) {
      p = p0 = 1.0;
      q = q0 = 1.0;
      for (l = 1;l < MAXLOOP; l++) {
        b[0] = 1.0;
        c[0] = 1.0;
        b[1] = a[1] - p;
        c[1] = b[1] - p;
        for(k = 2; k <= n; k++) {
          b[k] = a[k] - p*b[k-1] - q*b[k-2];
          c[k] = b[k] - p*c[k-1] - q*c[k-2];
        }
        D = c[n-2]*c[n-2] - c[n-3]*(c[n-1] - b[n-1]);
        if (D == 0.0)
          return FAULT;
        delta_p = ( b[n-1]*c[n-2] - b[n]*c[n-3] ) / D;
        delta_q = ( b[n]*c[n-2] - b[n-1]*(c[n-1] - b[n-1]) ) / D;
        p = p0 + delta_p;
        q = q0 + delta_q;
        if( (fabs(delta_p) < EPS) && ( fabs(delta_q) < EPS) )
          break ;
        p0 = p;
        q0 = q;
      }
      if (l > max_loop)
        max_loop = l;
      tmp[1] = p ;  tmp[2] = q ;
      Equation_2(tmp, &x[count]) ;
      n -= 2;
      count += 2;
      for(i = 0; i <= n; i++)
        a[i] = b[i];
    }
    if (n == 2) {
      tmp[1] = a[1];
      tmp[2] = a[2];
      Equation_2(tmp, &x[count]);
    }
    if ( n & 1 ) {
      x[count].real = -a[1];
      x[count].image = 0.0;
    }
    return max_loop;
}

/* ------------------------------------------------------------ */
void Equation_2(double a[], COMPLEX x[])
{
    double tmp, D;
    D = a[1]*a[1] - 4*a[0]*a[2] ;
    if(D >= 0.0) {
      tmp = -a[1] + sqrt(D);
      x[0].real = tmp / (2 * a[0]);
      tmp = -a[1] - sqrt(D);
      x[1].real = tmp / (2 * a[0]);
    }
    else {
      x[0].real = -a[1] / (2 * a[0]) ;
      x[0].image = sqrt(-D) / (2 * a[0]) ;
      x[1].real = x[0].real ;
      x[1].image = -x[0].image ;
    }
}
```

AMX1OPTI.C

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include "armax1.h"

/* ------------------------------------------------------------ */
int Remake_File(void)
{
    FILE   *fremake;
    char   *ptr1, buff[128], message_buff[256];
    FunctionKey(EMPTY);
    gotoxy(1, 4);
    printf("%c[33mNew Set Up File Name : %c[37m", 0x1B,0x1B);
    if (Manual_Set_F(buff, WITH_EXTENTION, 102, 7) == FAULT)
      return FAULT;
    if ( (fremake = fopen(buff, "wt")) == NULL ) {
      ptr1 = String_Adjust_1(buff, 18);
      sprintf(message_buff,  "New Set Up File [ %s ] Cannot Open", ptr1);
      Message(message_buff,  CONTINUE, LIGHTRED, 5);
      return FAULT;
    }
    Setupfile_Writing(fremake);
    fclose(fremake);
    return SUCCESS;
}

/* ------------------------------------------------------------ */
void Help(void)
{
    extern char tmp_buff1[];
    static char *help_message[] ={
      "現在のバージョンは１．　０です．",
      "",
      "",
      "使用方法における疑問点，またはバグリポートについては，下記に連絡して下さい．",
      "",
      "",
      "          〒５７２　寝屋川市池田中町１７－８",
      "                    摂南大学　工学部　電気工学科　　山田　澄",
      "                    電話番号 (0720)-26-5101 ext. 307",
      "",
      "",
      "",
      "プログラムモードを利用する場合は",
      "          ＡＲＭＡＸ１　［制御ファイル名］［リポートファイル名］",
```

```
    "と入力して下さい. ";
  int i;
  FunctionKey(EMPTY);
  gotoxy(1, 2);
  printf("                              ¥n");
  printf("                              ¥n");
  gotoxy(1, 4);
  for (i = 0; i <= 12; i++)
    printf("%s¥n", help_message[i]);
  sprintf(tmp_buff1,  "END OF HELP");
  Message(tmp_buff1,  CONTINUE, LIGHTCYAN, MAXBOTTOM);
}
```

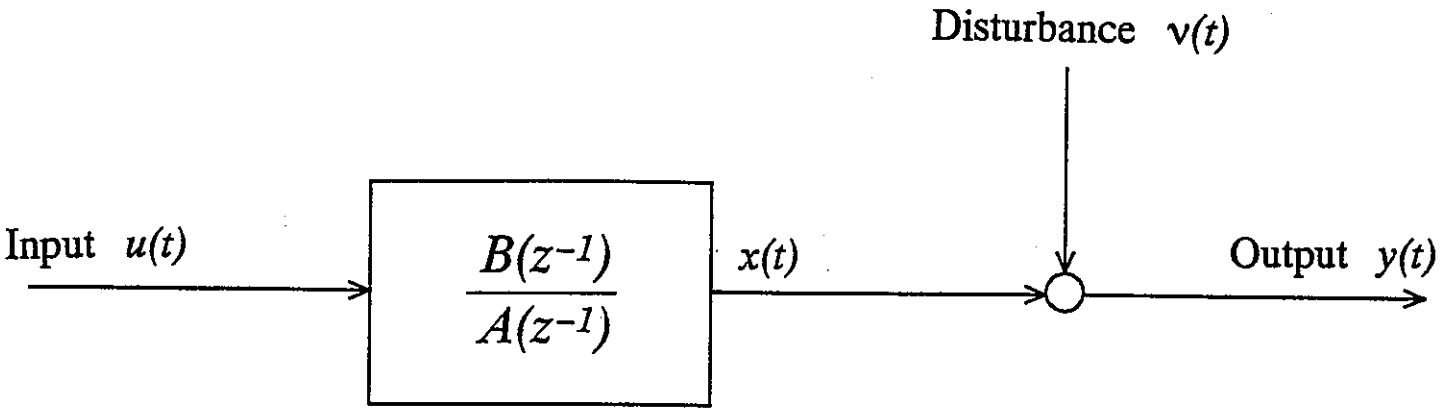


Fig. 2.1 Discrete linear system.

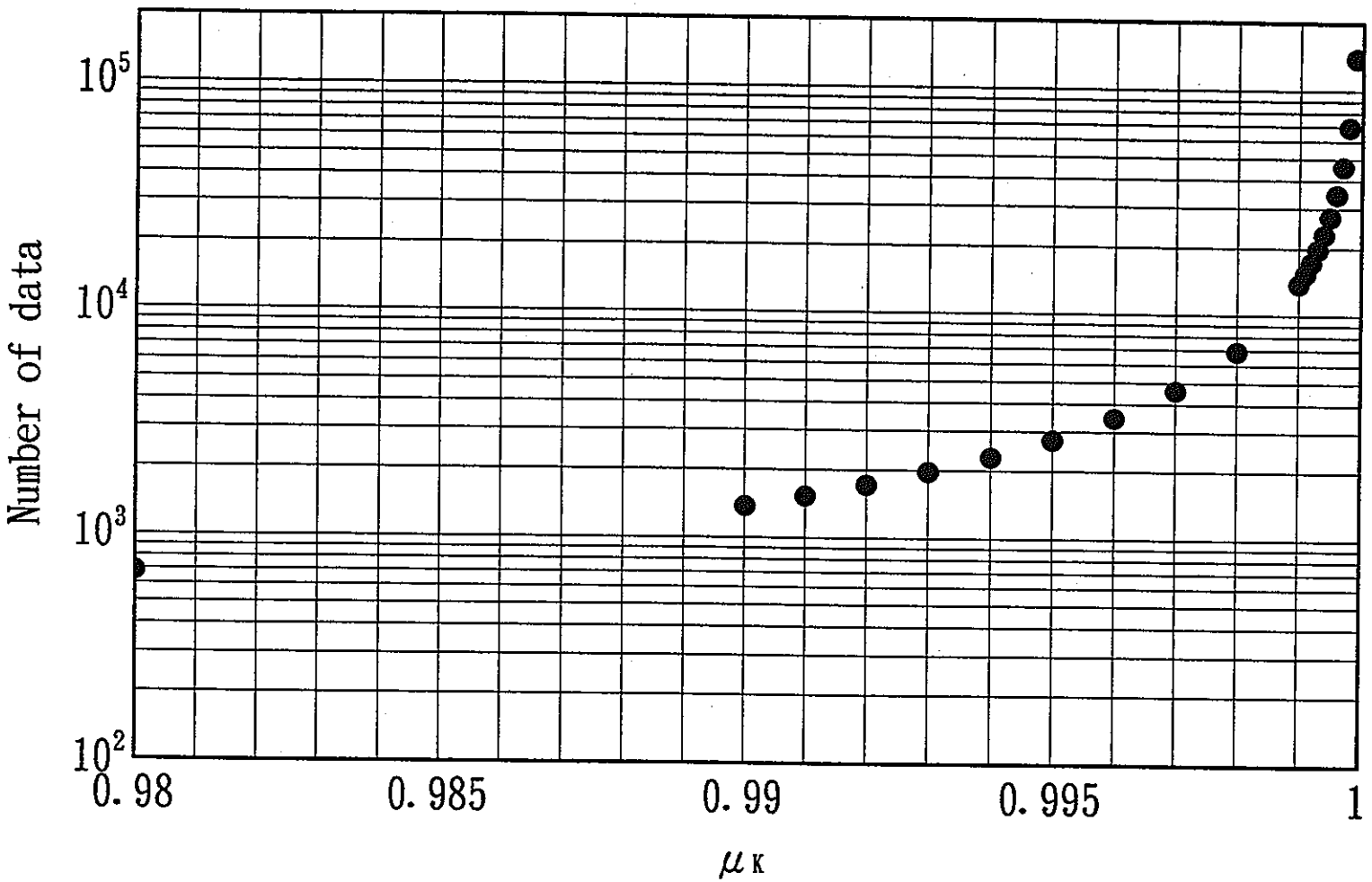
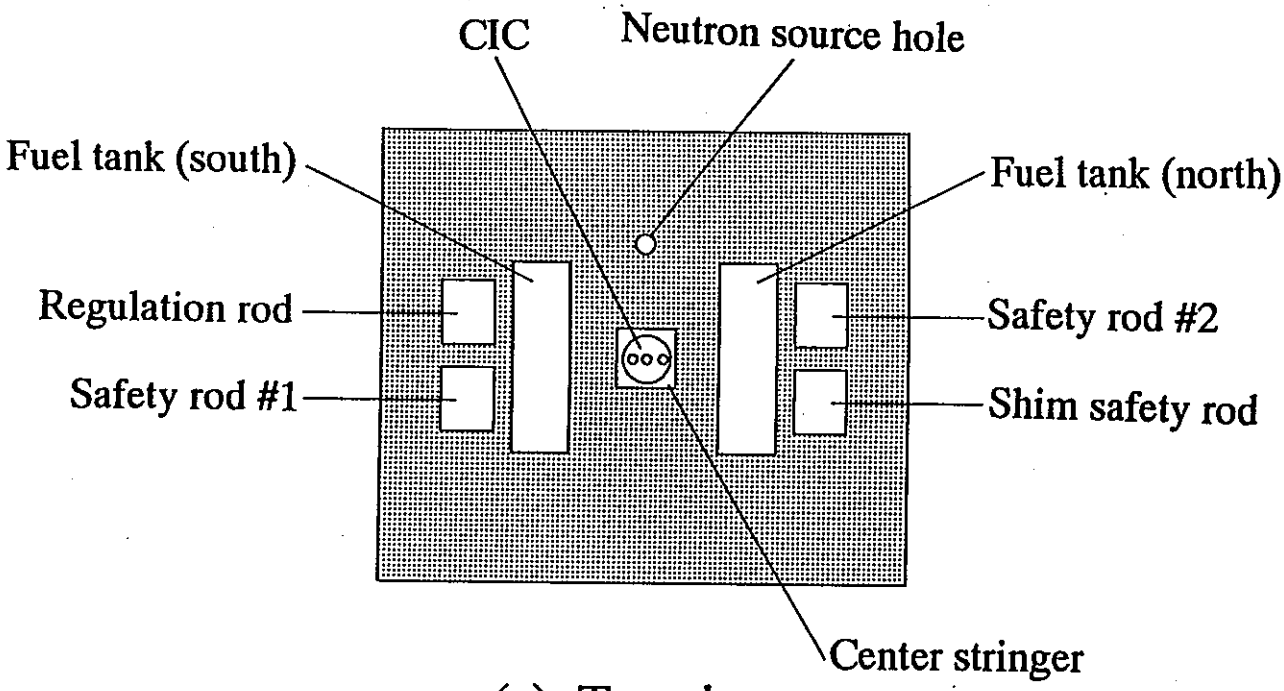
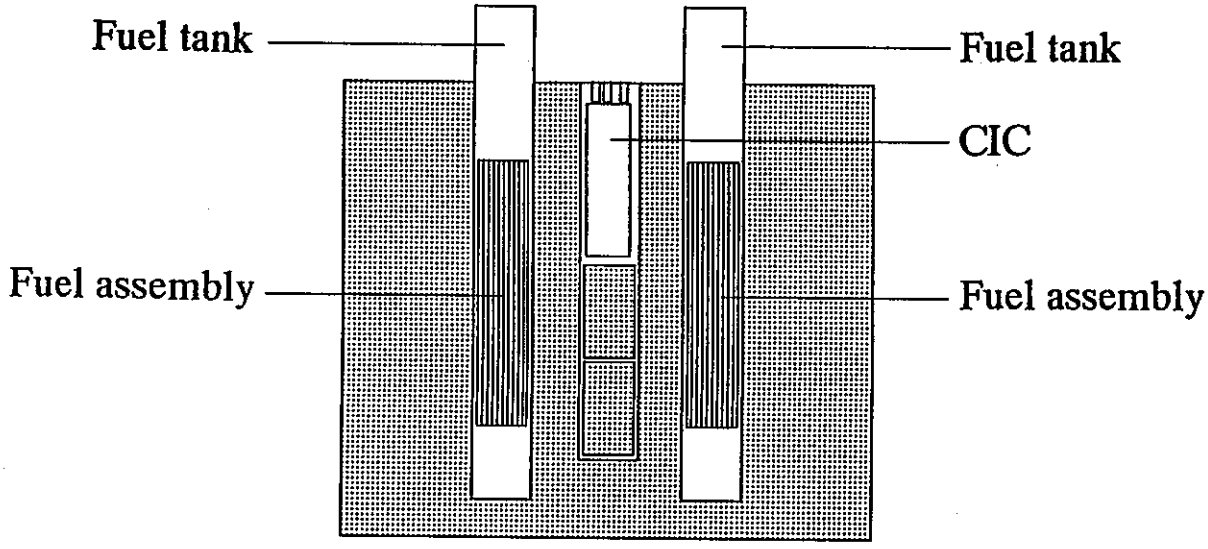


Fig. 2.2 Number of required data for $K_t > 0.999999$. (K_t : contraction factor.)

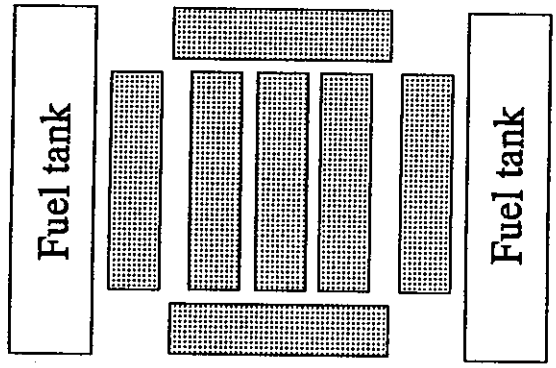


(a) Top view.

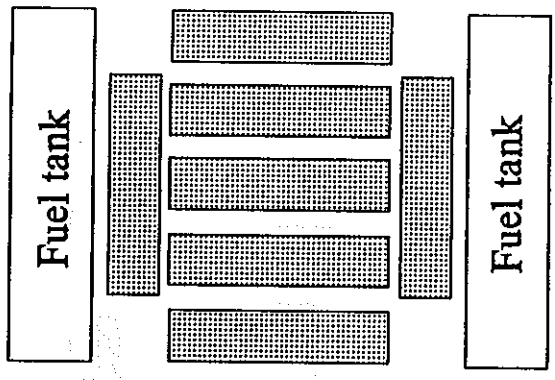


(b) Cross section view.

Fig. 3.1 Position of CIC in the reactor.



(a) Lower part.



(b) Upper part.

Fig. 3.2 Arrangement of the carbon block for compensating the reactivity.

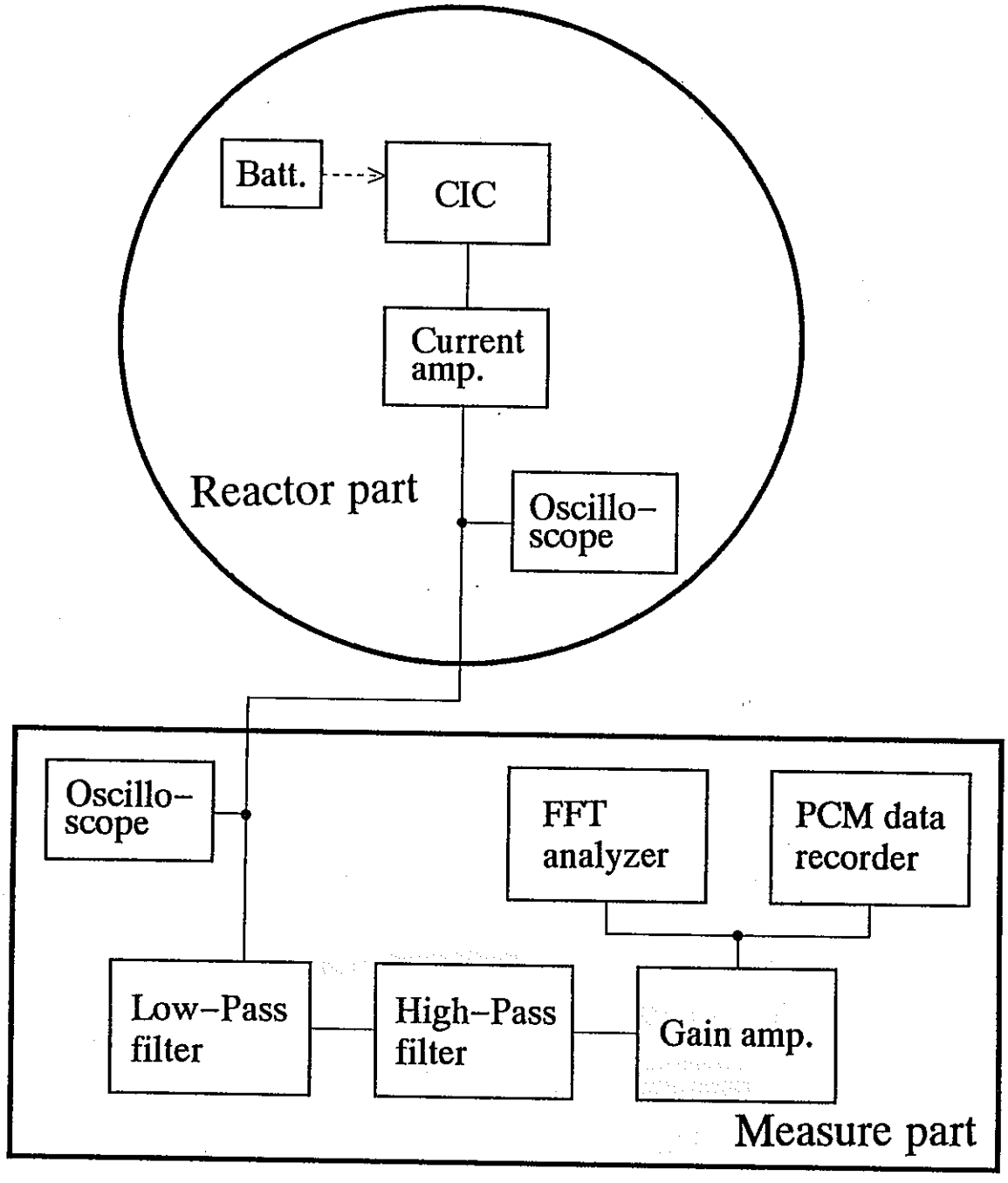
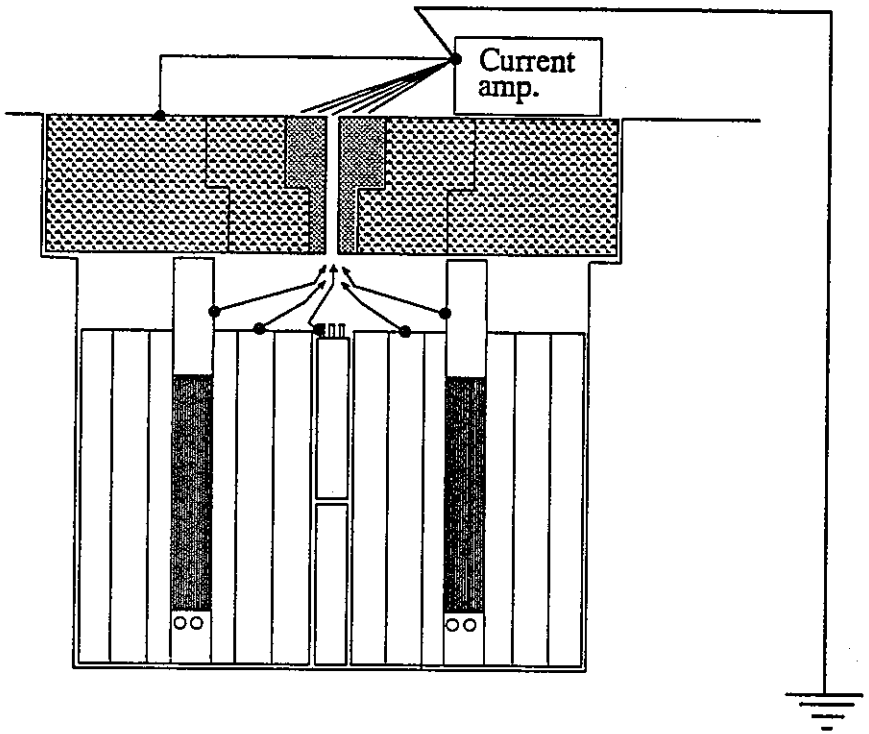


Fig. 3.3 Block diagram of data acquisition system.



Grand points.

Current amplifier.
Concrete cover at top of reactor.
CIC.
Carbon block.
Fuel tank.

Fig. 3.4 Earth points for neutron detecting system.

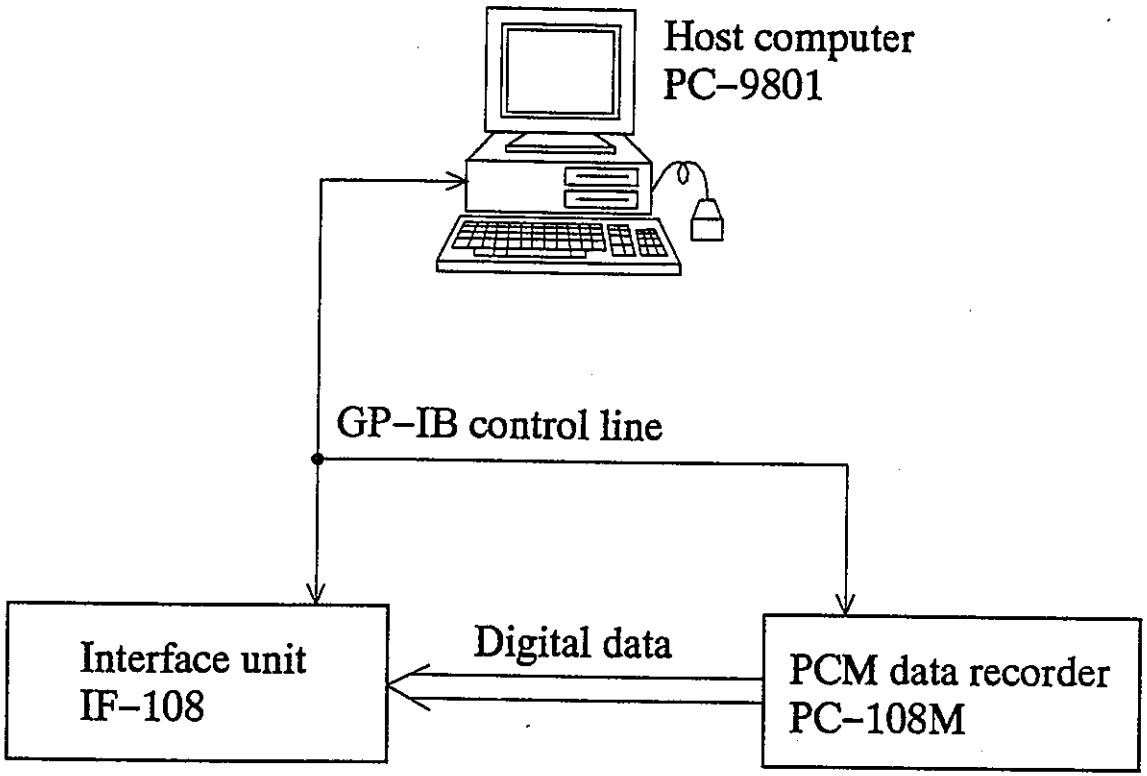


Fig. 3.5 Digital data retreading system from PCM data recorder.

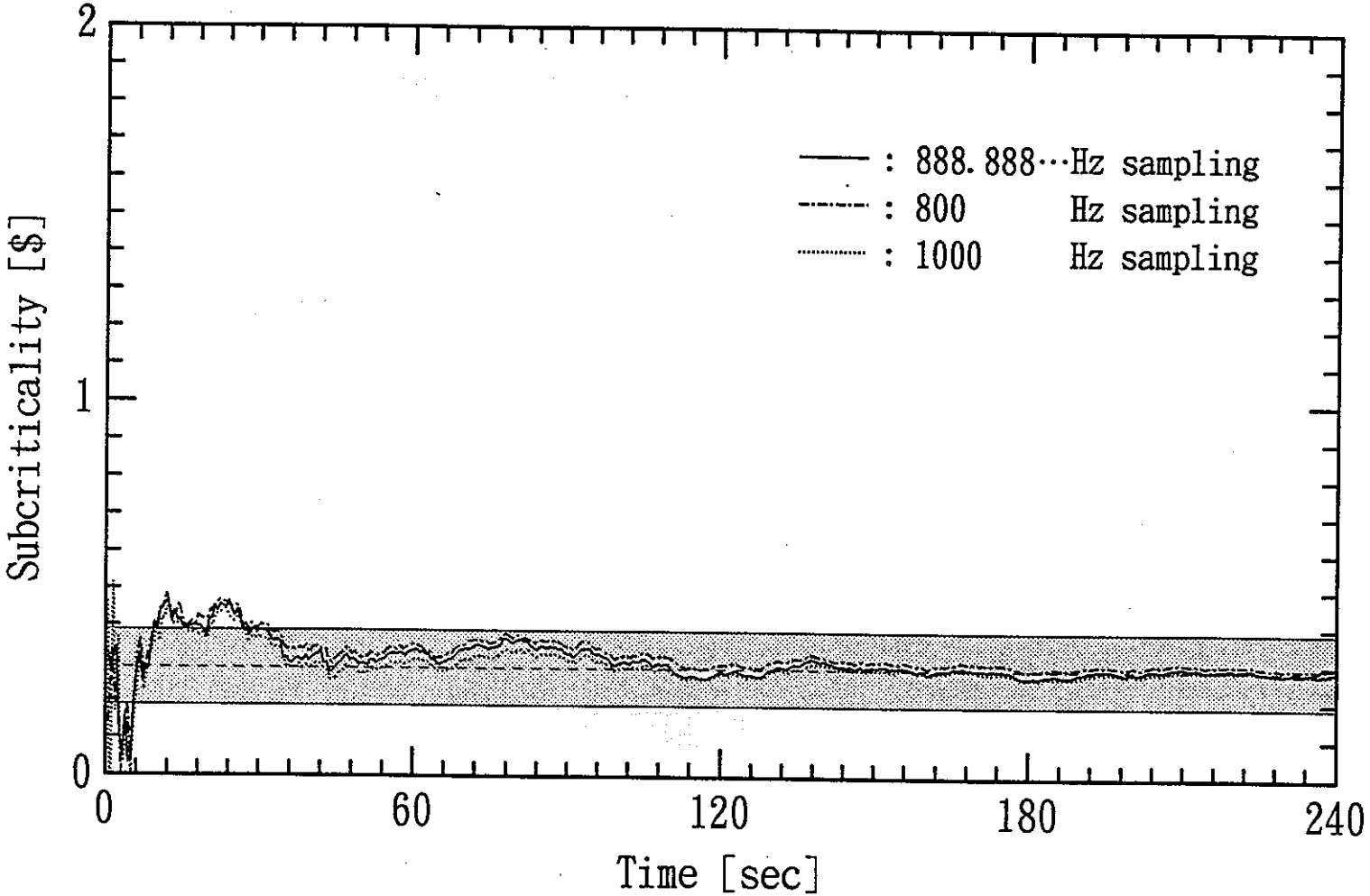
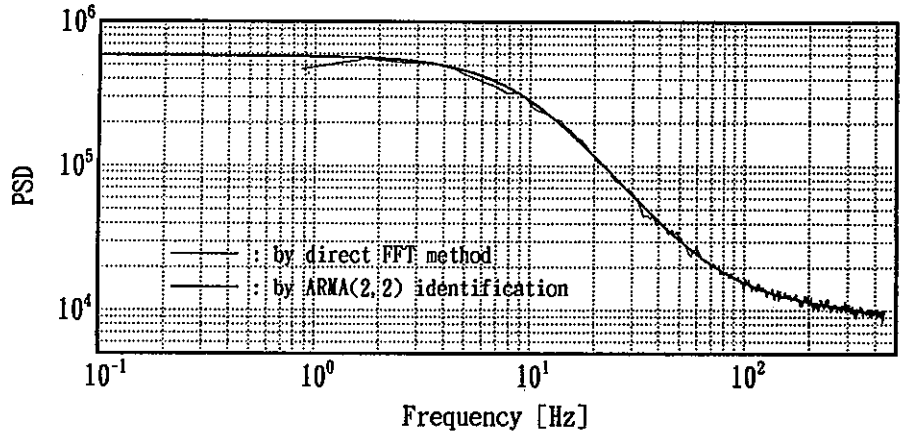
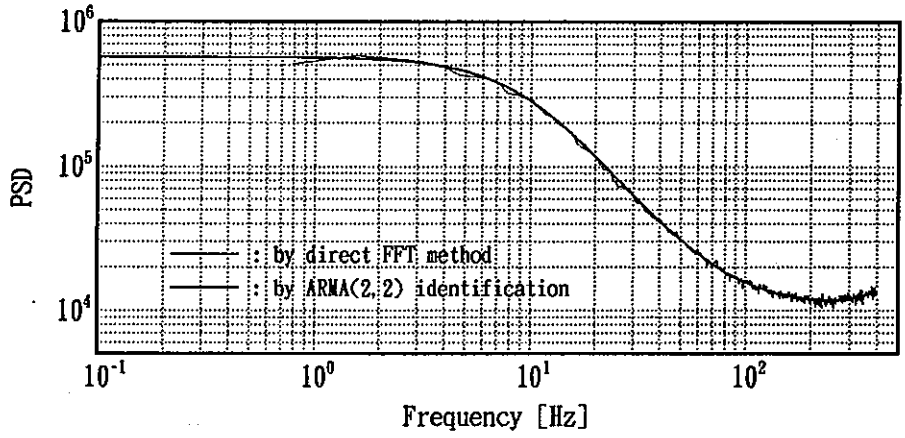


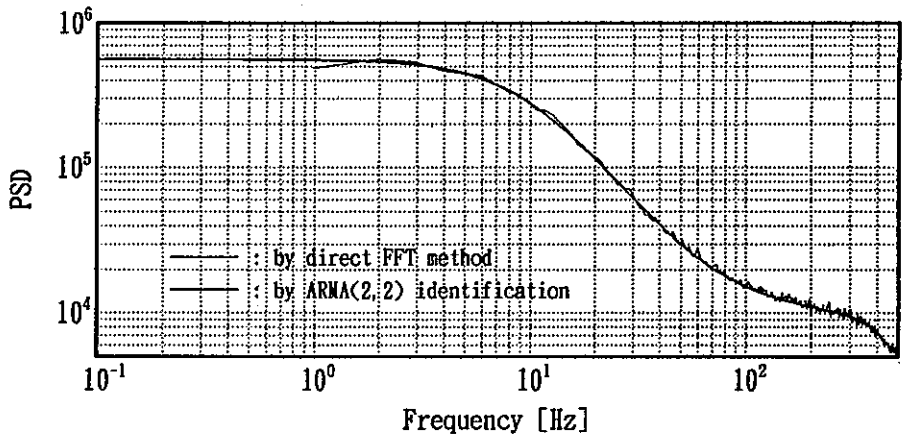
Fig. 3.6 Subcriticality estimations by ARMA(2,2) model identification for three time series sampled at 888.888…Hz, 800Hz and 1000Hz through Low-Pass filter with cutoff frequency 500Hz. [Reference subcriticality = 0.289 $.]



(a) Sampling frequency = 888.888…Hz.



(b) Sampling frequency = 800Hz.



(c) Sampling frequency = 1000Hz.

Fig. 3.7 Power spectrum densities estimated by two kinds of method for the time series (reference subcriticality 0.289 $) sampled at 888.888…Hz, 800Hz and 1000 Hz.

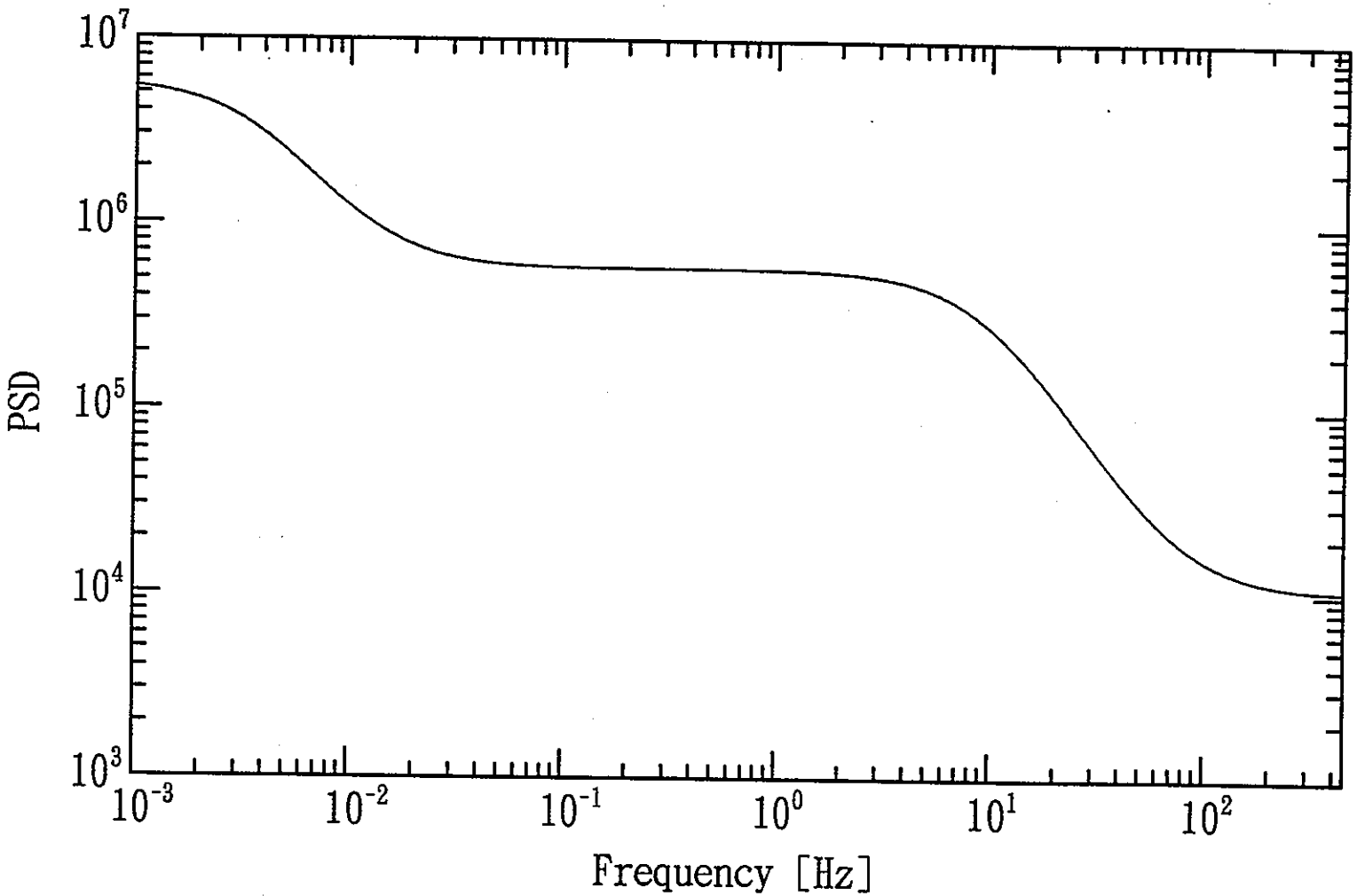


Fig. 3.8 Power spectrum density of theoretical model for the subcriticality 0.289 $.

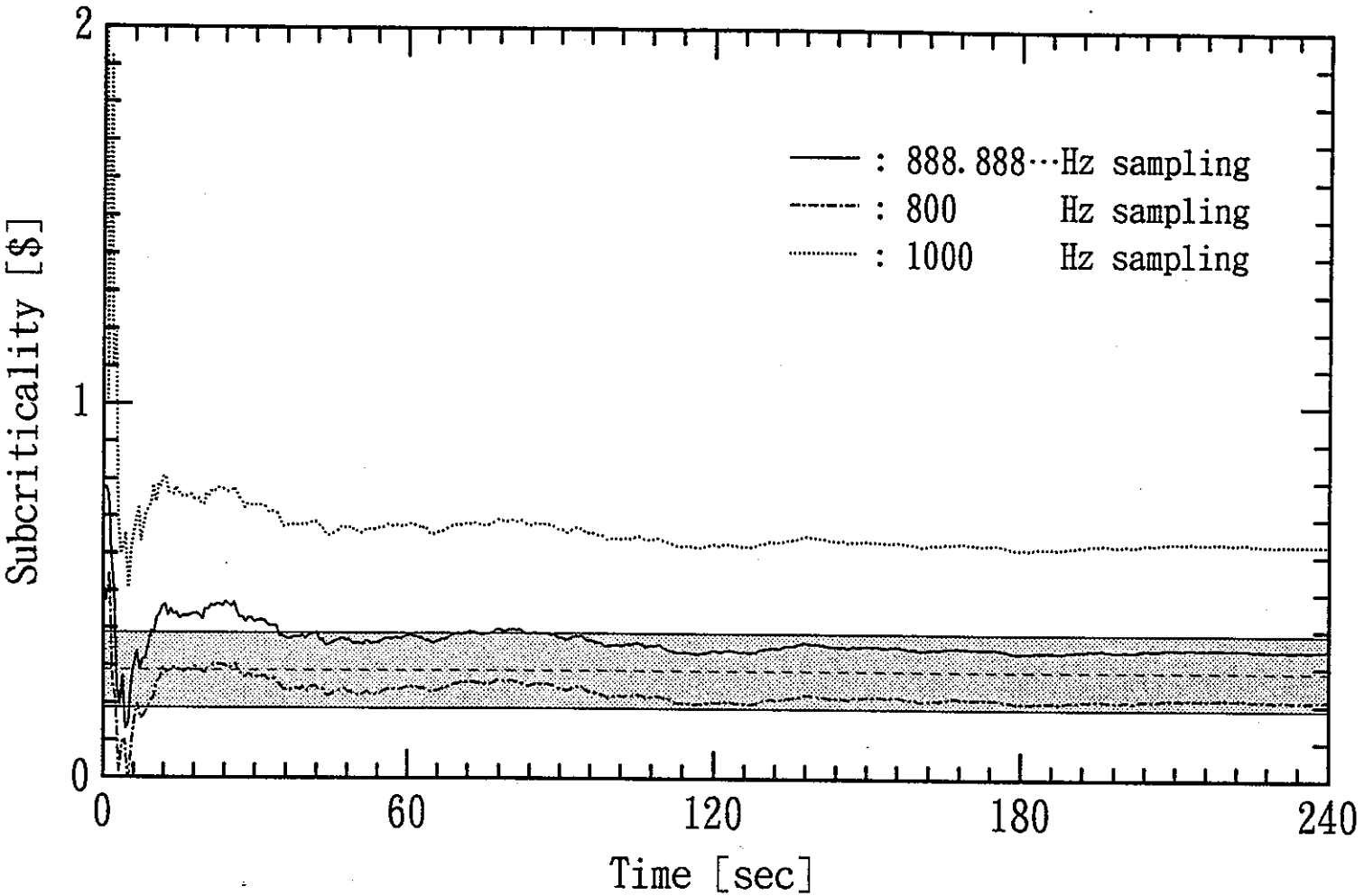
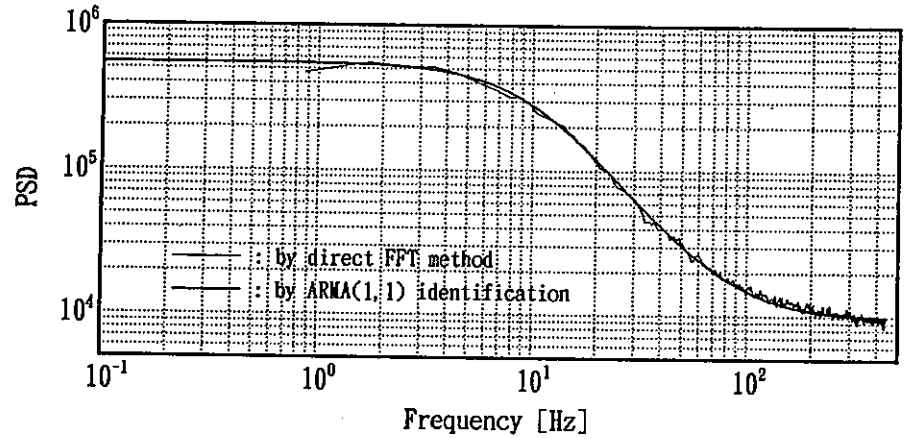
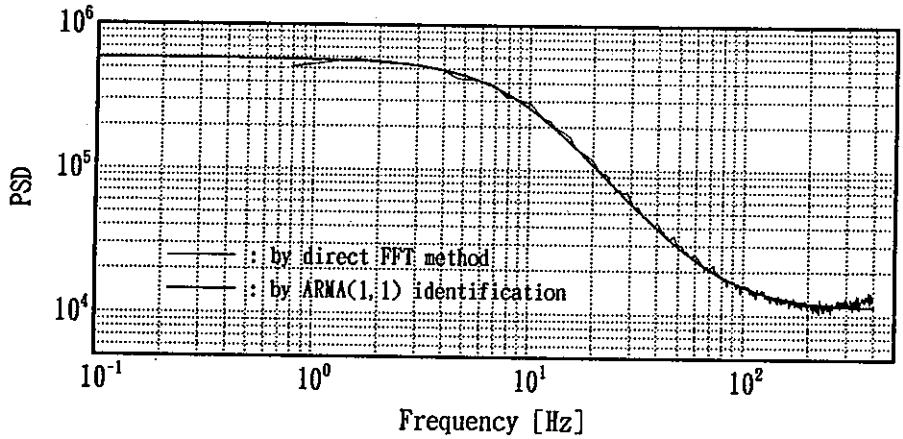


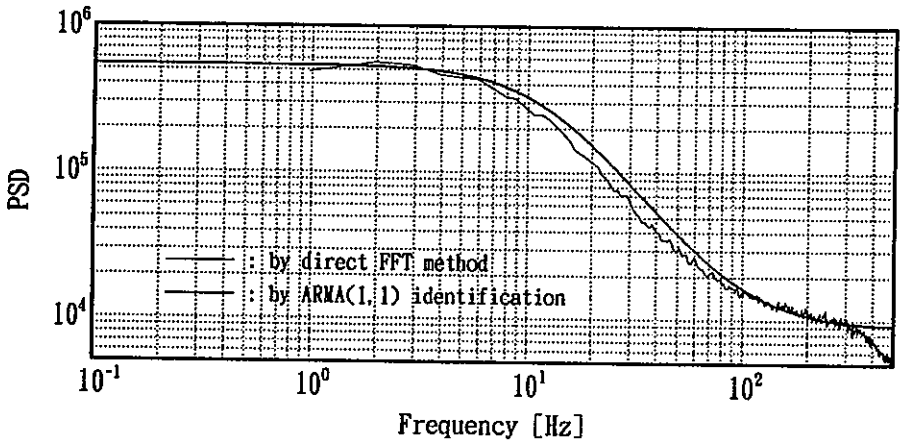
Fig. 3.9 Subcriticality estimations by ARMA(1,1) model identification for three time series sampled at 888.888…Hz, 800Hz and 1000Hz through Low-Pass filter with cutoff frequency 500Hz. [Reference subcriticality = 0.289 $.]



(a) Sampling frequency = 888.888…Hz.



(b) Sampling frequency = 800Hz.



(c) Sampling frequency = 1000Hz.

Fig. 3.10 Power spectrum densities estimated by two kinds of method for the time series (reference subcriticality 0.289 $) sampled at 888.888…Hz, 800Hz and 1000Hz.

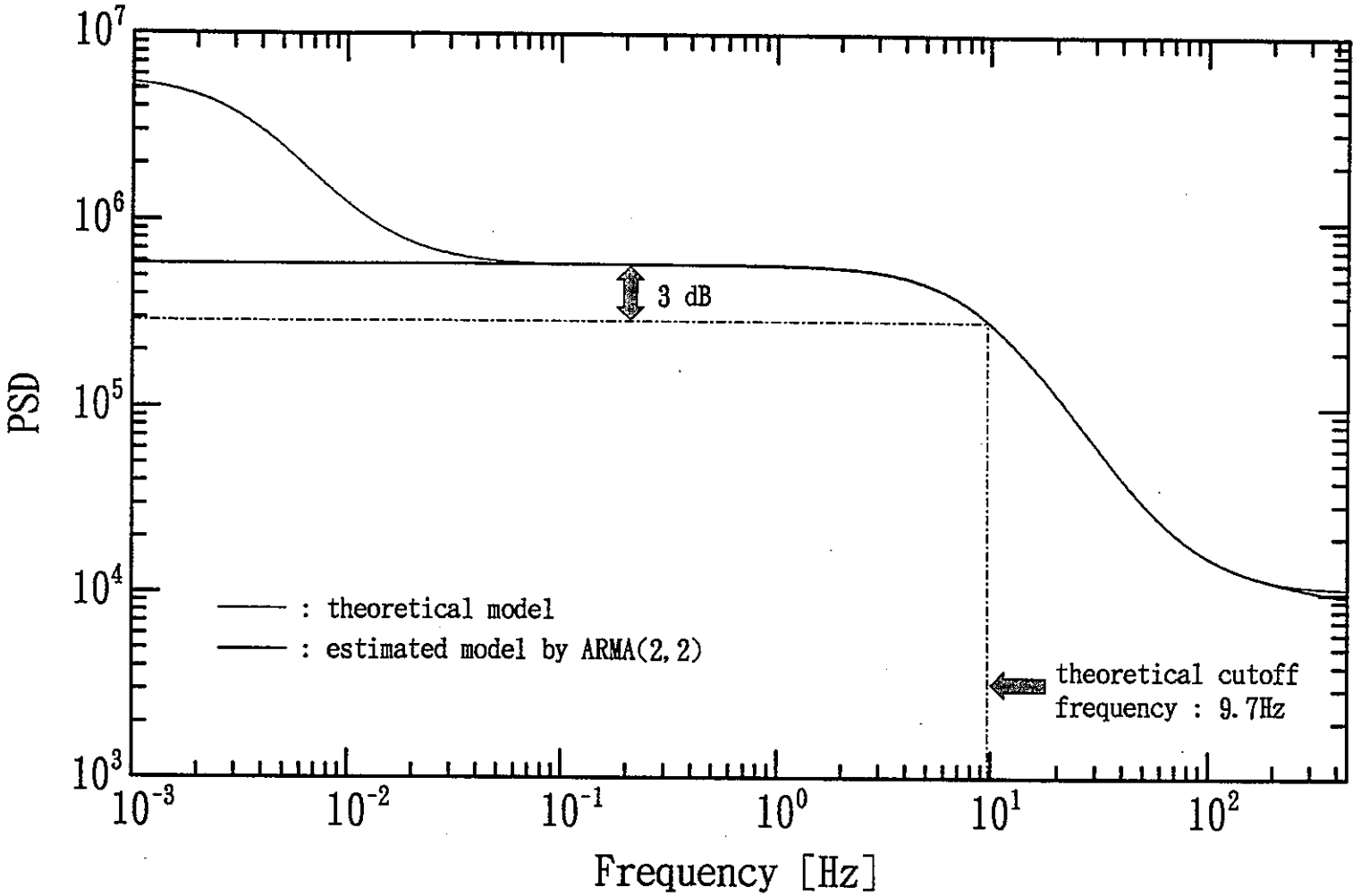


Fig. 3.11 Power spectrum density of the theoretical model for subcriticality 0.289$ and that of the estimated model by ARMA(2,2) model identification of UTR-KINKI reference subcriticality 0.289$.

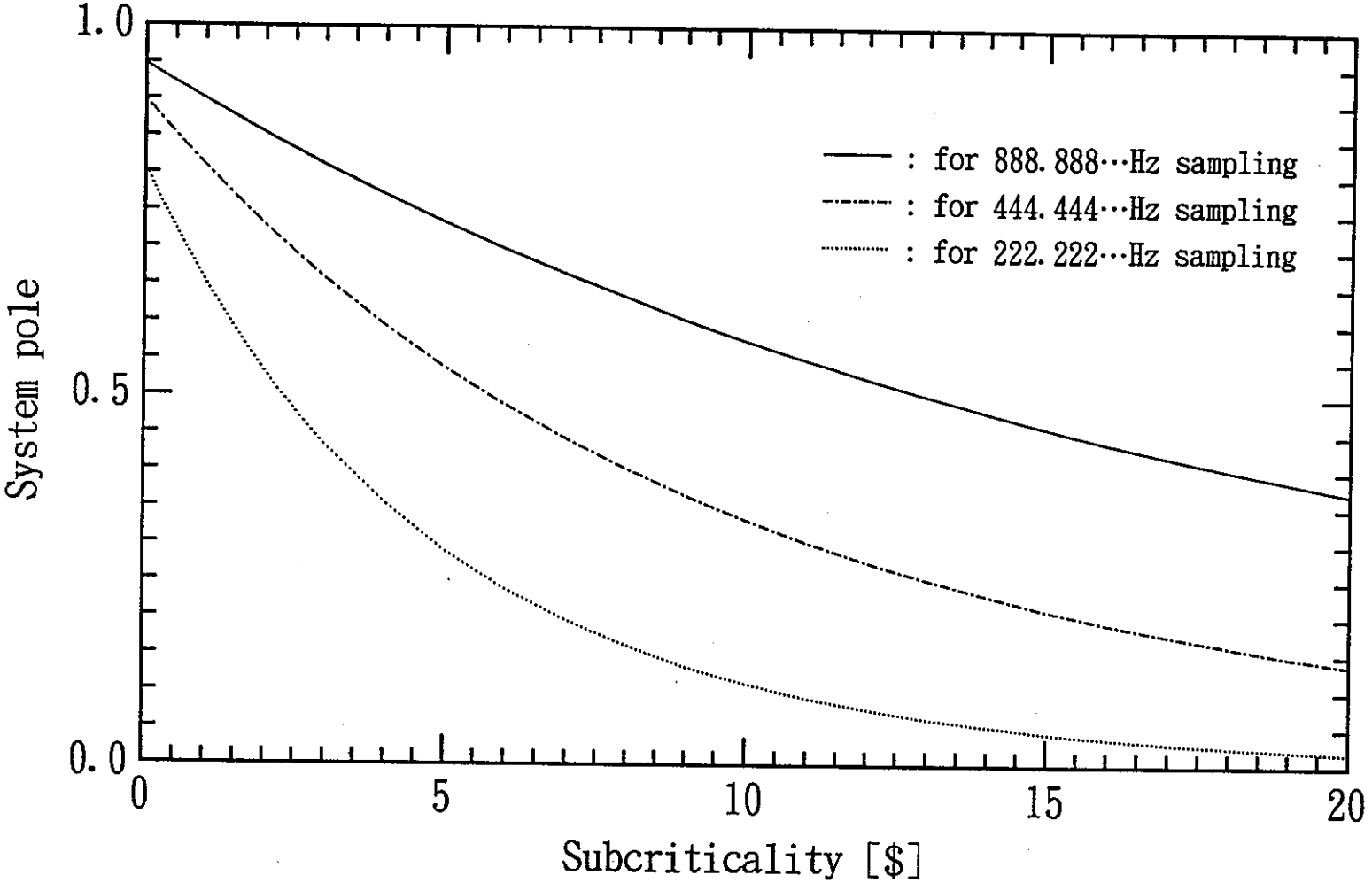
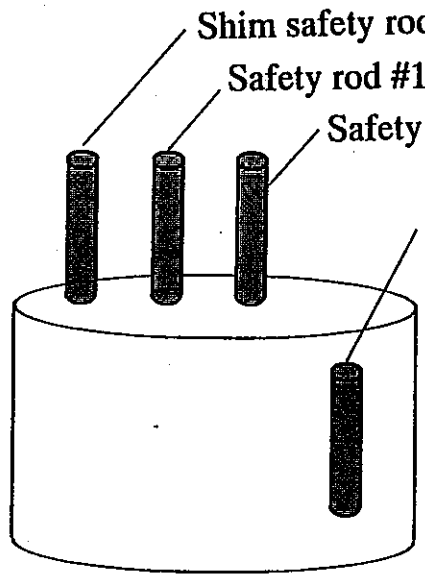
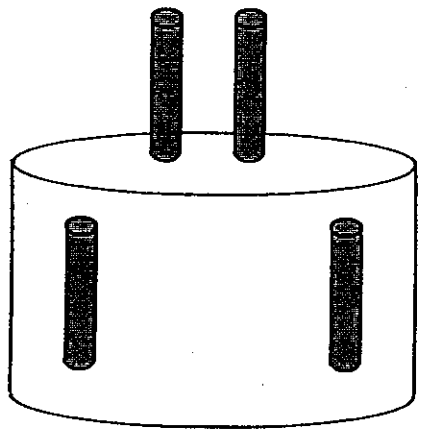


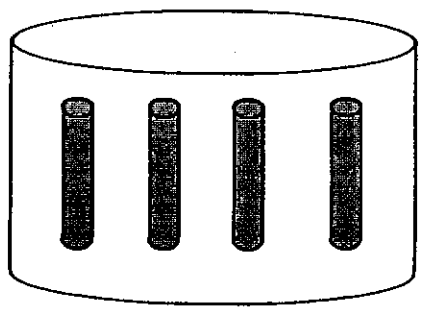
Fig. 3.12 Calibration curves for conversion of the system pole of ARMA model to subcriticality.



(a) Critical system.



(b) Subcritical system 1.
subcriticality 0.7129 $



(c) Subcritical system 2.
subcriticality 2.259 $

Fig. 3.13 Position of the control rods at critical and the two types of subcritical system.

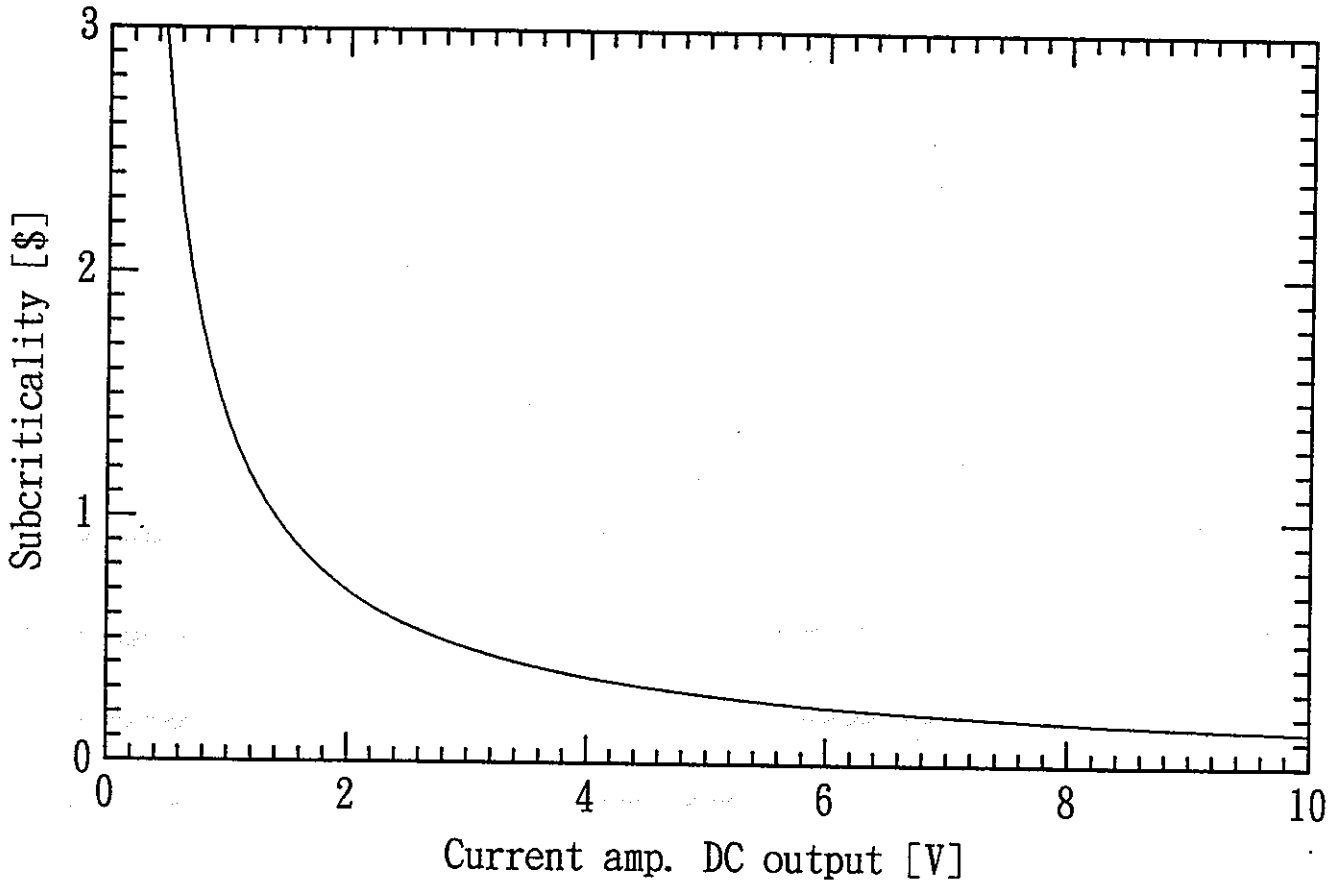
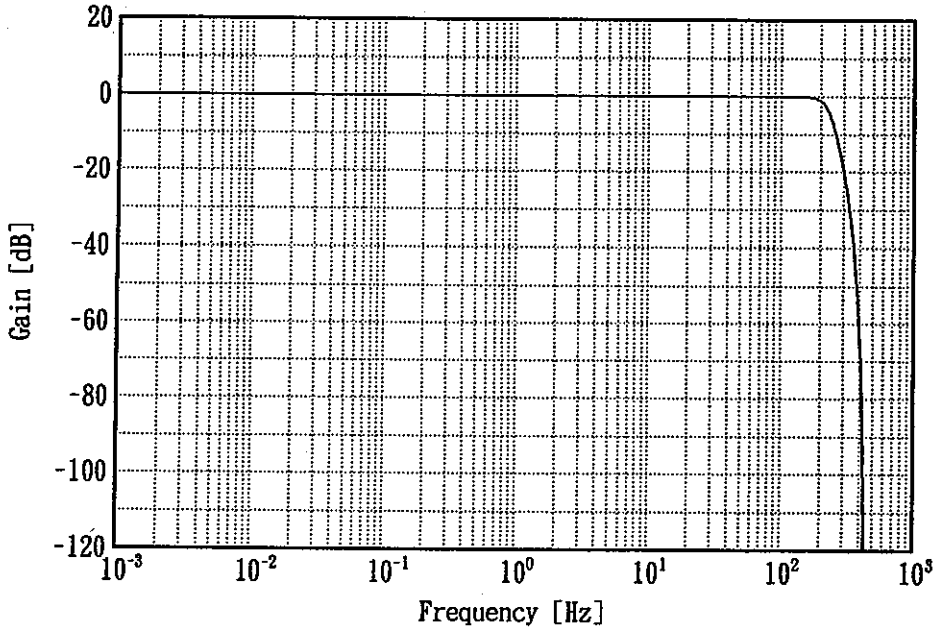
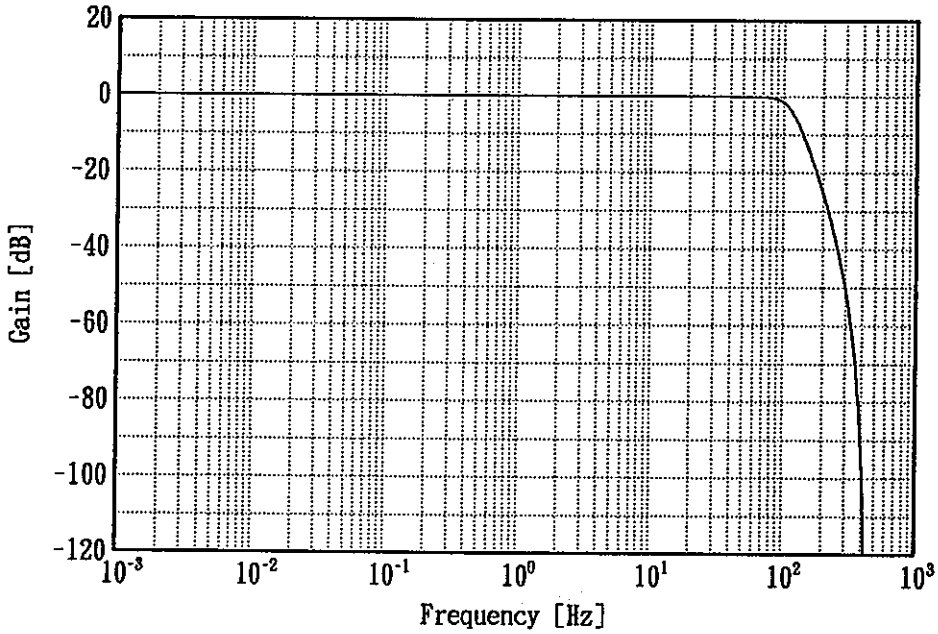


Fig. 3.14 Calibration curves for conversion of the DC output of current amplifire to subcriticality.



(a) Cutoff frequency at 222.222…Hz.



(b) Cutoff frequency at 111.111…Hz.

Fig. 4.1 Frequency characteristics of digital Low-Pass filter for preventing aliasing.

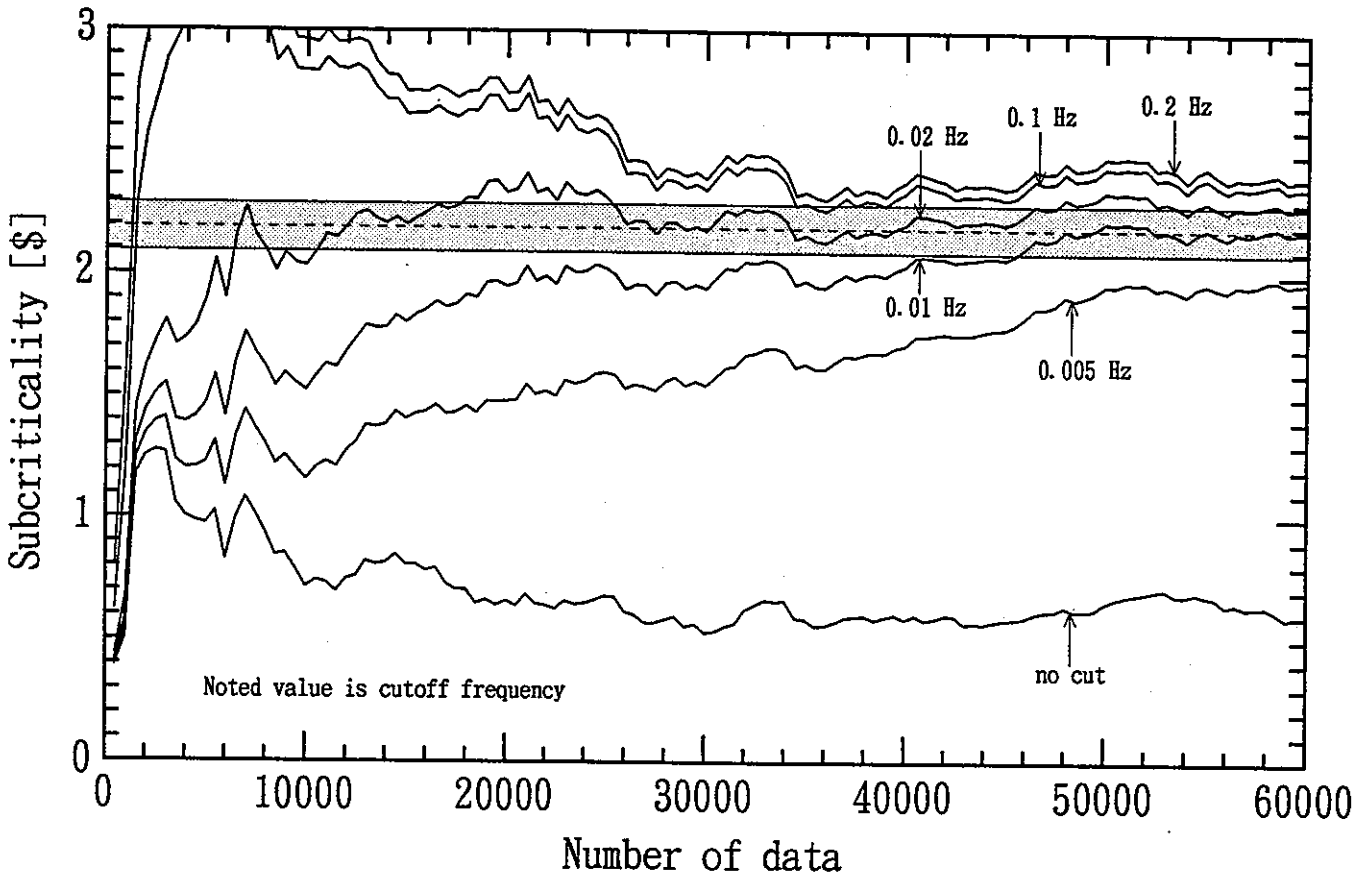


Fig. 4.2.a Subcriticality estimation for "ID003.888 DATA" filtered by the various High-Pass filters.
[Reference subcriticality = 2.19 $.]

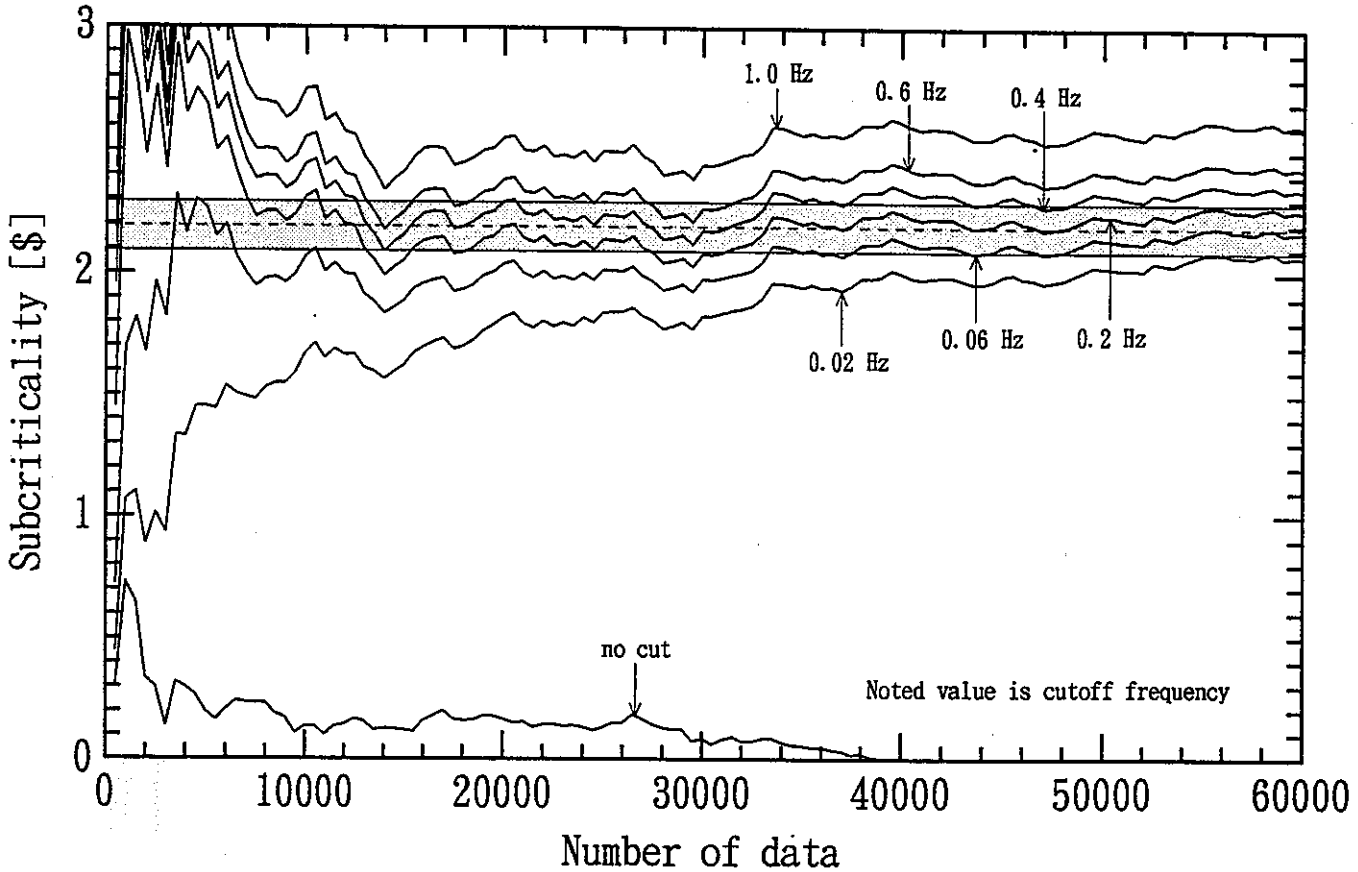


Fig. 4.2.b Subcriticality estimation for "ID003.444 DATA" filtered by the various High-Pass filters.
[Reference subcriticality = 2.19 $.]

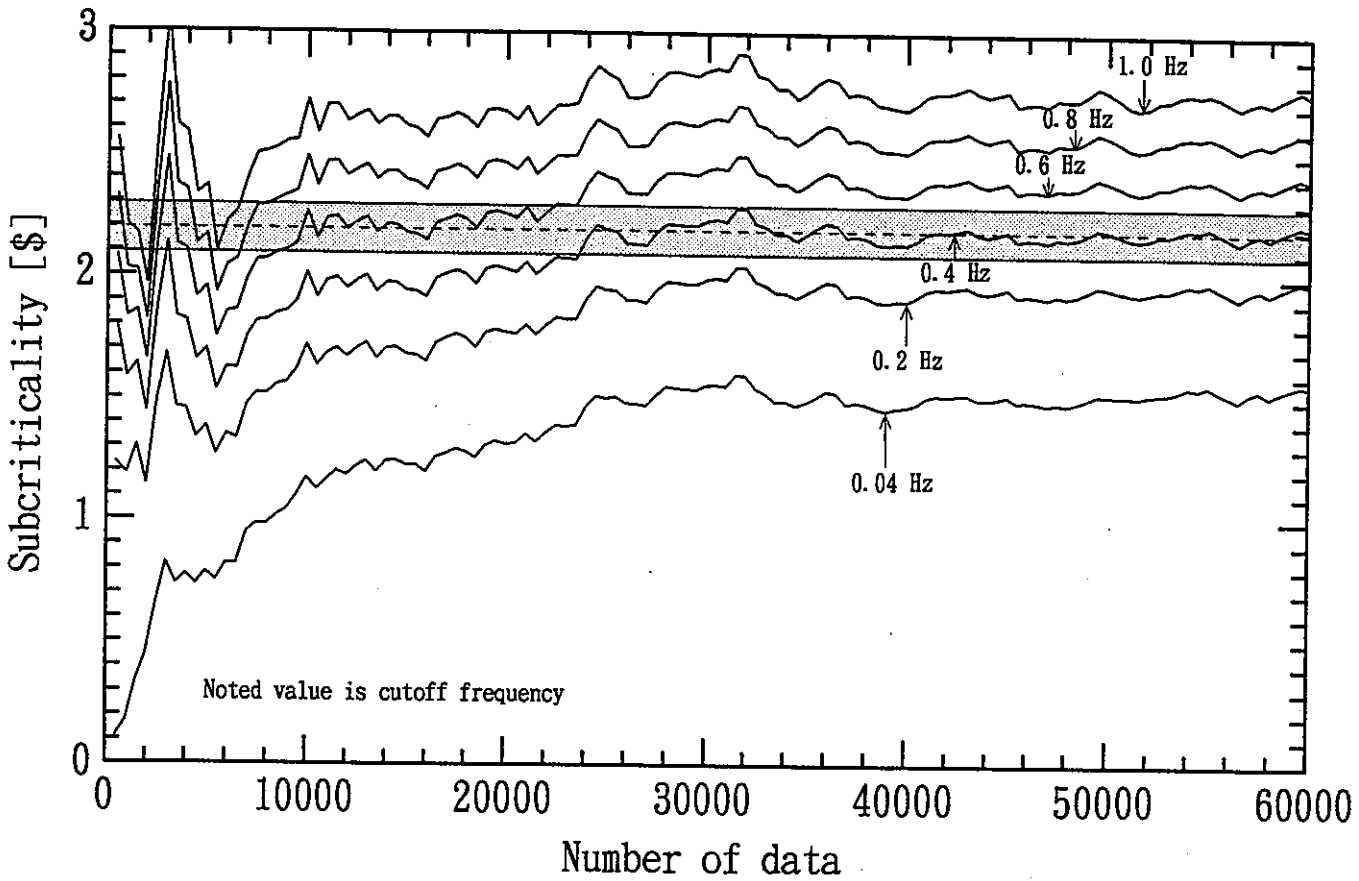


Fig. 4.2.c Subcriticality estimation for "ID003.222 DATA" filtered by the various High-Pass filters.
[Reference scubcriticality = 2.19 $.]

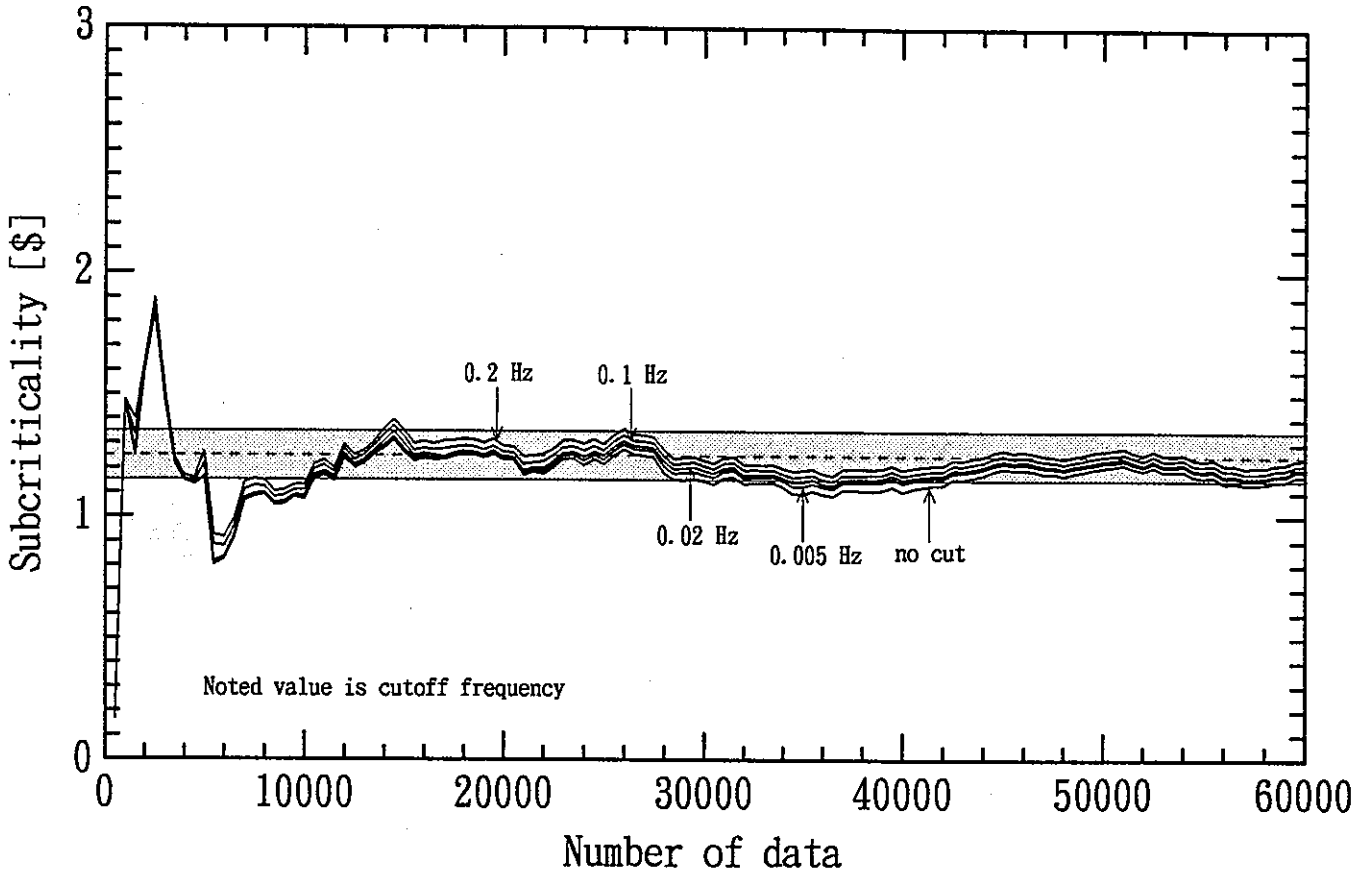


Fig. 4.3.a Subcriticality estimation for "ID005.888 DATA" filtered by the various High-Pass filters.
[Reference subcriticality = 1.25 $.]

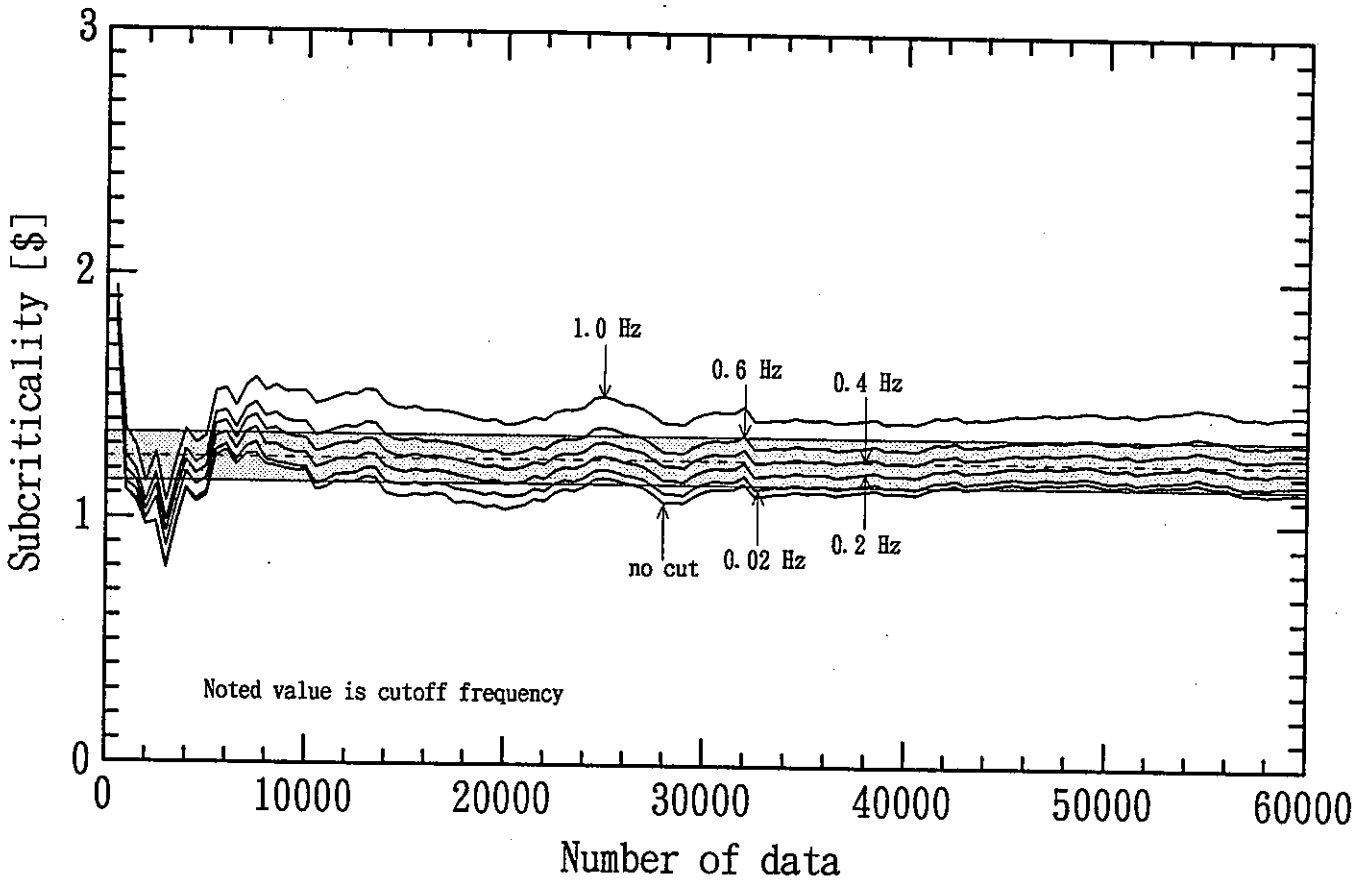


Fig. 4.3.b Subcriticality estimation for "ID005.444 DATA" filtered by the various High-Pass filters.
[Reference subcriticality = 1.25 $.]

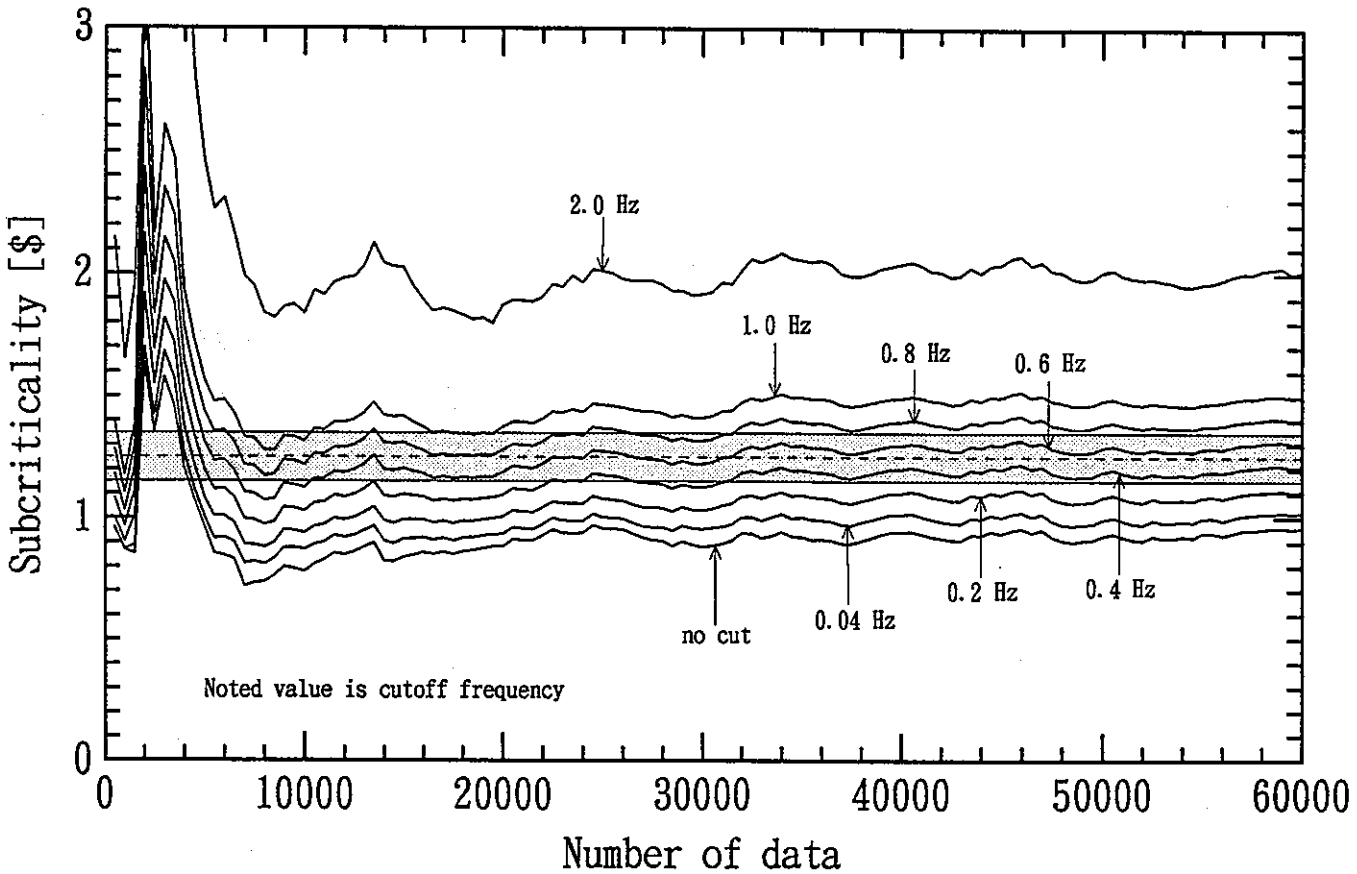


Fig. 4.3.c Subcriticality estimation for "ID005.222 DATA" filtered by the various High-Pass filters.
[Reference subcriticality = 1.25 $.]

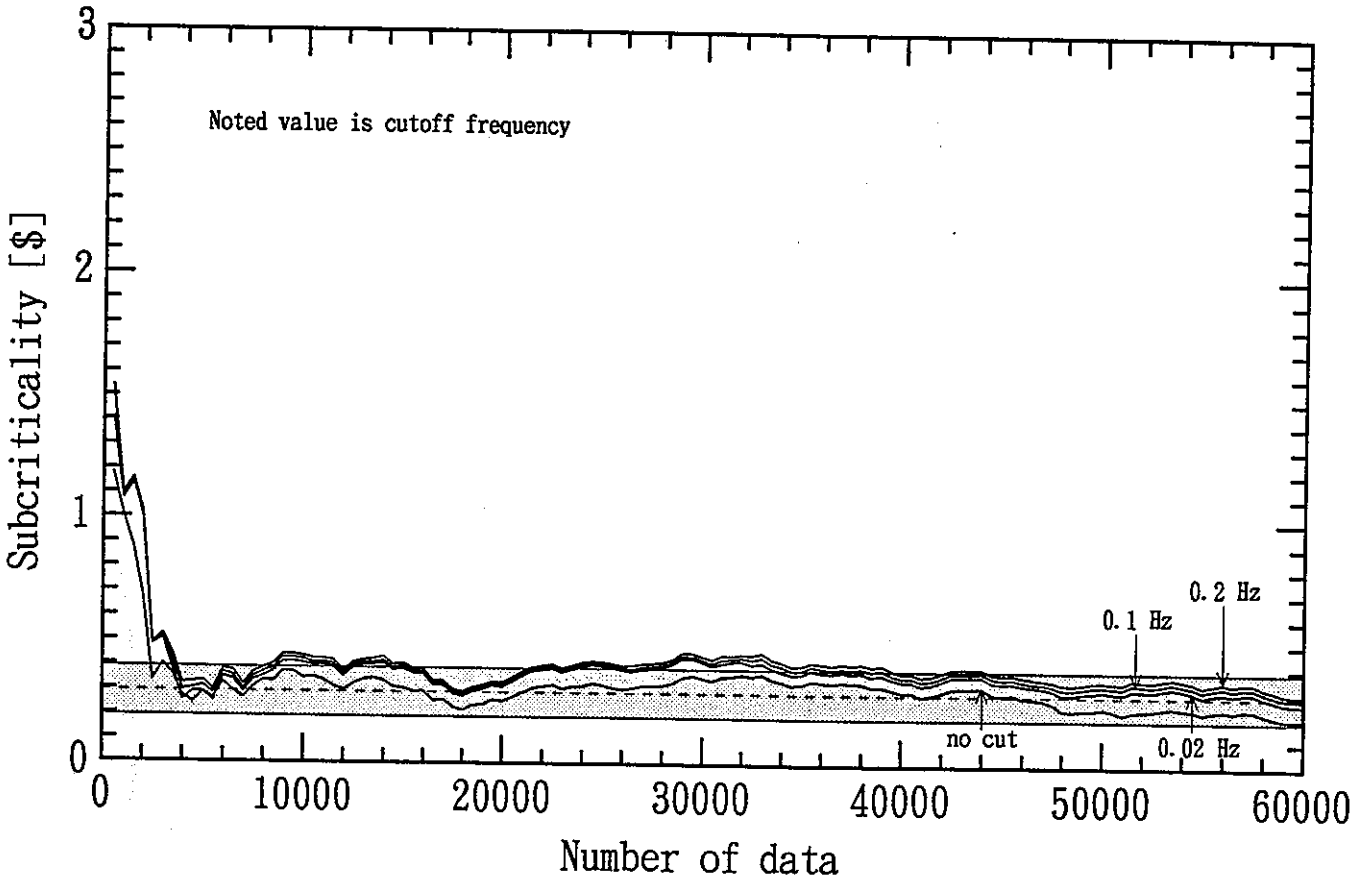


Fig. 4.4.a Subcriticality estimation for "ID010.888 DATA" filtered by the various High-Pass filters.
[Reference subcriticality = 0.289 $.]

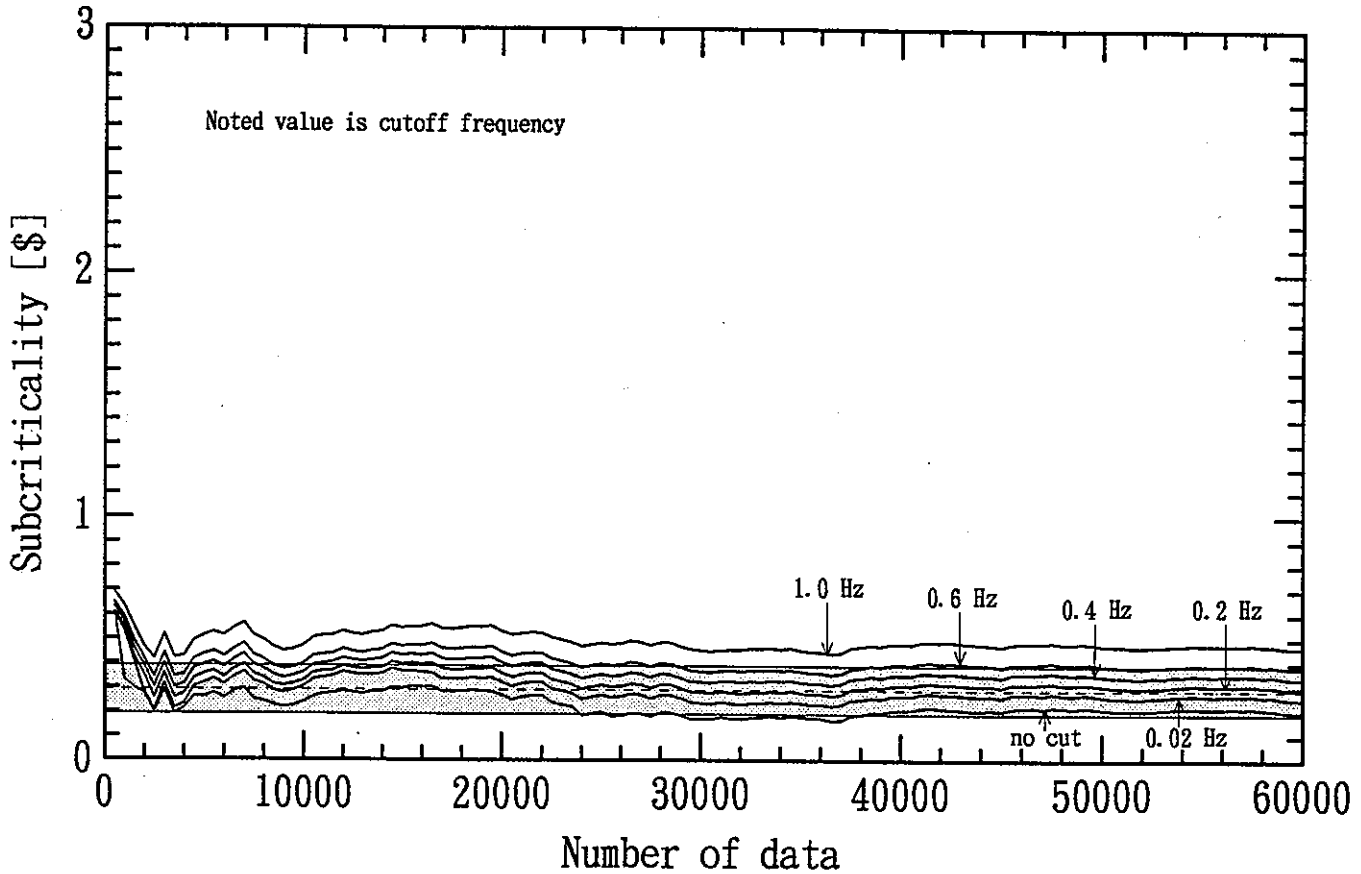


Fig. 4.4.b Subcriticality estimation for "ID010.444 DATA" filtered by the various High-Pass filters.
[Reference subcriticality = 0.289 $.]

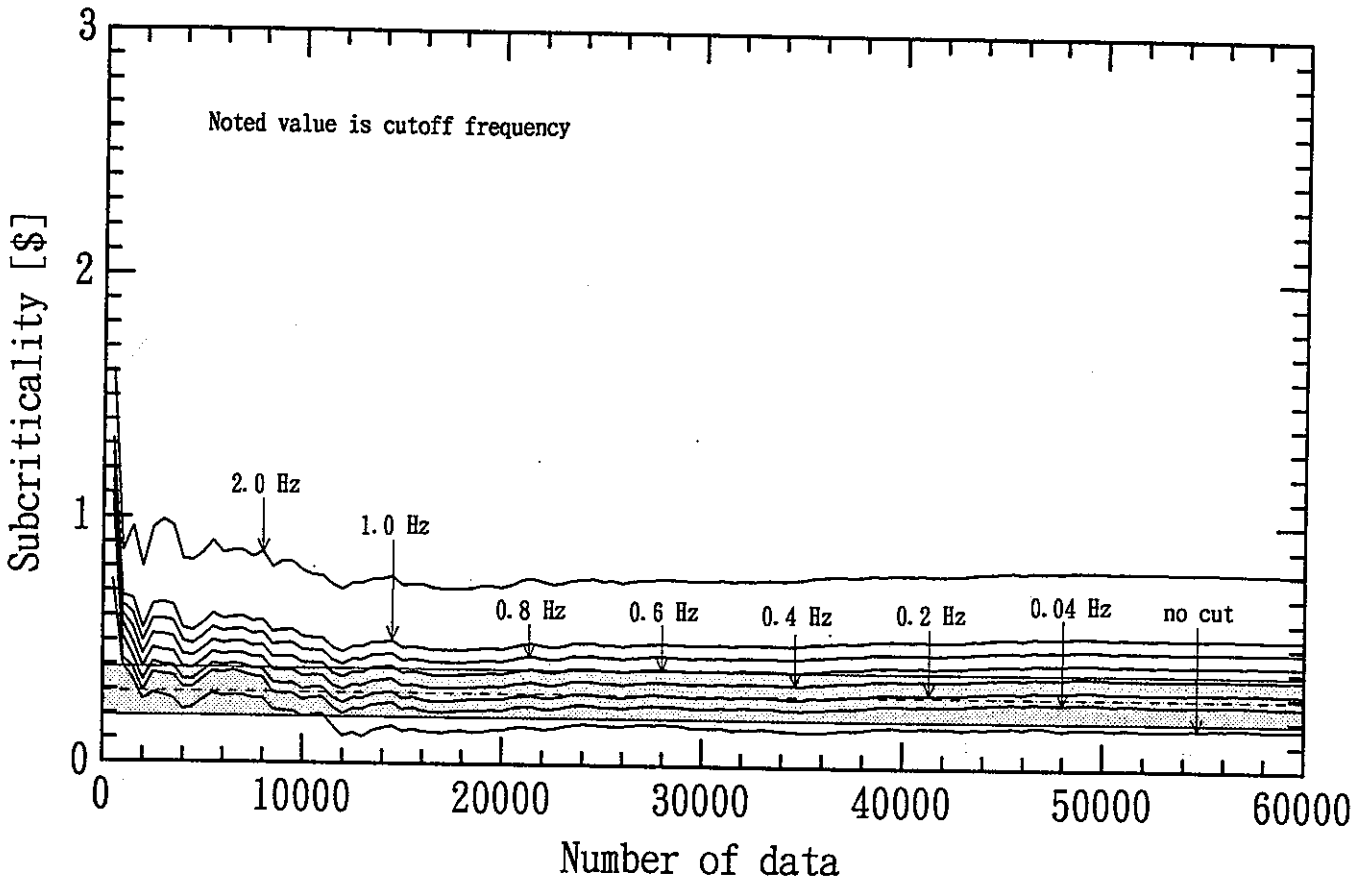
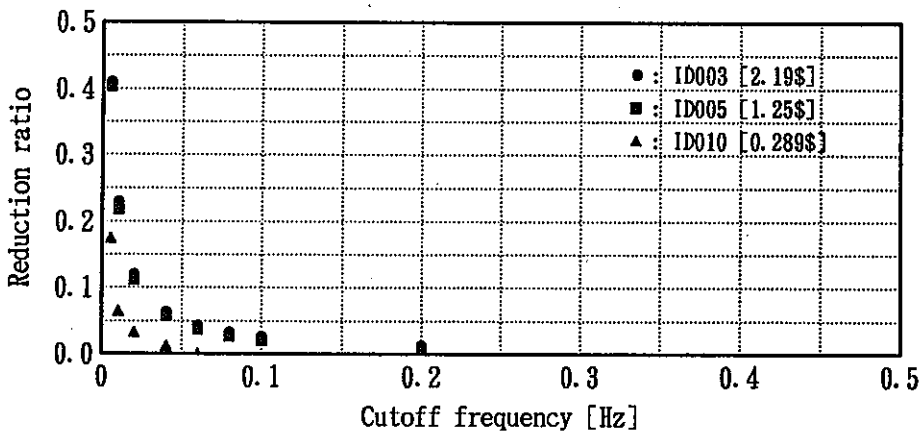
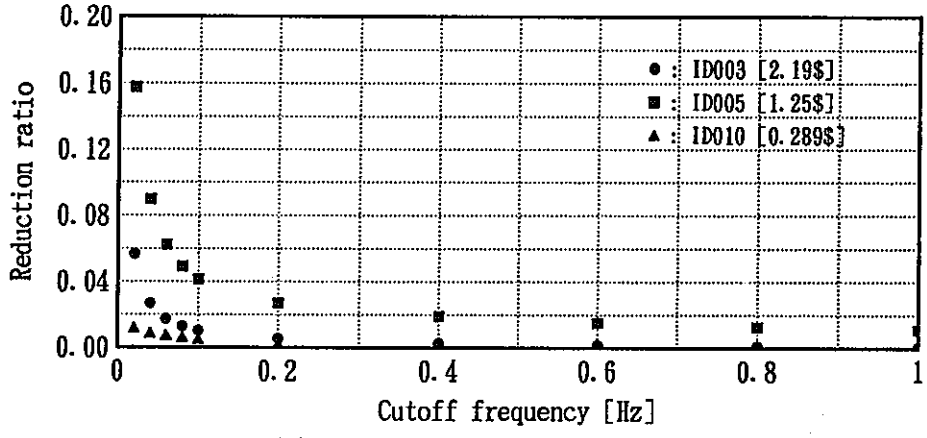


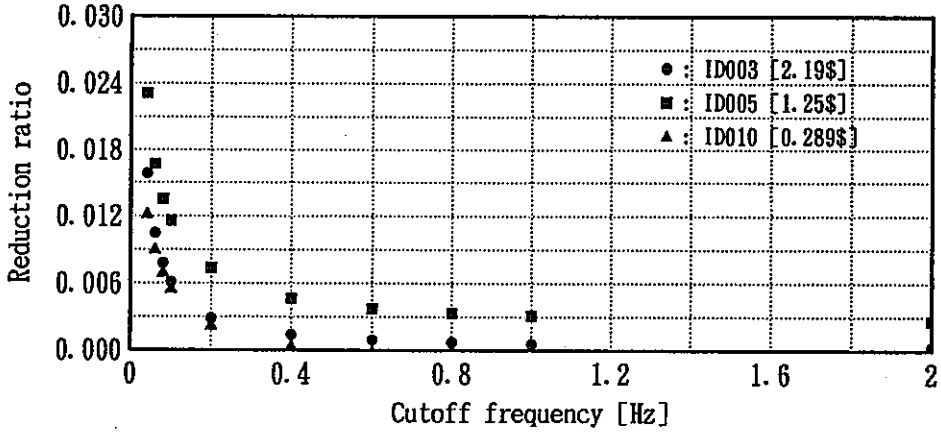
Fig. 4.4.c Subcriticality estimation for "ID010.222 DATA" filtered by the various High-Pass filters.
[Reference subcriticality = 0.289 $.]



(a) Sampling frequency = 888.888…Hz.

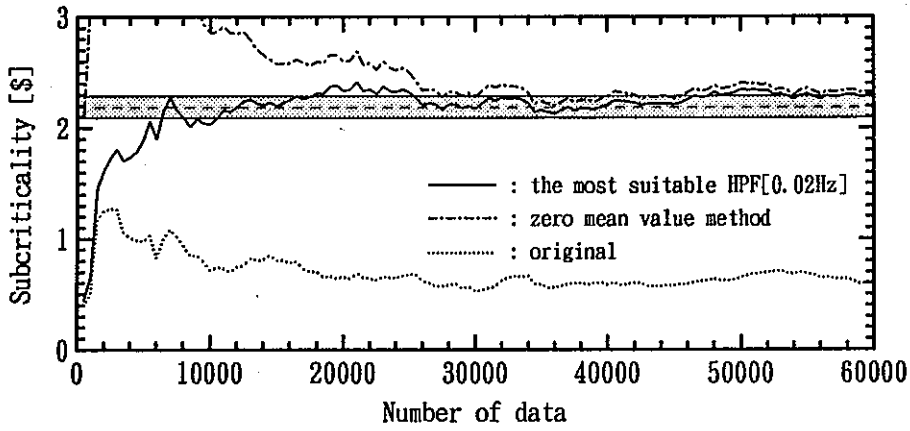


(b) Sampling frequency = 444.444…Hz.

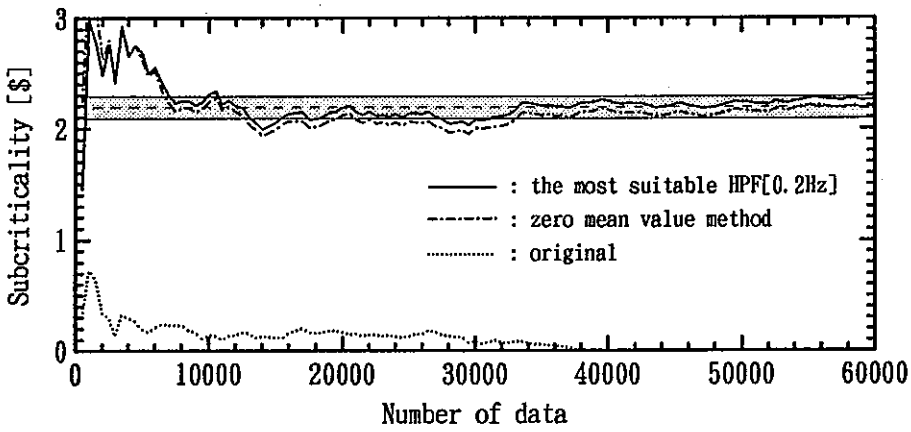


(c) Sampling Frequency = 222.222…Hz.

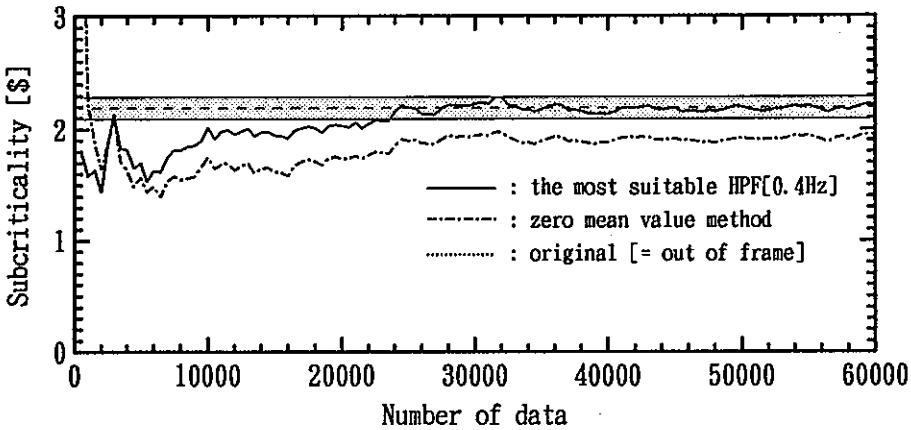
Fig. 4.5 Dependence of DC component reduction ratio on the cutoff frequency of High-Pass filter by filtering for the fundamental time series.



(a) Sampling frequency = 888.888…Hz.

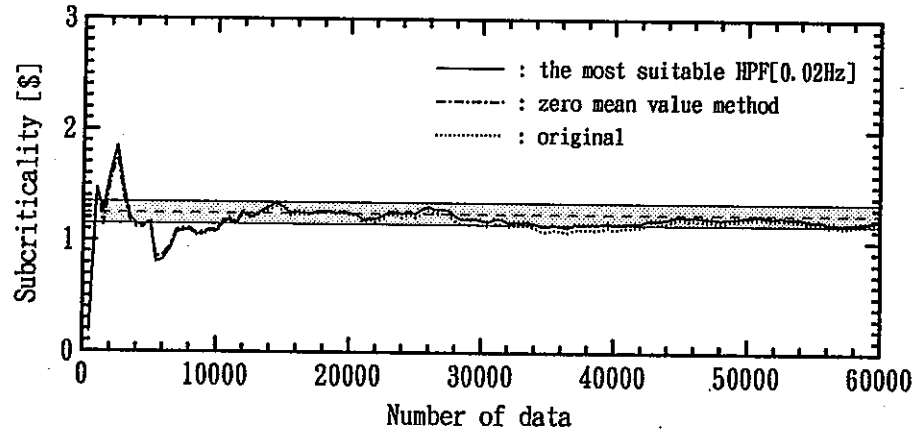


(b) Sampling frequency = 444.444…Hz.

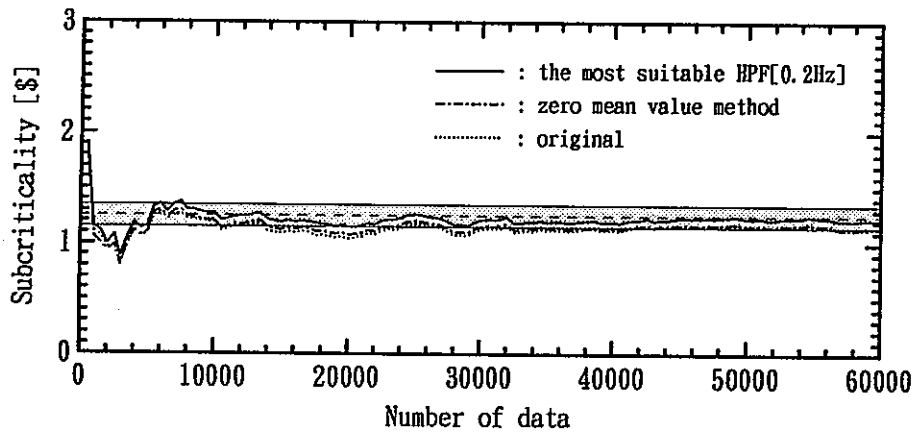


(c) Sampling frequency = 222.222…Hz.

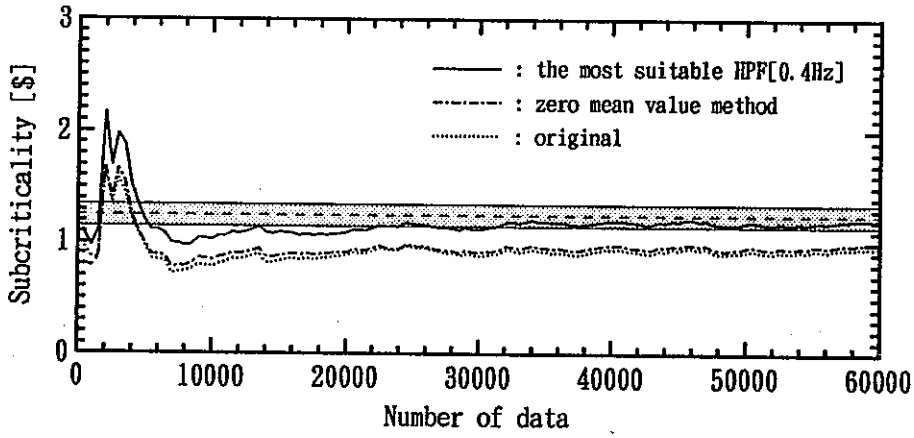
Fig. 4.6.a Subcriticality estimations for the time series through the most suitable High-Pass filter and the time series without DC component in "ID003 Series DATA".
[Reference subcriticality = 2.19 $.]



(a) Sampling frequency = 888.888…Hz.

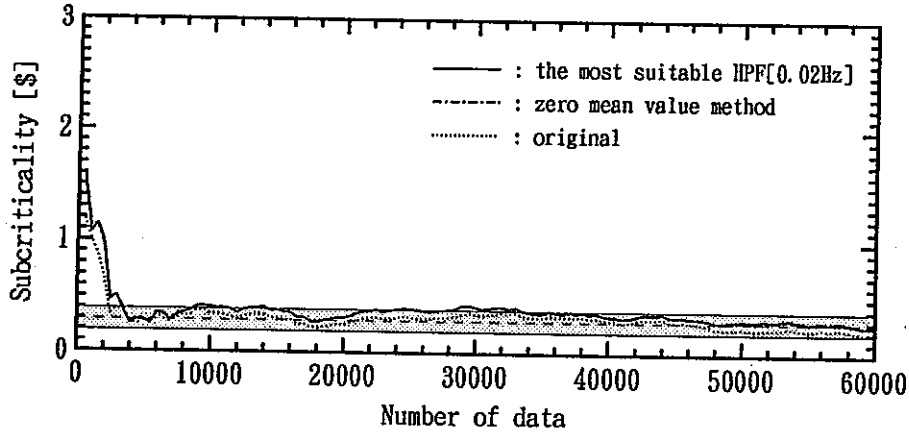


(b) Sampling frequency = 444.444…Hz.

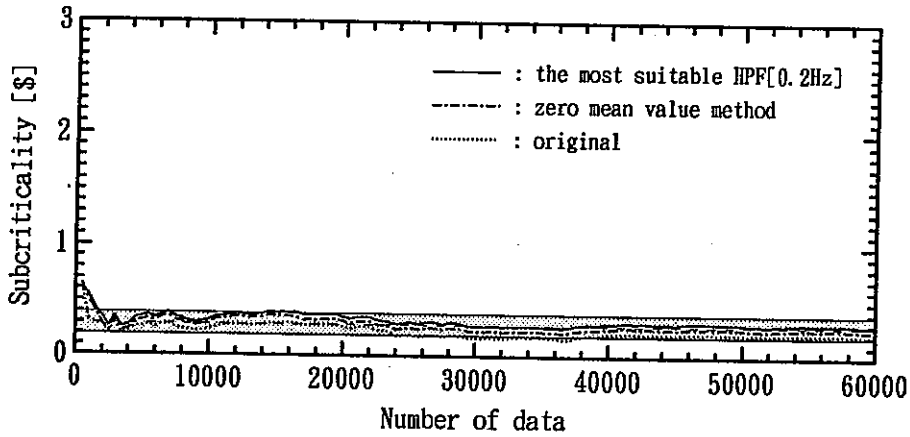


(c) Sampling frequency = 222.222…Hz.

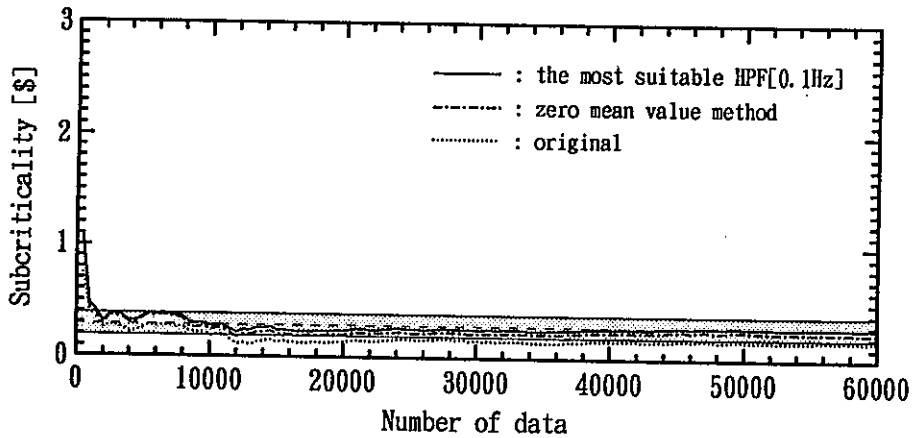
Fig. 4.6.b Subcriticality estimations for the time series through the most suitable High-Pass filter and the time series without DC component in "ID005 Series DATA".
[Reference subcriticality = 1.25 $.]



(a) Sampling frequency = 888.888…Hz.



(b) Sampling frequency = 444.444…Hz.



(c) Sampling frequency = 222.222…Hz.

Fig. 4.6.c Subcriticality estimations for the time series through the most suitable High-Pass filter and the time series without DC component in "ID010 Series DATA".
[Reference subcriticality = 0.289 $.]

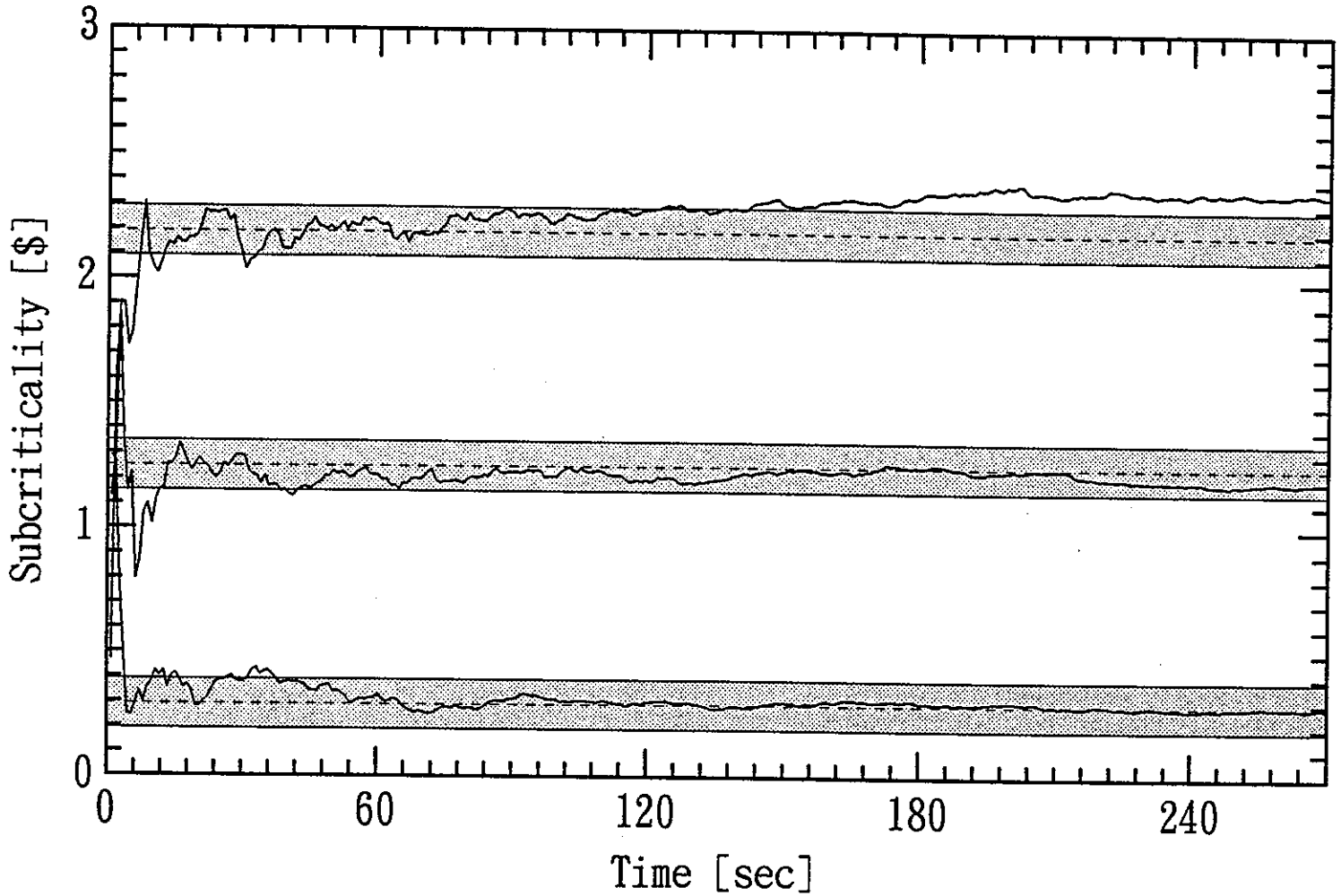


Fig. 4.7.a Subcriticality estimations for the time series sampled at 888.888…Hz.
[Reference subcriticality = 2.19 $, 1.25 $ and 0.289 $.]

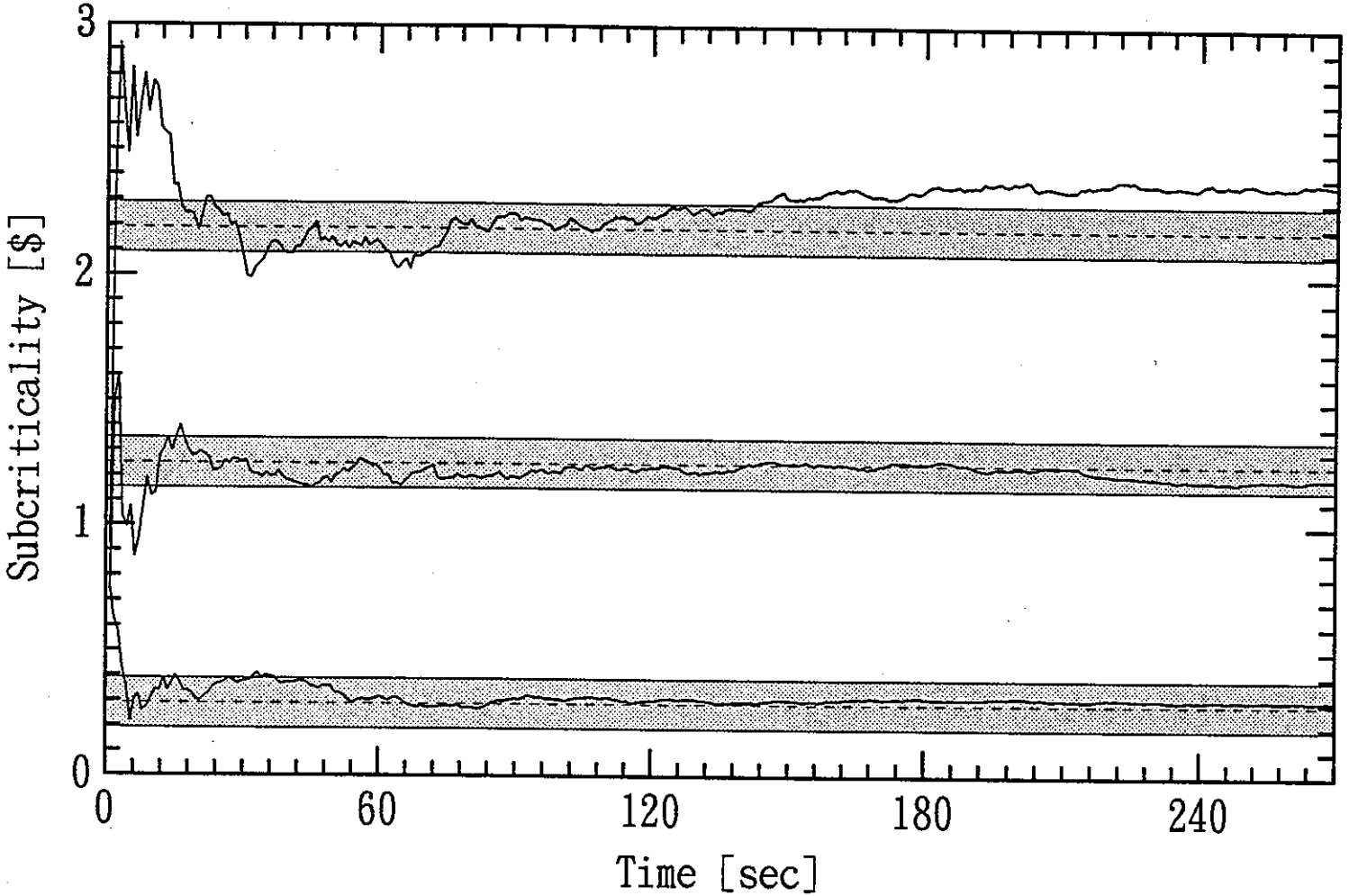


Fig. 4.7.b Subcriticality estimations for the time series sampled at 444.444…Hz.
[Reference subcriticality = 2.19 $, 1.25 $, amd 0.289 $.]

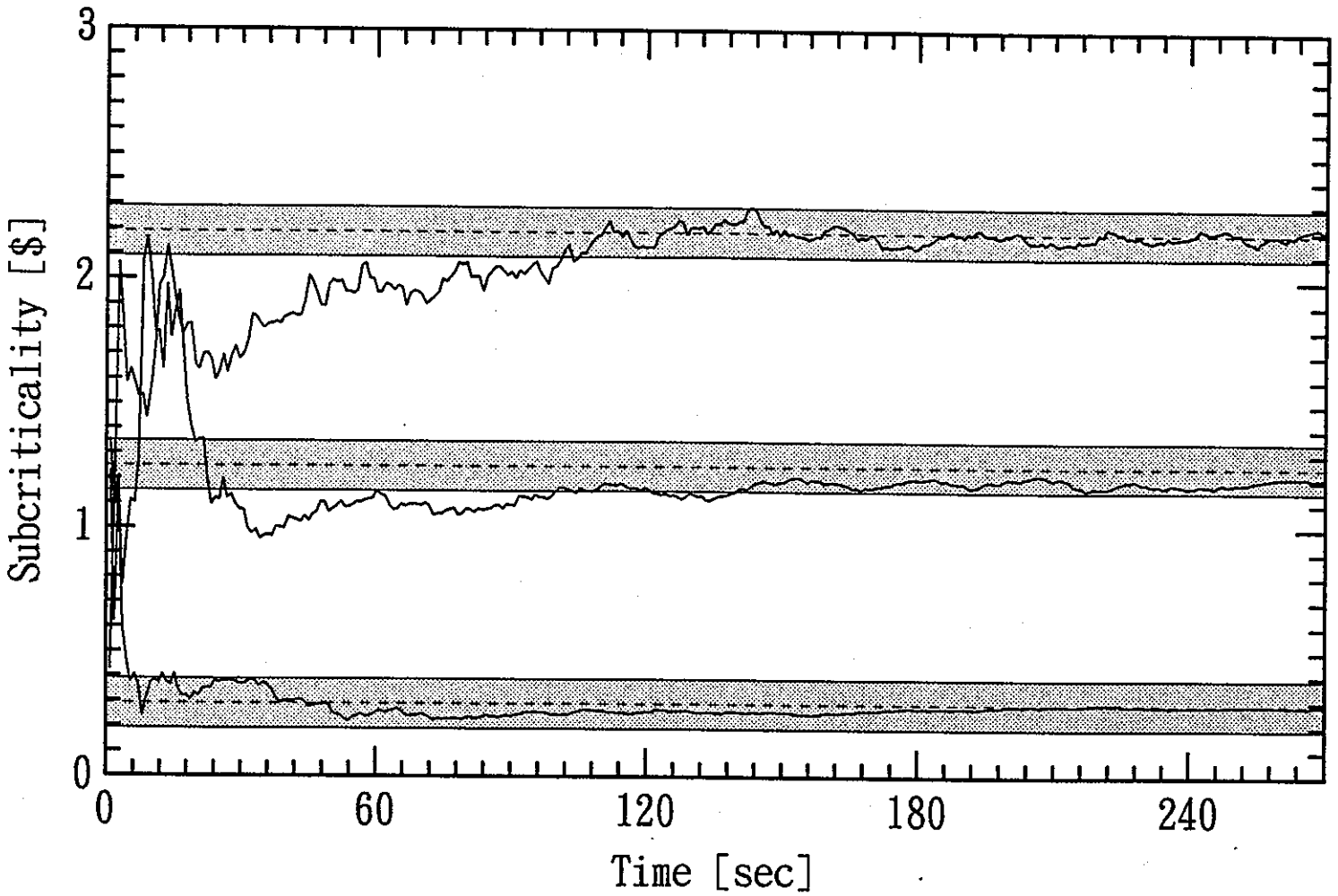
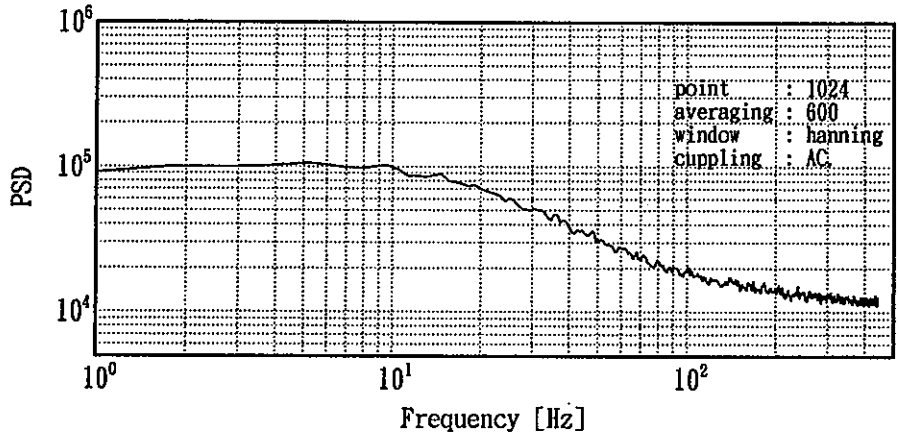
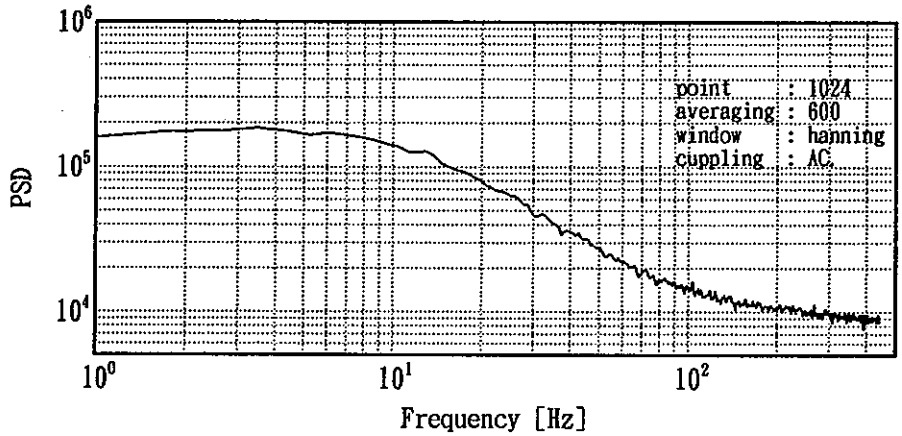


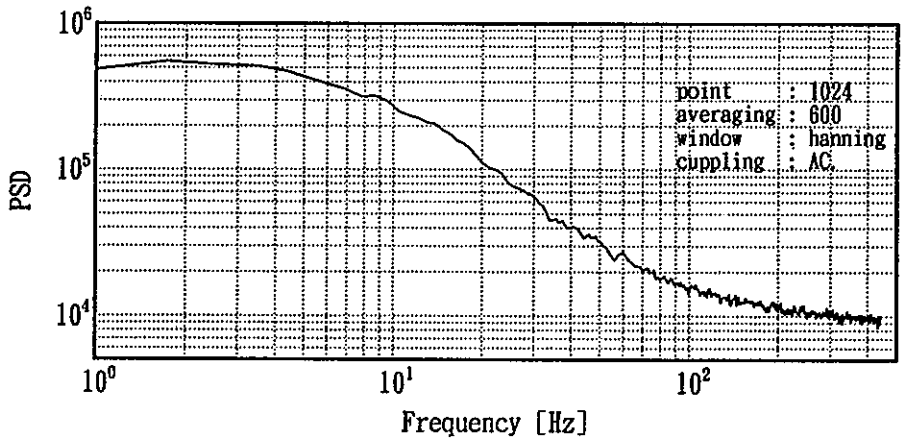
Fig. 4.7.c Subcriticality estimations for the time series sampled at 222.222…Hz.
[Reference subcriticality = 2.19 $, 1.25 $ and 0.289 $.]



(a) Reference subcriticality = 2.19 $.

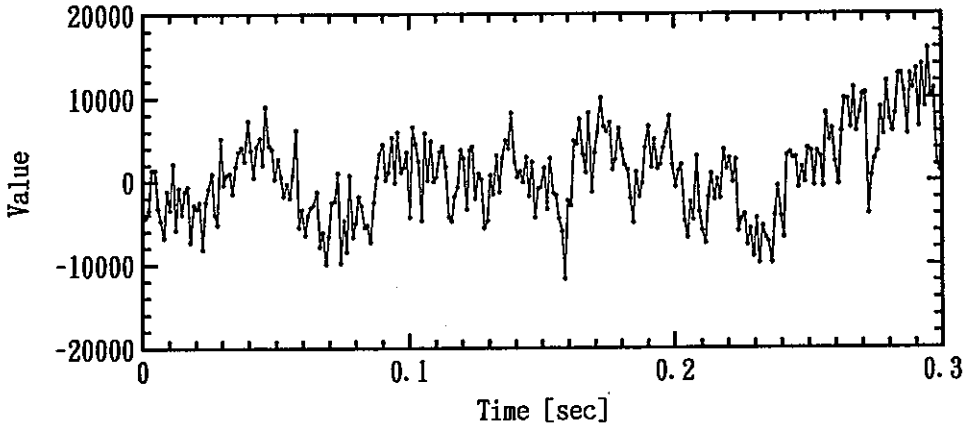


(b) Reference subcriticality = 1.25 $.

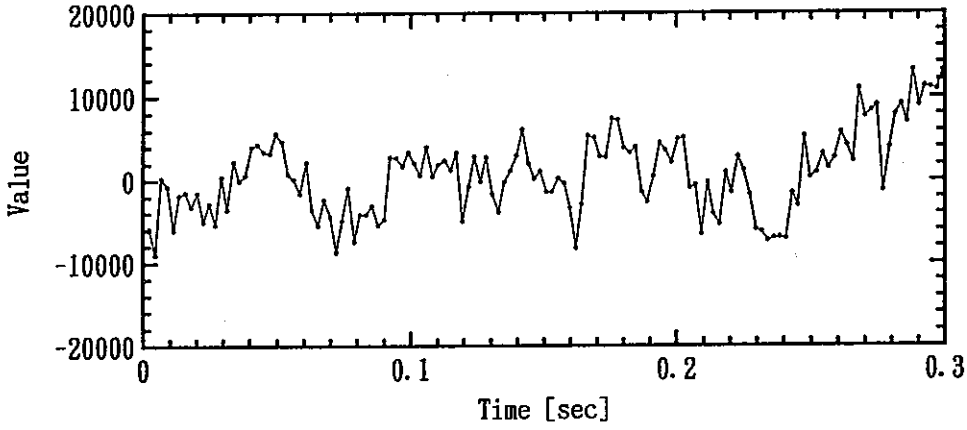


(c) Reference subcriticality = 0.289 $.

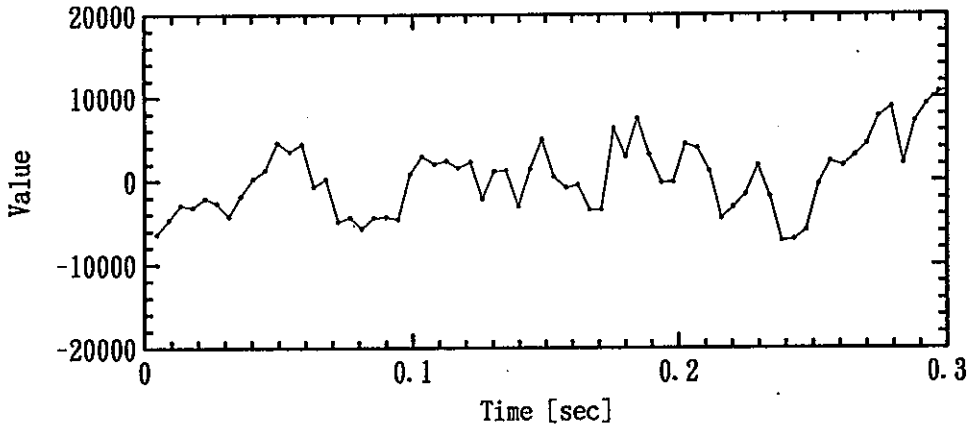
Fig. 4.8 Power spectrum density of the time series sampled at 888.888…Hz by direct FFT method.



(a) Sampling frequency = 888.888…Hz.

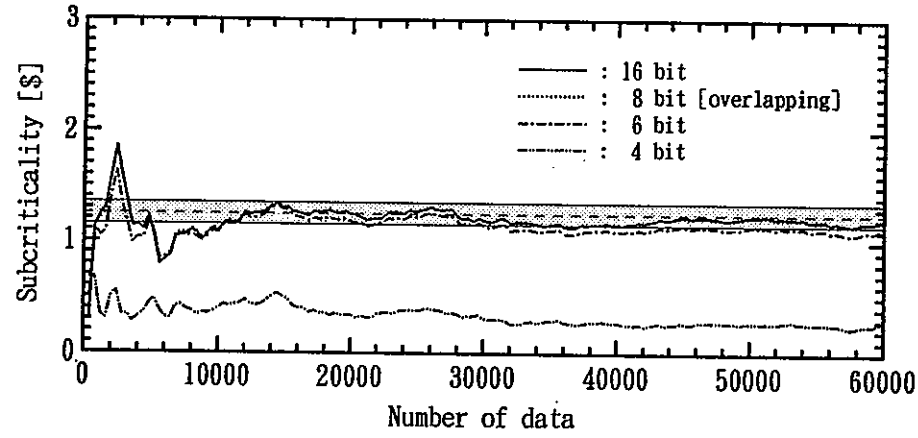


(b) Sampling frequency = 444.444…Hz.

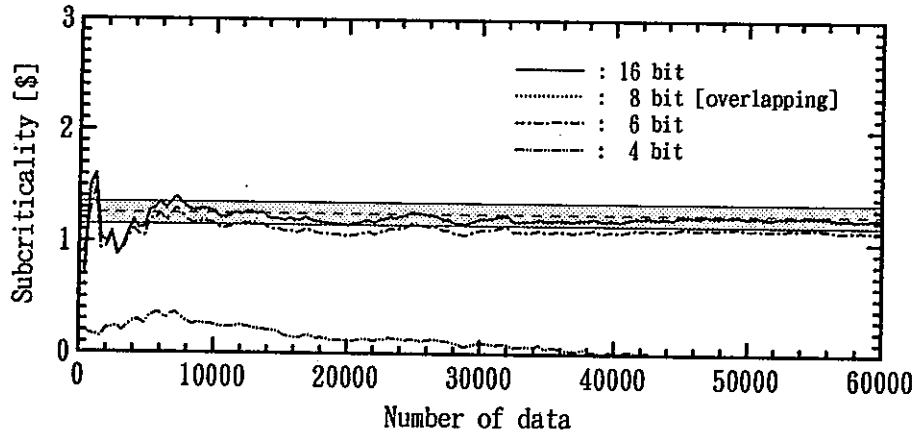


(c) Sampling frequency = 222.222…Hz.

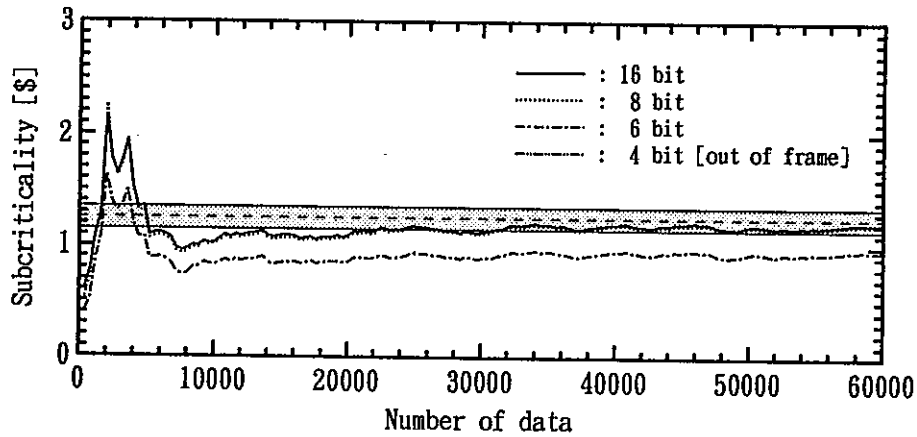
Fig. 4.9 The time series data sampled at 888.888…Hz, 444.444…Hz, and 222.222…Hz.



(a) Sampling frequency = 888.888…Hz



(b) Sampling frequency = 444.444…Hz



(c) Sampling frequency = 222.222…Hz

Fig. 4.10 Dependence of subcriticality estimation on the number of bits for A/D conversion for "ID005 Series DATA".
[Reference subcriticality = 1.25 $.]